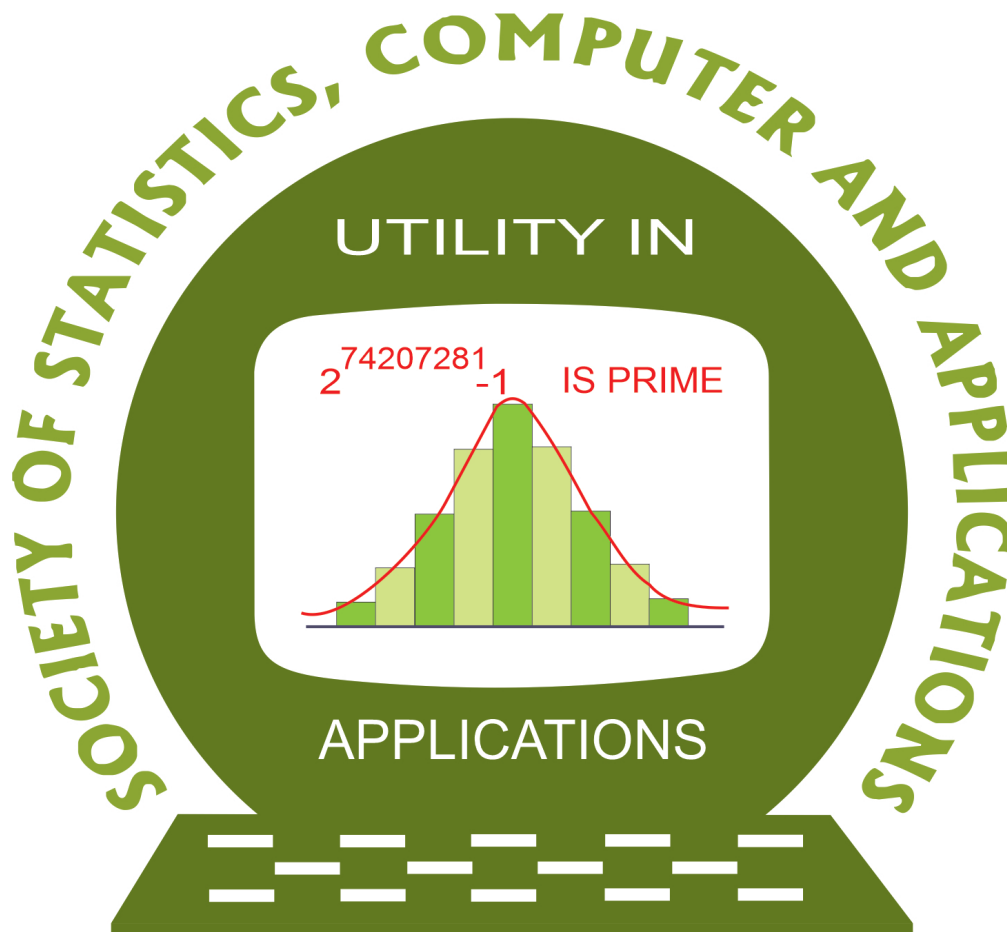


ISSN 2454-7395(online)

STATISTICS AND APPLICATIONS



FOUNDED 1998

Journal of the Society of
Statistics, Computer and Applications

<https://ssca.org.in/journal.html>

Special Issue on Recent Advances in Bayesian Statistics
and Machine Learning

Volume 24, No. 2, 2026 (New Series)

Society of Statistics, Computer and Applications

Council and Office Bearers

President-in-Chief

Late M.N. Das (Founder)

Vinod Kumar Gupta

Presidents

Ashish Das

Rajender Parsad

Executive Presidents

Jyotirmoy Sarkar

Manish Sharma

Manisha Pal

Patrons

A.C. Kulshreshtha

A.K. Nigam

Bikas Kumar Sinha

D.K. Ghosh

K.J.S. Satyasai

Pankaj Mittal

Prithvi Yadav

R.B. Barman

R.C. Agrawal

Rahul Mukerjee

Vice Presidents

A. Dhandapani

P. Venkatesan

Praggya Das

Ramana V. Davuluri

V.K. Bhatia

Secretary

Vishal Deo

Foreign Secretary

Abhyuday Mandal

Treasurer

Ashish Das

Joint Secretaries

Bishal Gurung

Raakhi Singh

Shibani Roy Choudhury

Council Members

Aloke Lahiri

B. Re. Victor Babu

Banti Kumar

Dipak Roy Choudhury

Imran Khan

Mukesh Kumar

P.K. Das

Piyush Kant Rai

Rajni Jain

Ramasubramanian V.

Raosaheb V. Latpate

Shalini Chandra

Sukanta Dash

V.M. Chako

Vishnu Vardhan R.

Ex-Officio Members (By Designation)

Director, ICAR-Indian Agricultural Statistics Research Institute, New Delhi

Editor-in-Chief, Statistics and Applications

Chair Editors, Statistics and Applications

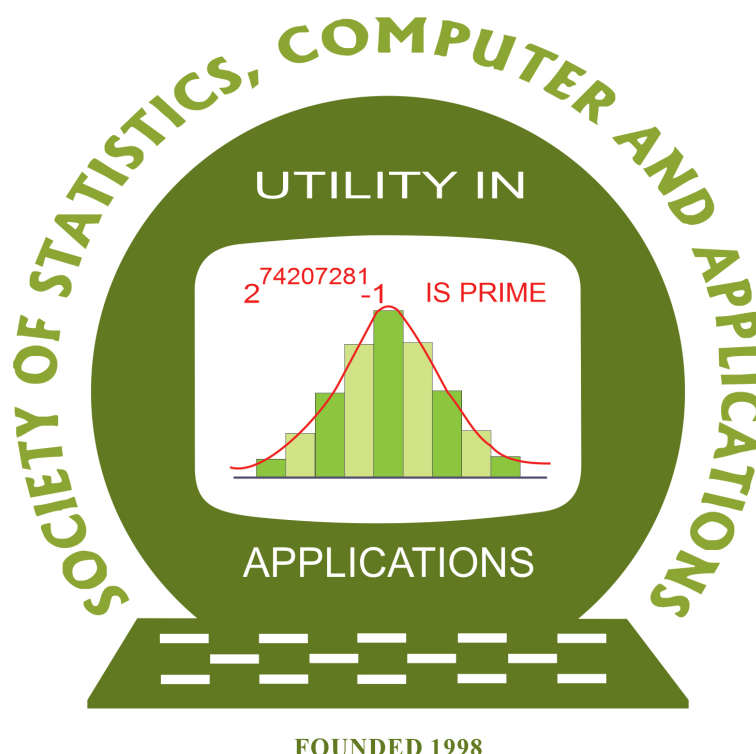
Executive Editors, Statistics and Applications

Society of Statistics, Computer and Applications

Mailing Address: B-133, Ground Floor, Chittaranjan Park, New Delhi-110019, India

Statistics and Applications

ISSN 2454-7395 (online)



**Journal of the Society of
Statistics, Computer and Applications**

<https://ssca.org.in/journal.html>

**Special Issue on Recent Advances in Bayesian Statistics
and Machine Learning
Volume 24, No. 2, 2026 (New Series)**

We are pleased to announce the launch of the official **LinkedIn** page (<https://www.linkedin.com/company/statistics-and-applications-journal-issn-2454-7395-online/>)



and **YouTube** channel (<https://www.youtube.com/@sscassca939>)



of the journal *Statistics and Applications*.

These platforms are intended to facilitate dissemination and engagement with the academic community. The LinkedIn page will provide updates on journal activities, announcements, and access to published issues. The YouTube channel will host video recordings of invited talks, lectures, and other academic events organized under the auspices of the journal. In addition, video presentations corresponding to accepted papers, delivered by the respective corresponding authors, will be made available, and links to these recordings will be included in the abstracts of the published articles. Readers, authors, and researchers are encouraged to visit and follow these platforms.

Statistics and Applications

Special Issue on Recent Advances in Bayesian Statistics and Machine Learning
Volume 24, No. 2, 2026 (New Series)

Editorial Panel

Editor-in-Chief

V.K. Gupta, Former ICAR National Professor at IASRI, Library Avenue, Pusa, New Delhi -110012; vkgupta_1751@yahoo.co.in

Chair Editors

Durba Bhattacharya, Department of Statistics, St. Xavier's College (Autonomous), Kolkata - 700016; durba0904@gmail.com; durba@sxccal.edu

Sourish Das, Data Science Group, Chennai Mathematical Institute, H1 Sipcot IT Park, Siruseri, Chennai 603103, India; sourish.das@gmail.com; sourish@cmi.ac.in

Executive Editors

Rajender Parsad (Senior), ICAR-IASRI, Library Avenue, Pusa, New Delhi - 110012; rajender1066@yahoo.co.in; rajender.parsad@icar.gov.in

Snehanshu Saha, Computer Science and Information System, Head - APPCAIR (All Campuses), BITS Pillani K. K. Birla Goa Campus, India; snehanshus@goa.bits-pilani.ac.in

Sourabh Bhattacharya, Interdisciplinary Statistical Research Unit, Indian Statistical Unit, Kolkata, India; bhsourabh@gmail.com

Editors

Baidya Nath Mandal, Managing Editor, ICAR-Indian Agricultural Research Institute, Hazaribagh-825405, Jharkhand; mandal.stat@gmail.com

Suman Guha, Managing Editor, Department of Statistics, Presidency University, 86/1, College Street, Kolkata 700073, India; bst0404@gmail.com

Jyoti Gangwani, Production Executive, Formerly at ICAR-IASRI, Library Avenue, New Delhi 110012; jyoti0264@yahoo.co.in

Associate Editors

Abhyuday Mandal, Professor and Undergraduate Coordinator, Department of Statistics, University of Georgia, Athens, GA 30602; amandal@stat.uga.edu

Anirban Chakraborti, School of Computational and Integrative Sciences and School of Sanskrit and Indic Studies, Jawaharlal Nehru University, New Delhi 110067, India; anirban.chakraborti@gmail.com

Ashish Das, 210-C, Department of Mathematics, Indian Institute of Technology Bombay, Mumbai - 400076; ashish@math.iitb.ac.in; ashishdas.das@gmail.com

David Banks, Department of Statistical Science, Duke University, Durham, NC27708-0251 USA; david.banks@duke.edu

Deepayan Sarkar, Indian Statistical Institute, Delhi Centre, 7 SJS Sansanwal Marg, New Delhi - 110016; deepayan.sarkar@gmail.com; deepayan@isid.ac.in

Indranil Mukhopadhyay, Professor and Head, Human Genetics Unit, Indian Statistical Institute, Kolkata, India; indranilm100@gmail.com

Janet Godolphin, Department of Mathematics, University of Surrey, Guildford, GU2 7XH, UK; j.godolphin@surrey.ac.uk

Jiani Yin, Associate Director at Servier Biostatistics Project Lead Boston, MA, USA; jianiyin@gmail.com

Jyoti Thanvi, School of Mathematics, University of Birmingham, Dubai Campus, Dubai; jyotithanvi_28@hotmail.com;j.thanvi@bham.ac.uk

Jyotirmoy Sarkar, Department of Mathematical Sciences, Indiana University Purdue University, Indianapolis, IN 46202-3216 USA; jsarkar@iupui.edu

K. Muralidharan, Professor, Department of Statistics, faculty of Science, Maharajah Sayajirao University of Baroda, Vadodara; lmv_murali@yahoo.com

K. Srinivasa Rao, Professor, Department of Statistics, Andhra University, Visakhapatnam, Andhra Pradesh; ksraoau@gmail.com

Lu Chen, NISS-NASS, USDA, USA, Research and Development Division, Sampling and Estimation Research Section; luchan459@gmail.com

M.R. Srinivasan, Visiting Professor, Chennai Mathematical Institute, Siruseri, Chennai-6031035, India; mrsrin8@gmail.com

Murari Singh, 65 Fulbert Crescent, Scarborough, ON M1S1C5, Canada; mandrsingh2010@gmail.com

Pranabendu Mishra, Computer Science Division, CMI, Chennai; pranabendu@cmi.ac.in

Pritam Ranjan, Indian Institute of Management, Indore - 453556; MP, India; pritam.ranjan@gmail.com

R. Vishnu Vardhan, Department of Statistics, Pondicherry University, Puducherry - 605014; vrstatsguru@gmail.com

Ramana V. Davuluri, Department of Biomedical Informatics, Stony Brook University School of Medicine, Health Science Center Level 3, Room 043 Stony Brook, NY 11794-8322, USA; ramana.davuluri@stonybrookmedicine.edu; ramana.davuluri@gmail.com

Rituparna Sen, Indian Statistical Institute Bengaluru, Karnataka 560059, India; ritupar.sen@gmail.com

S. Ejaz Ahmed, Faculty of Mathematics and Science, Mathematics and Statistics, Brock University, ON L2S 3A1, Canada; sahmed5@brocku.ca

Sat N. Gupta, Department of Mathematics and Statistics, 126 Petty Building, The University of North Carolina at Greensboro, Greensboro, NC -27412, USA; sngupta@uncg.edu

Satya Prakash Singh, Department of Mathematics and Statistics, IIT Kanpur, Kanpur 208016, UP, India; sngstypkrsh@gmail.com

Satyaki Mazumdar, Indian Science Education and Research Kolkata, Mohanpur, Nadia-741246, West Bengal; satyaki@iiserkol.ac.in

Saumyadipta Pyne, Health Analytics Network, and Department of Statistics and Applied Probability, University of California Santa Barbara, USA; spyne@ucsb.edu, SPYNE@pitt.edu

Shuvo Bakar, Faculty of Medicine and Health, University of Sydney, Australia; shuvo.bakar@sydney.edu.au

Snigdhasu Chatterjee, School of Statistics, University of Minnesota, Minneapolis, MN -55455, USA; chatt019@umn.edu

Steffano Marchetti, Department of Economics and Management, University of Pisa, Via Ridolfi n, 10, 56124 Pisa, Italy; stefano.marchetti@unipi.it

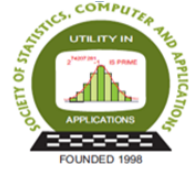
T.V. Ramanathan, Department of Statistics; Savitribai Phule Pune University, Pune; madhavramanathan@gmail.com

Tapio Nummi, Faculty of Natural Sciences, Tampere University, Tampere Area, Finland; tapio.nummi@tuni.fi

Tirupati Rao Padi, Department of Statistics, Ramanujan School of Mathematical Sciences, Pondicherry University, Puducherry; drtrpadi@gmail.com

Utkarsh Tripathi, Solventum (3M Health Care), Pittsburgh Pennsylvania, USA; utkarshbitsp@gmail.com

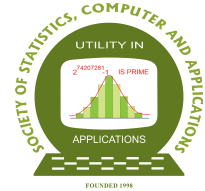
V. Ramasubramanian, ICAR-NAARM, Rajendranagar, Hyderabad, Telangana – 500030; ram.vaidyanathan@gmail.com



CONTENTS

Preface	vii-ix
1. Bayesian Predictive Inference for Multiple Series with Correlated Spatial Priors on Autoregressive Parameters <i>Hammed A. Olayinka and Balgobin Nandram</i>	1–21
2. Hamiltonian Monte Carlo for (Physics) Dummies <i>Arghya Mukherjee and Dootika Vats</i>	23–57
3. On the Choice of Model Space Priors and Multiplicity Control in Bayesian Variable Selection: An Application to Streaming Logistic Regression <i>Joyee Ghosh</i>	59–77
4. Function Optimization with Posterior Gaussian Derivative Process <i>Sucharita Roy and Sourabh Bhattacharya</i>	79–124
5. Neural Architectures for Amortized Bayesian Inference: Statistical Foundations and Empirical Assessments <i>Roy Shivam Ram Shreshth, Arnab Hazra and Gourab Mukherjee</i>	125–154
6. Bayesian Feature Extraction using Gaussian and Diffused-gamma Priors for High Dimensional Spatio-Temporal Data <i>Garrett Frady, Dipak K. Dey and Shariq Mohammed</i>	155–185
7. Bayesian Predictive Inference in Logistic Regression with Covariates and Survey Weights <i>Lingli Yang and Balgobin Nandram</i>	187–210
8. From Classical Probability to Quantum Amplitudes: A Pedagogical Introduction for Statisticians <i>Shyam Dhamapurkar and Sourish Das</i>	211–245

Statistics and Applications {ISSN 2454-7395 (online)}
Special Issue on Recent Advances in Bayesian Statistics
and Machine Learning
Volume 24, No. 2, 2026 (New Series)
<http://www.ssca.org.in/journal>



PREFACE

For many decades, Bayesian statistics—with its impeccable foundation but demanding computational challenges—and machine learning—with its vast repertoire of effective methods and quick computational strategies—operated largely in isolation. In modern times, however, breakthrough computational methods have enriched the Bayesian paradigm, while connections of various machine learning approaches to Bayesian principles have become topics of fervent investigation. Indeed, the intersection of Bayesian statistics and machine learning represents one of the most vibrant and consequential frontiers in modern data science. This special issue of *Statistics and Applications* brings together a remarkable collection of invited papers to celebrate this grand confluence, showcasing methodological and computational innovations that span the full spectrum from fundamental theory to practical implementation across both paradigms.

The eight papers assembled here address challenges that lie at the heart of contemporary Bayesian statistics and machine learning, while also offering a glimpse into the future. The topics range from general function optimization theory, powered by Bayesian sophistication and versatility, to handling intricate spatio-temporal dependence structures including Bayesian feature extraction; from unraveling the mysteries of Hamiltonian Monte Carlo—the robust computational engine driving much of Bayesian computation—to Bayesian variable selection in streaming data; from amortized Bayesian inference with neural architectures, offering a magical blend between the two paradigms, to a new technique for AI explainability; from survey-weighted Bayesian logistic regression to the futuristic world of quantum computation and the esoteric elegance and power of quantum Markov Chain Monte Carlo.

We begin this volume with Roy and Bhattacharya’s “Function Optimization with Posterior Gaussian Derivative Process,” which proposes a Bayesian optimization algorithm that leverages derivative information to achieve almost sure convergence to true optima. The method demonstrates substantially greater accuracy than conventional approaches including Fisher scoring, BFGS, and simulated annealing, across problems ranging from one to one hundred dimensions.

Spatial and temporal dependence structures are addressed by two complementary papers. Olayinka and Nandram, in “Bayesian Predictive Inference for Multiple Series with Correlated Spatial Priors on Autoregressive Parameters,” develop a panel data framework

with conditional autoregressive priors that borrow information across neighboring geographical areas, achieving median RMSE reductions of over 30% relative to non-spatial benchmarks. Frady, Dey, and Mohammed, in “Bayesian Feature Extraction using Gaussian and Diffused-gamma Priors for High Dimensional Spatio-Temporal Data,” present a two-stage feature extraction procedure combining FDR-controlled screening with spatial-temporal clustering, applied successfully to EEG data for identifying brain regions associated with chronic alcohol exposure.

The computational engine of modern Bayesian inference is the focus of Mukherjee and Vats’s “Hamiltonian Monte Carlo for (Physics) Dummies,” a pedagogical review that demystifies HMC for researchers without physics backgrounds by framing it within the Metropolis-Hastings-Green algorithm and elucidating its physical intuition through running examples.

Model selection receives timely attention in Ghosh’s “On the Choice of Model Space Priors and Multiplicity Control in Bayesian Variable Selection,” which compares priors from discrete uniform through Beta-Binomial to the Matryoshka Doll prior in streaming logistic regression, revealing that no single prior dominates across all scenarios.

The amortization of Bayesian inference through neural networks is explored by Shreshth, Hazra, and Mukherjee in “Neural Architectures for Amortized Bayesian Inference,” which examines feedforward networks, Deep Sets, and Transformers as adaptive basis expansions, neural generalizations of the method of moments, and learned kernel smoothers respectively, characterizing their sample efficiency and capacity trade-offs.

Yang and Nandram tackle survey-weighted inference in “Bayesian Predictive Inference in Logistic Regression with Covariates and Survey Weights,” systematically comparing six models with original, trimmed, and calibrated weights, demonstrating that normalized densities with trimmed weights perform best in the presence of outliers.

Finally, Dhamapurkar and Das’s “From Classical Probability to Quantum Amplitudes” provides a self-contained pedagogical bridge from Kolmogorov’s axioms to quantum analogues of conditional probability, Bayes’ rule, expectation, and maximum likelihood. Through the running example of estimating a coin’s bias, the paper explains Grover’s search, Quantum Amplitude Estimation, the HHL algorithm, and Quantum Markov Chain Monte Carlo, while honestly addressing the practical bottlenecks of data loading and readout that determine when quantum speedups translate into real-world advantage.

Taken together, these papers illustrate the remarkable breadth and depth of contemporary Bayesian and machine learning research. They demonstrate that the Bayesian paradigm does not rest on its lofty ideals to the exclusion of other frameworks; rather, it embraces the strengths of modern-day machine learning to launch a two-pronged attack on problems often regarded as insurmountable in the past. The key themes that emerge across the collection—the importance of structured prior specification, the power of hierarchical modeling, the necessity of computational innovation with an eye to the future, embedding deterministic functions and machine learning approaches in broad Bayesian frameworks, and

the value of theoretical guarantees—testify to the enduring relevance of Bayesian thinking in collaboration with machine learning toward reaching the zenith of AI.

We are particularly pleased that this special issue reflects the global nature of the Bayesian and machine learning community, with contributions from researchers based in India, the United States, and perhaps beyond. The diversity of perspectives and approaches represented here is a testament to the vitality of Bayesian wisdom in combination with the machine learning enterprise as an international intellectual endeavor.

We extend our deepest gratitude to the authors for their outstanding contributions, to the reviewers for their careful and constructive evaluations, and to the editorial team of *Statistics and Applications* for their support in bringing this project to fruition. We hope that this collection will serve as both a valuable reference for researchers and an inspiration for future work at the intersection of Bayesian statistics and machine learning.

As the field continues to evolve—driven by new data sources, new computational capabilities, and new scientific questions—the principles and methods showcased in this volume will undoubtedly play a central role. The papers collected here represent not only the state of the art but also signposts for the directions in which our field is heading. We invite readers to engage with these contributions, to build upon them, and to join in the ongoing endeavor to develop statistical methods that are at once principled, powerful, and practical, carrying forward to generations yet to arrive.

August 12, 2026

Guest Editors

Sourabh Bhattacharya

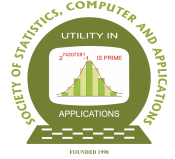
Interdisciplinary Statistical Research Unit, Indian Statistical Institute, Kolkata

Sourish Das

Chennai Mathematical Institute, Chennai

Durba Bhattacharya

St. Xavier's College (Autonomous), Kolkata



Bayesian Predictive Inference for Multiple Series with Correlated Spatial Priors on Autoregressive Parameters

Hammed A. Olayinka and Balgobin Nandram

Department of Mathematical Sciences
Worcester Polytechnic Institute, Worcester, MA 01609

Received: 14 November 2025; Revised: 14 April 2026; Accepted: 16 April 2026

Abstract

We present a Bayesian multiple series model for panel data that improves forecasting by borrowing information across neighboring areas. Each unit is modeled with a simple univariate autoregressive likelihood, while spatial dependence is introduced entirely through a conditional autoregressive prior placed on unit-level intercept and autoregressive coefficient deviations. Inference is conducted using a collapsed Gibbs sampler, which reduces posterior dependence, improves mixing and effective sample size, and preserves interpretable unit-level dynamics. This design avoids the over-parameterization of multivariate formulations while enabling spatially targeted pooling that respects geographic adjacency. We derive stable posterior updates under standard positive-definiteness conditions for the conditional autoregressive precision. Empirically, we study an annual panel of average hourly earnings for production employees across selected California metropolitan statistical areas and compare the proposed model to independent single-series autoregressive models and a pooled multiple time-series model without spatial structure as benchmarks. The proposed approach delivers consistent gains in point accuracy and interval calibration, providing a flexible and interpretable framework for forecasting and small-area estimation in regional economic analysis and related domains. All R codes available at: <https://github.com/haolayinka/MTS-CAR-Model>.

Key words: Gibbs sampler; Hierarchical Bayesian model; Markov-chain Monte Carlo algorithm; Panel forecasting; Pooling time series; Small area estimation.

AMS Subject Classifications: 62M10, 62M30, 62F15, 65C40

1. Introduction

In panel settings, three pragmatic Bayesian strategies are common. First, fit each series independently (*e.g.* Autoregressive of order p ($AR(p)$) per unit), which preserves interpretability but ignores recurring cross-unit patterns and can overfit noise when number of time (T) is small. Second, adopt fully multivariate specifications such as Bayesian

vector autoregressive, which explicitly model cross-sectional dependence but can be over-parameterized for moderate number of areas (ℓ) and modest T , require careful prior design to avoid over-shrinkage, and often entail substantial computation as the dimension grows. Third, pool information through shared hyperparameters or exchangeable hierarchical priors across units (*e.g.*, Nandram and Petrucci (1997)), which stabilizes estimation yet treats units as roughly interchangeable and still does not exploit that nearby areas tend to move together.

Sims (1980) positioned vector autoregressive (VAR) as flexible, data-driven models for multiple time series, but their parameter count grows rapidly with the number of variables and lags, leading to overfitting in small panels and motivating shrinkage or alternative designs - here, adjacency-aware pooling via conditional autoregressive (CAR) priors on univariate autoregressive (AR) likelihoods. A central distinction is between contemporaneous covariance and spatial adjacency. Allowing a non-diagonal cross-sectional covariance in the likelihood captures same-time correlation, but it does not encode where borrowing should occur. In many regional applications, dependence decays with distance and is strongest along shared borders or network links. Models that omit adjacency risk “global” shrinkage - useful on average but potentially blunt - whereas adjacency-aware borrowing can target geographically localized shocks and shared dynamics among neighbors.

We propose a Bayesian multiple time-series (MTS) framework that keeps unit-level dynamics univariate (*e.g.*, $AR(1)$) while introducing cross-sectional structure via CAR priors on unit components. This setup preserves simple likelihoods, enables adjacency-aware pooling across neighbors that is crucial in data-sparse small-area settings, and delivers higher forecast accuracy with tighter uncertainty (narrower Highest Posterior Density Intervals (HPDIs), and lower dispersion). The approach remains computationally light and modular in the choice of spatial weights (adjacency) matrix, W , and it accommodates covariates and time-varying dependence to produce decision-relevant intervals for regional planning.

From a forecasting perspective, design choice matters especially when series are short, noisy, or uneven in length. Unit-wise fits may deliver wide intervals and unstable point forecasts; fully multivariate specifications can be over-parameterized and sensitive to misspecification; and hierarchical pooling without geographic consideration can reduce variance but blur meaningful spatial gradients. The key requirement is a framework that (i) keeps per-series likelihoods simple and interpretable, (ii) borrows strength preferentially across adjacent units, (iii) scales to moderate ℓ and modest T without heavy tuning, and (iv) yields coherent predictive distributions and stable computation.

Our proposed model draws on two established strands. First, univariate dynamic modeling with hierarchical pooling for many series (West and Harrison, 1997; Prado *et al.*, 2021) supports keeping per-series likelihoods simple while sharing information across units. Second, CAR/spatio-temporal(ST)-CAR priors are standard for areal and space-time modeling, providing localized, adjacency-aware smoothing with clear conditional structure (Gelfand and Vounatsou, 2003; Jin *et al.*, 2005; Lee *et al.*, 2014; Rushworth *et al.*, 2014; Lawson, 2018; Desjardins *et al.*, 2020). Recent work strengthens this view of areal panels as collections of time series by combining CAR-based spatio-temporal effects with clustering across areas Mozdzen *et al.* (2022). See Sahu (2021) for a comprehensive treatment of Bayesian spatio-temporal models, including CAR and ST-CAR implementations in R.

In small-area estimation and panel time-series analysis, classic non-spatial pooling offers useful comparators: Fay and Herriot (1979) provide cross-sectional pooling via exchangeable area effects; Binder and Dick (1989) develop modeling and estimation for repeated surveys - effectively panels of univariate series - providing context for pooling across waves; Rao and Yu (1994) incorporate time-series evolution for area effects to borrow across time and domains; and Nandram and Petrucci (1997) propose hierarchical Bayesian pooling for AR processes across units. For a comprehensive overview, see Rao and Molina (2015). These studies differ from our formulation by not encoding geographic adjacency - borrowing is global or exchangeable rather than spatially adaptive. When series form groups or hierarchies, forecast reconciliation ensures coherent aggregation and offers an orthogonal (non-spatial) comparators Girolimetto and Di Fonzo (2024).

Alternative routes to leveraging geography include space-time autoregressive with spatial lags (Shoesmith, 2013) and variational panel spatial autoregressive (SAR) estimators with unrestricted/learned spatial weights (Gefang *et al.*, 2025). Complementary perspectives outside the time-domain Gaussian setting include frequency-domain borrowing via spectral mixtures (Cadonna *et al.*, 2019) and nonparametric error pooling across series (Nieto-Barajas and Quintana, 2016). Bayesian models that keep series univariate yet share information hierarchically provide a non-spatial benchmark with Markov-chain Monte Carlo (MCMC) inference (Müller and Watson, 2025). Spatial Bayesian variants also refine CAR structure, including localized CAR formulations that allow smoothing to switch on/off across the map (Lee *et al.*, 2014). Recent scalable Bayesian space-time forecasting approaches (Orozco-Acosta *et al.*, 2023) provide a complementary spatial baseline focused on short-term prediction.

The proposed MTS-CAR framework does not employ likelihood-based spatial autoregressive structures such as SAR or SARAR, nor does it rely on high-dimensional VARs or black-box machine-learning forecasters. Instead, spatial dependence is introduced exclusively through a conditional autoregressive (CAR) prior on unit-specific coefficients. This enables information sharing across neighboring units while preserving simple, interpretable univariate time-series dynamics. By keeping spatial dependence in the prior rather than in the likelihood, the framework avoids the complexity and potential instability of large VAR systems or spatial lag models, while still capturing meaningful cross-sectional structure in a principled Bayesian way. As such, this study makes three key contributions. First, spatial dependence is introduced entirely through the prior distribution, rather than through the likelihood, which conceptually distinguishes the proposed framework from likelihood-based spatial autoregressive models such as SAR or SARAR. Second, spatial structure is imposed at the coefficient level, specifically on intercepts and autoregressive parameters, allowing adjacency-aware borrowing of information across neighboring units while preserving simple, interpretable univariate dynamics for each time series. Third, we adopt a collapsed Gibbs sampling strategy that is not merely a computational convenience, but a deliberate inferential design choice that reduces posterior dependence, improves mixing, and yields more efficient posterior inference.

Beyond Bayesian methods, graph neural networks learn adjacency-aware spatiotemporal dependencies and are widely used as strong forecasting baselines (Yu *et al.*, 2017), while global deep forecasters train a single model across many series and set competitive non-Bayesian baselines for long-horizon prediction (Nie *et al.*, 2023). Our MTS-CAR model remains in the time domain and encodes adjacency solely through priors on unit-level effects,

yielding targeted spatial smoothing and modular computation.

Beyond proposing the model, we formalize an adjacency-aware Bayesian MTS specification with proper CAR priors on unit-level components, give clear positivity conditions for $D - \rho W$ that guarantee well-posed posteriors, and derive efficient Gibbs updates with conjugate blocks for fast computation. We also provide a clean benchmarking design against SS and non-spatial MTS baselines, report calibration diagnostics, and document robust predictive gains in data-sparse panels. The framework is modular in W and readily accommodate covariates and higher-order AR dynamics via companion-form constraints, offering a practical template for adjacency-aware forecasting in areal panels.

Across 20 areas, our MTS-CAR model delivers a median root mean square error reduction of 30.8% (maximum 50.9%) relative to the intermediate baseline model, MTS, with tighter HPDIs and lower dispersion, while preserving unit-level interpretability. Sensitivity checks indicate that these gains are robust to reasonable choices of W and prior hyperparameters.

The paper proceeds as follows: Section 2 introduces the MTS-CAR framework, detailing the likelihood and hierarchical priors including the CAR prior along with the Gibbs sampling scheme. Section 3 presents the two benchmarks: the MTS and single-series (SS) AR(1) models, along with the Bayesian estimation results and compares predictive performance across the three models. Section 4 concludes the paper with key findings, implications for spatial modeling in panel data, and avenues for future research and extensions.

2. MTS-CAR model

We introduce the proposed Bayesian multiple time series model with CAR priors (MTS-CAR). The specification retains a univariate AR(1) likelihood for each series while inducing spatial borrowing through CAR priors on the area-specific intercept and AR-slope deviations. Section 2.1 details the MTS-CAR model framework, and in Section 2.2, we describe how to run efficient Gibbs sampler with collapsed updates, which is innovative.

2.1. Model framework

For a panel of ℓ areas observed over T periods, we observe a univariate series $\{y_{it}\}_{t=1}^T$ and specify

$$y_{it} = \theta_0 + \nu_{0i} + (\theta_1 + \nu_{1i}) y_{i,t-1} + \varepsilon_{it}, \quad i = 1, 2, \dots, \ell, \quad t = 1, 2, \dots, T, \quad (1)$$

where (ν_{0i}, ν_{1i}) are spatial random effects. Following Jin *et al.* (2005), with a symmetric adjacency matrix W and diagonal degree matrix D , spatial borrowing enters via CAR priors, so the CAR prior is placed on the spatial deviations ν_{0i} (intercepts) and ν_{1i} (AR coefficients).

From (1), let $\boldsymbol{\nu}_1 = (\nu_{01}, \dots, \nu_{0\ell})'$, and $\boldsymbol{\nu}_2 = (\nu_{11}, \dots, \nu_{1\ell})'$, we define the CAR prior structure $\boldsymbol{\nu}_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_1)$ and $\boldsymbol{\nu}_2 \mid \boldsymbol{\nu}_1 \sim \mathcal{N}(A_{\boldsymbol{\nu}_1}, \Sigma_2)$, leading to

$$\boldsymbol{\nu} = \begin{pmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}),$$

with the covariance matrix and the precision matrix define as

$$\Sigma = \begin{pmatrix} \Sigma_1 & \Sigma_1 A' \\ A \Sigma_1 & \Sigma_2 + A \Sigma_1 A' \end{pmatrix}, \quad \Sigma^{-1} = \begin{pmatrix} A' \Sigma_2^{-1} A + \Sigma_1^{-1} & -A' \Sigma_2^{-1} \\ -\Sigma_2^{-1} A & \Sigma_2^{-1} \end{pmatrix},$$

where $\Sigma_s = \tau_s^2 (D - \rho_s W)^{-1}$ for $s = 1, 2$. D is a diagonal matrix containing the row/column sums of W as diagonal elements. For objective-Bayes guidance on propriety and prior specification with intrinsic condition autoregressive (ICAR) components, see Keefe *et al.* (2019). The matrix $A = (m_0 I + m_1 W)$ and $\frac{1}{\lambda_{(1)}} \leq \rho_s \leq \frac{1}{\lambda_{(\ell)}}$, such that $\lambda_{(1)} < \dots < \lambda_{(\ell)}$ are eigenvalues of $D^{-1}W$. Under standard spectral bounds for ρ_s , $D - \rho_s W$ is positive definite, yielding a proper Gaussian prior and stable computations. We show how to obtain the joint distribution of $\underline{\nu} = (\nu_1, \nu_2)'$ in Appendix A (supplement), and the derivation of the covariance matrix in Appendix B (supplement). See Gelfand and Vounatsou (2003) and Jin *et al.* (2005) for more details on multivariate CAR models. We assign flat or non-informative priors to the hyperparameters and latent initial condition, y_{i0} as

$$\pi(\theta_0, \theta_1, \sigma^2, m_0, m_1, \tau_1^2, \tau_2^2, \rho_1, \rho_2) \propto \frac{1}{\sigma^2} \cdot \frac{1}{\tau_1^2} \cdot \frac{1}{\tau_2^2}, \quad y_{i0} \sim \mathcal{N}(\mu_0, \sigma_0^2),$$

where μ_0 , the prior mean and σ_0^2 the prior variance of y_{0i} , are to be specified.

For simplicity, we define $X = (1, y_{i,t-1})'$, $\underline{\theta} = (\theta_0, \theta_1)'$; $Z = (Z_0, Z_1) \in \mathbb{R}^{\ell T \times 2\ell}$, where $Z_0 \in \mathbb{R}^{\ell T \times \ell}$ is an indicator matrix that maps ν_1 to the appropriate rows (1 if the row belongs to unit i , otherwise 0), and $Z_1 \in \mathbb{R}^{\ell T \times \ell}$ has the same pattern as Z_0 but with entries $y_{i,t-1}$ in column i for rows of unit i , mapping ν_2 to the appropriate row. Let $\underline{y} = (y_{1t}, \dots, y_{\ell t})' \in \mathbb{R}^{T \times \ell}$ (stacked over t). Then (1) becomes $\underline{y} = X\underline{\theta} + Z\underline{\nu} + \varepsilon$. By Bayes' theorem, we obtained the joint posterior distribution as

$$\begin{aligned} \pi(\underline{\nu}, \underline{\theta}, \sigma^2, m_0, m_1, \tau_1^2, \tau_2^2, \rho_1, \rho_2, y_0 \mid \underline{y}) &\propto \pi(y_0) \\ &\times \left(\frac{1}{\sigma^2}\right)^{\frac{T\ell}{2}+1} \exp\left\{-\frac{1}{2\sigma^2}(\underline{y} - X\underline{\theta} - Z\underline{\nu})'(\underline{y} - X\underline{\theta} - Z\underline{\nu})\right\} \\ &\times \left(\frac{1}{\tau_1^2}\right)^{\frac{\ell}{2}+1} |(D - \rho_1 W)|^{1/2} \left(\frac{1}{\tau_2^2}\right)^{\frac{\ell}{2}+1} |(D - \rho_2 W)|^{1/2} \exp\left\{-\frac{1}{2}\underline{\nu}'\Sigma^{-1}\underline{\nu}\right\}. \end{aligned} \quad (2)$$

We show the computations of $\det(\Sigma)$ and Σ_s^{-1} in Appendix C and D (supplement), respectively.

2.2. Conditional posteriors and Gibbs updates

Now, we show how to fit the posterior density described in (2) using the Gibbs sampler. We derive the full conditional posterior distributions (CPDs) required for the Gibbs sampler. Let the unknowns parameters be partitioned as $\{\underline{\theta}, \underline{\nu}, \sigma^2, \tau_1^2, \tau_2^2, \rho_1, \rho_2, y_0\}$. For each block we derive the conditional kernel, up to a proportionality constant, with respect to the joint posterior, and identify the sampling family when conjugacy holds. Parameters without closed-form conditionals (*e.g.*, ρ_1, ρ_2) are updated by a fixed grid step.

The initial condition y_{i0} affects the first time step in the autoregressive process. Its CPD is derived from both the likelihood and its normal prior. From (2), when $t = 1$, the

observation equation in area i reads

$$y_{i1} = \theta_0 + \nu_{0i} + (\theta_1 + \nu_{1i})y_{i0} + \varepsilon_{i1}.$$

With a weakly informative Gaussian prior, $\pi(y_{i0})$, the conditional posterior, a Gaussian, is obtained by completing the square as

$$y_{i0} \mid \theta_0, \theta_1, \nu_{0i}, \nu_{1i}, \sigma^2, y_{i1} \sim \mathcal{N}(A^{-1}B, A^{-1}), \quad (3)$$

where

$$A = \frac{1}{\sigma^2}(\theta_1 + \nu_{1i})^2 + \frac{1}{\sigma_0^2}; \quad B = \frac{1}{\sigma^2}(\theta_1 + \nu_{1i})[y_{i1} - (\theta_0 + \nu_{0i})] + \frac{\mu_0}{\sigma_0^2}.$$

The posterior mean is a precision-weighted compromise between (i) the “reverse regression” estimate implied by the $AR(1)$ measurement and (ii) the prior mean μ_0 . When the local persistence is large or the observation noise σ^2 is small, the likelihood contribution dominates; otherwise the prior stabilizes y_{0i} . This update makes the first time row coherent with the area-specific dynamics before the latent spatial effects are drawn.

The observation variance, σ^2 is the model’s noise level, measuring idiosyncratic dispersion around each series’ conditional mean and governing forecast uncertainty. From (2), we have

$$\pi(\sigma^2 \mid \underline{\theta}, \underline{\nu}, m_0, m_1, \tau_1^2, \tau_2^2, \rho_1, \rho_2, y_0, y) \propto \left(\frac{1}{\sigma^2}\right)^{\frac{T\ell}{2}+1} \exp\left\{-\frac{1}{2\sigma^2}(\underline{y} - X\underline{\theta} - Z\underline{\nu})'(\underline{y} - X\underline{\theta} - Z\underline{\nu})\right\}.$$

Let $K = (\underline{y} - X\underline{\theta} - Z\underline{\nu})'(\underline{y} - X\underline{\theta} - Z\underline{\nu})$. Then

$$\sigma^2 \mid \underline{\theta}, \underline{\nu}, m_0, m_1, \tau_1^2, \tau_2^2, \rho_1, \rho_2, y_0, y \sim \text{IG}\left(\frac{T\ell}{2}, \frac{K}{2}\right). \quad (4)$$

We update the common coefficients $\underline{\theta}$ and the area-specific effects $\underline{\nu}$ in a single block using a collapsed-then-conditional scheme. From (2), we have

$$\pi(\underline{\nu}, \underline{\theta} \mid \underline{y}) \propto \exp\left\{-\frac{1}{2\sigma^2}(\underline{y} - X\underline{\theta} - Z\underline{\nu})'(\underline{y} - X\underline{\theta} - Z\underline{\nu})\right\} \exp\left\{-\frac{1}{2}\underline{\nu}'\Sigma^{-1}\underline{\nu}\right\}.$$

We sampled $(\underline{\nu}, \underline{\theta})$ using $\pi(\underline{\nu}, \underline{\theta}) = \pi(\underline{\nu} \mid \underline{\theta}, \sigma^2, y)\pi(\underline{\theta} \mid \sigma^2, y)$. To do this, we marginalize $\underline{\nu}$ and defined $V = \sigma^2 I_n + Z\Sigma Z'$, the conditional posterior is Gaussian

$$\underline{\theta} \mid \underline{y}, \sigma^2 \sim \mathcal{N}(\mu_\theta, \Sigma_\theta), \quad \text{where} \quad \Sigma_\theta = (X'V^{-1}X)^{-1}, \quad \mu_\theta = \Sigma_\theta X'V^{-1}\underline{y}. \quad (5)$$

Conditioning on the newly sampled $\underline{\theta}$, the effects $\underline{\nu}$ are again Gaussian

$$\underline{\nu} \mid \underline{\theta}, \sigma^2, \underline{y} \sim \mathcal{N}(V_0^{-1}V_a, V_0^{-1}), \quad \text{where} \quad V_0 = \left[\frac{1}{\sigma^2}Z'Z + \Sigma^{-1}\right], \quad V_a = \left[\frac{1}{\sigma^2}Z'(\underline{y} - X\underline{\theta})\right]. \quad (6)$$

Sampling $\underline{\theta}$ after integrating out the high-dimensional $\underline{\nu}$ removes the strong posterior dependence between $\underline{\theta}$ and $\underline{\nu}$ that otherwise causes high auto-correlation and slow

exploration. The collapsed likelihood has well-behaved curvature (through V), so θ updates adapt to the uncertainty already induced by the spatial effects. We then condition on θ and draw ν from its conjugate Gaussian CPD. Empirically this collapsed-then-conditional order substantially reduces within-chain auto-correlation for θ while retaining simple, stable linear-algebra for ν . Within each MCMC iteration we jointly treat (θ, ν) as one block updated in two substeps: (i) sample θ from (5) above; (ii) immediately sample ν given this θ . No other parameters are updated in between these two substeps.

The hyperparameters, τ_s^2 sets the overall precision of the latent spatial effects. From (2), we have the terms with τ_s^2 as

$$\pi(\tau_s^2 \mid \theta, \nu, m_0, m_1, \rho_1, \rho_2, y_0, y) \propto \left(\frac{1}{\tau_s^2}\right)^{\ell/2+1} \exp\left\{-\frac{1}{2}\nu'\Sigma^{-1}\nu\right\},$$

expanding $\nu'\Sigma^{-1}\nu$ with $\Sigma_s^{-1} = \frac{1}{\tau_s^2}(D - \rho_s W)$, and after a little algebra we have

$$\pi(\tau_s^2 \mid \theta, \nu, m_0, m_1, \rho_1, \rho_2, y_0, y) \propto \left(\frac{1}{\tau_s^2}\right)^{\ell/2+1} \exp\left\{-\frac{1}{2}\left\{\frac{1}{\tau_s^2}Q_s\right\}\right\}.$$

Thus,

$$\tau_s^2 \mid \theta, \nu, m_0, m_1, \rho_1, \rho_2, y_0, y \sim \text{Inverse} - \text{Gamma}\left(\frac{\ell}{2}, \frac{1}{2}Q_s\right), \quad s = 1, 2. \quad (7)$$

where

$$Q_1 = \nu_1'(D - \rho_1 W)\nu_1, \quad Q_2 = (\nu_2 - A\nu_1)'(D - \rho_2 W)(\nu_2 - A\nu_1).$$

Because $\Sigma_s = \tau_s^2(D - \rho_s W)^{-1}$, the covariance increases linearly with τ_s^2 . Consequently, larger τ_s^2 reduces precision and weakens smoothing (less pooling across areas, more spatial variability), whereas smaller τ_s^2 increases precision and strengthens smoothing (more pooling, smoother latent fields). Under the inverse-gamma conditional posterior for τ_s^2 with a scale proportional to the quadratic form $a_s'(D - \rho_s W)a_s$, larger quadratic forms shift mass toward larger τ_s^2 (weaker smoothing), while smaller quadratic forms shift mass toward smaller τ_s^2 (stronger smoothing).

With an admissible interval $(1/\lambda_{(1)}, 1/\lambda_{(\ell)})$, where $\lambda_{(1)}$ and $\lambda_{(\ell)}$ are the minimum and maximum eigenvalues of $D^{-1}W$, respectively, ρ_s governs the strength and effective range of spatial coupling defined by the adjacency structure. From (2), and using the expansion of $\nu'\Sigma^{-1}\nu$, we have

$$\begin{aligned} \pi(\rho_s \mid \theta, \nu, \sigma^2, m_0, m_1, \tau_s^2, y_0, y) &\propto \\ &|D - \rho_s W|^{1/2} \exp\left\{-\frac{1}{2}\left\{\frac{1}{\tau_s^2}a_s'(D - \rho_s W)a_s\right\}\right\}, \\ &\propto \left(\prod_{i=1}^{\ell} d_{ii}\right)^{1/2} \prod_{k=1}^{\ell} (1 - \rho_s \lambda_k)^{1/2} \exp\left\{-\frac{1}{2}\left\{\frac{1}{\tau_s^2}a_s'(D - \rho_s W)a_s\right\}\right\}, \end{aligned} \quad (8)$$

where $a_1 = \nu_1$, $a_2 = \nu_2 - A\nu_1$, and λ_k is an eigenvalues of $D^{-1}W$. In the marginal likelihood, the log-determinant term penalizes values outside the admissible, stable region, while the

quadratic form rewards values that reduce spatial roughness in a_s . Small ρ_s implies weak dependence and minimal smoothing across neighbors; larger ρ_s induces stronger, longer-range smoothing over the graph, subject to the positive-definiteness constraints determined by the adjacency matrix.

For the coupling parameters (m_0, m_1) , the only dependence on m_0, m_1 is through $A = m_0 I + m_1 W$ inside Σ^{-1} . Let $\underline{m} = (m_0, m_1)'$, from (2), we have

$$\pi(\underline{m} \mid \underline{\theta}, \underline{\nu}, \sigma^2, \rho_s, \tau_s^2, y_0, \underline{y}) \propto \exp\left\{-\frac{1}{2} \underline{\nu}' \Sigma^{-1} \underline{\nu}\right\}.$$

In the expansion of $\underline{\nu}' \Sigma^{-1} \underline{\nu}$, the only term having m_0 and m_1 through A are $(\underline{\nu}_2 - A \underline{\nu}_1)' \Sigma_2^{-1} (\underline{\nu}_2 - A \underline{\nu}_1)$. Expanding this term we have $\underline{\nu}_2' \Sigma_2^{-1} \underline{\nu}_2 - 2 \underline{\nu}_1' A' \Sigma_2^{-1} \underline{\nu}_2 + \underline{\nu}_1' A \Sigma_2^{-1} A' \underline{\nu}_1$. Let $U = m_0 I$ and $B = m_1 W$. Then $\underline{\nu}_1' A \Sigma_2^{-1} A' \underline{\nu}_1 = \underline{\nu}_1' (U' \Sigma_2^{-1} U + U' \Sigma_2^{-1} B + B' \Sigma_2^{-1} U + B' \Sigma_2^{-1} B) \underline{\nu}_1$. Therefore

$$\begin{aligned} (\underline{\nu}_2 - A' \underline{\nu}_1)' \Sigma_2^{-1} (\underline{\nu}_2 - A' \underline{\nu}_1) &= \underline{\nu}_2' \Sigma_2^{-1} \underline{\nu}_2 - 2 \underline{\nu}_1' U' \Sigma_2^{-1} \underline{\nu}_2 - 2 \underline{\nu}_1' B' \Sigma_2^{-1} \underline{\nu}_2 \\ &\quad + \underline{\nu}_1' U \Sigma_2^{-1} U' \underline{\nu}_1 + \underline{\nu}_1' U' \Sigma_2^{-1} B \underline{\nu}_1 + \underline{\nu}_1' B' \Sigma_2^{-1} U \underline{\nu}_1 + \underline{\nu}_1' B' \Sigma_2^{-1} B \underline{\nu}_1. \end{aligned}$$

Substituting $U = m_0 I$ and $B = m_1 W$, and pulling out scalars m_0, m_1 , we have the whole expression as the quadratic polynomial in (m_0, m_1) as

$$Q(m_0, m_1) = m_0^2 a + m_1^2 c + 2m_0 m_1 b - 2m_0 d - 2m_1 e + C,$$

where $a = \underline{\nu}_1' \Sigma_2^{-1} \underline{\nu}_1$, $b = \underline{\nu}_1' \Sigma_2^{-1} W \underline{\nu}_1$, $c = \underline{\nu}_1' W \Sigma_2^{-1} W \underline{\nu}_1$, $d = \underline{\nu}_1' \Sigma_2^{-1} \underline{\nu}_2$, $e = \underline{\nu}_1' W \Sigma_2^{-1} \underline{\nu}_2$, $C = \underline{\nu}_2' \Sigma_2^{-1} \underline{\nu}_2$, $W' = W$. Let $\underline{h} = \begin{pmatrix} d \\ e \end{pmatrix}$, and $G = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, then $Q(m_0, m_1) = \underline{m}' G \underline{m} - 2 \underline{h}' \underline{m} + C$.

Thus we have

$$\pi(\underline{m} \mid \underline{\theta}, \underline{\nu}, \sigma^2, \rho_s, \tau_s^2, y_0, \underline{y}) \propto \exp\left\{-\frac{1}{2} Q(m_0, m_1)\right\},$$

which has the bivariate normal kernel $\exp\left\{-\frac{1}{2} (\underline{m} - \mu)' G (\underline{m} - \mu)\right\}$ with $\mu = G^{-1} \underline{h}$ and $\text{Cov}(\underline{m}) = G^{-1}$. Thus

$$\underline{m} \mid \underline{\theta}, \underline{\nu}, \sigma^2, \rho_s, \tau_s^2, y_0, \underline{y} \sim \mathcal{N}(G^{-1} \underline{h}, G^{-1}), \quad (9)$$

where $\tilde{\Delta} = (\underline{\nu}_1' \tilde{\Sigma} \underline{\nu}_2) (\underline{\nu}_2' W \tilde{\Sigma} W \underline{\nu}_2) - (\underline{\nu}_2' \tilde{\Sigma} W \underline{\nu}_2)^2$, $G^{-1} = \frac{1}{\tilde{\Delta}} \begin{pmatrix} \underline{\nu}_2' W \tilde{\Sigma} W \underline{\nu}_2 & -\underline{\nu}_2' \tilde{\Sigma} W \underline{\nu}_2 \\ -\underline{\nu}_2' \tilde{\Sigma} W \underline{\nu}_2 & \underline{\nu}_2' \tilde{\Sigma} \underline{\nu}_2 \end{pmatrix}$,

$$G^{-1} \underline{h} = \frac{1}{\tilde{\Delta}} \begin{pmatrix} (\underline{\nu}_2' W \tilde{\Sigma} W \underline{\nu}_2) (\underline{\nu}_2' \tilde{\Sigma} \underline{\nu}_1) - (\underline{\nu}_2' \tilde{\Sigma} W \underline{\nu}_2) (\underline{\nu}_2' W \tilde{\Sigma} \underline{\nu}_1) \\ (\underline{\nu}_2' \tilde{\Sigma} \underline{\nu}_2) (\underline{\nu}_2' W \tilde{\Sigma} \underline{\nu}_1) - (\underline{\nu}_2' \tilde{\Sigma} W \underline{\nu}_2) (\underline{\nu}_2' \tilde{\Sigma} \underline{\nu}_1) \end{pmatrix}, \quad \tilde{\Sigma} := (D - \rho_2 W)^{-1}.$$

Under the squared-scale precision parameterization, the common multiplier τ_2^2 cancels from this conditional; hence the mean and precision above are invariant to the change $\Sigma_2^{-1} = \frac{1}{\tau_2^2} (D - \rho_2 W)$. We show the derivation of G^{-1} and $G^{-1} \underline{h}$ in Appendix E (supplement). For the joint distribution of $\underline{m}$ to be well-defined (*i.e.*, G invertible/PD), we need $a > 0$, $c > 0$, $\det(G) = \tilde{\Delta} = ac - b^2 > 0$, which are algebraic conditions on the scalars a, b, c . We prove that these conditions are satisfied almost surely in Appendix F (supplement).

Finally, we made a few numerical remarks. The matrix $D - \rho W$ is positive definite for $\rho \in (1/\lambda_{(1)}, 1/\lambda_{(\ell)})$ (proved in Appendix G (supplement)) and $D - I$ is positive semi-definite, *i.e.* $D \succeq I$, then the CAR precision components are symmetric positive definite (SPD); consequently, the block prior precision Σ^{-1} is SPD as well. We operate only in this admissible region, so the joint prior for the latent effects is well-behaved. In the collapsed updates, we never form large covariance matrices explicitly. instead we apply the Sherman-Morrison-Woodbury identity (Sherman and Morrison, 1950; Woodbury, 1950; Henderson and Searle, 1981; Golub and Van Loan, 2013) to reduce computations to solving linear systems with a single symmetric positive-definite matrix and performing a small 2×2 Cholesky decomposition for the regression block. This keeps the algebra stable, avoids explicit matrix inverses, and yields a fast, numerically reliable implementation.

3. Data analysis

We assess the MTS-CAR model’s performance against the two baselines using a panel of annual average hourly earnings for production employees across $\ell = 20$ California Metropolitan Statistical Areas (MSAs), spanning 2003–2024 (22 years). To encode geography, we build a spatial incidence (adjacency) matrix capturing proximity and inter-regional links among MSAs. Because autoregressive specifications requires stationarity, we applied a square-root transformation to enforce positivity and then performed second-difference of the transformed series (after performing unit root/stationarity test using Augmented Dickey-Fuller (ADF) and Phillips-Perron tests). We do not use the logarithmic transformation because when the logarithm is re-transformed, the Bayesian predictive distribution is not well defined; *e.g.*, see Manandhar and Nandram (2021, 2025) who used generalized gamma distributions instead of the logarithmic transformation. For evaluation, we reserve the final time point in each series as a one-step-ahead holdout and use this left-out observation’s forecast to compare predictive performance across models.

For the panel analysis, the SS model relied on a simple random sample of 1,000 posterior draws. For the MTS model, we ran 75,000 Gibbs iterations, discarded the first 15,000 as burn-in, and thinned by keeping every 60th draw to obtain 1,000 samples. For the MTS-CAR model, we ran 13,000 iterations, dropped the first 3,000, and kept every 10th draw - again yielding 1,000 posterior samples to keep sample sizes comparable across models. The MTS-CAR Gibbs sampler was executed in R-studio on a Windows 11 desktop computer equipped with Intel(R) Core(TM) i7-10700 CPU 2.90GHz processor and 16 GB RAM. A full MCMC run of 13,000 iterations took approximately 48 minutes.

After sampling, we assessed convergence using the Geweke and Gelman-Rubin diagnostic tests for each parameter. We also computed the effective sample size (ESS) to quantify the amount of independent information and examined trace plots to visually evaluate mixing of the Markov chain. For the MTS Gibbs samplers, Geweke p-values ranged from 0.074 to 0.999, and the smallest ESS across parameters was 855, indicating strong convergence. Table 1 presents the ESS value, the Geweke test, and the Gelman-Rubin test of convergence for the hyperparameters in the MTS-CAR model. As shown in the table, the ESS (> 950) for all the hyperparameters are large, and the Geweke test p-value (> 0.05) indicates that these

Table 1: Convergence diagnostics

Parameter	ESS	Geweke-test	PSRF
θ_0	1000	0.814	1.001
θ_1	1694	0.954	1.000
m_0	1100	0.959	0.999
m_1	1064	0.700	1.009
ρ_1	1000	0.377	1.001
ρ_2	959	0.810	1.001
τ_1^2	1000	0.083	1.000
τ_2^2	1000	0.612	1.000
σ^2	984	0.722	0.999

Note: ESS = Effective Sample Size; PSRF = Potential Scale Reduction Factor; the Geweke test p-value was reported.

parameters converges well. The p-value of the Geweke test of the remaining parameters and latent variable (e.g, $\nu_{0i}, \nu_{1i}, y_{0i}$) ranged from 0.111 to 0.985, with the smallest ESS value of 902. As reported in Table 1, the potential scale reduction factor ($PSRF < 1.01$) for all the hyperparameters have excellent convergence. The remaining parameters show excellence convergence ($PSRF \leq 1.01$) as well.

3.1. Predictive evaluation and forecasting accuracy

To evaluate the predictive performance of the MTS-CAR model, we made prediction of the last observation ($t = T$) for each series. These predictions were computed based on posterior draws from the Gibbs sampler. Specifically, using $\hat{y}_{iT|T-1} = \theta_0 + \nu_{0i} + (\theta_1 + \nu_{1i})y_{i,T-1} + \varepsilon_{iT}$, we predicted the last observation for each series $i = 1, \dots, \ell$. We also made predictions using the baseline SS and MTS models.

Since our estimation used square-root-transformed, and second-differenced data, we mapped forecasts back to the original scale via the inverse transformations. Specifically, we first undid the second differencing using the past values $\sqrt{y_{T-1}}$ and $\sqrt{y_{T-2}}$ to recover $\sqrt{y_T}$, and then squared the result to invert the square-root transform. We summarized posterior predictions using the posterior mean (PM), posterior standard deviation (PSD), posterior bias (PB), posterior root mean squared error (RMSE), posterior coefficient of variation (PCV), and highest posterior density interval (HPDI), providing a comprehensive assessment of each forecasting approach.

In addition to posterior predictive summaries, we compute prediction relative errors as

$$R_{iT} = \frac{\hat{y}_{iT} - y_{iT}}{y_{iT}},$$

where $\hat{y}_{iT}$ is the predicted level and y_{iT} is the observed value at the final time point. From these errors, we report summary measures including the posterior error mean (PEM), posterior error standard deviation (PESD), Bayesian standard deviation (BSTD), and the 95% HPDI. These metrics are presented separately for the SS model, the MTS model without spatial coupling, and the proposed MTS-CAR model.

3.2. Model comparisons

Here we introduce the two baseline models used to contextualize the performance of the MTS-CAR model. The first is the SS $AR(1)$ model fitted independently to each unit/MSA, which preserves per-series interpretability but ignores cross-unit structure. The second is the MTS model that pools information across units through exchangeable hierarchical priors on intercepts and AR coefficients, yet imposes no spatial prior. Both baselines generate posterior predictive distributions against which we compare MTS-CAR model in terms of accuracy and calibration.

3.2.1. Single-Series (SS) model

As a baseline, we use a SS $AR(1)$ for a univariate sequence $\{y_t\}_{t=1}^T$. The specification is

$$y_t = \beta_0 + \beta_1 y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2), \quad t = 1, 2, \dots, T.$$

To simplify estimation, the model was described in matrix form as

$$\underline{y} = X\underline{\beta} + \underline{\varepsilon}, \quad \underline{y} \mid \underline{\beta}, \sigma^2 \sim \mathcal{N}(X\underline{\beta}, \sigma^2 I_T),$$

where X is the $T \times 2$ design matrix with a column of ones (intercept) and the lagged values y_{t-1} , $\underline{y} = (y_1, \dots, y_T)'$, $\underline{\beta} = (\beta_0, \beta_1)'$, and I_T denotes the $T \times T$ identity matrix. To complete the Bayesian setup, we assign the following prior distributions to the parameters:

$$\underline{\beta} \mid \sigma^2 \sim \mathcal{N}(\underline{\beta}_0, \sigma^2 \Sigma_0), \quad \sigma^2 \sim \text{Inverse-Gamma}\left(\frac{a_0}{2}, \frac{b_0}{2}\right).$$

where $\underline{\beta}_0$ is the prior mean of $\underline{\beta}$, Σ_0 is its prior scale/covariance, and a_0 and b_0 are the shape and scale, respectively, for the inverse-gamma prior on σ^2 . These choices are the standard conjugate priors for the regression coefficients and the error variance, enabling straightforward Gibbs sampling. The hyperparameters a_0 , and b_0 are fixed a priori, while $\underline{\beta}_0$ and Σ_0 are obtained from ordinary least squares estimate and held fixed during posterior sampling. Because y_0 is required to start the recursion in X , we place the following prior on y_0 :

$$y_0 \sim \mathcal{U}(c_0, d_0), \quad \text{where } c_0 = 0.75 \min(y_t), \quad d_0 = 1.25 \max(y_t).$$

This uniform prior over a scaled range of the observed series offers weakly informative yet plausible support for the initial value y_0 . Applying Bayes' theorem and collecting exponential terms involving $\underline{\beta}$, the joint posterior simplifies to

$$\begin{aligned} \pi(\underline{\beta}, \sigma^2, y_0 \mid \underline{y}) &\propto \left(\frac{1}{\sigma^2}\right)^{\left(\frac{T+a_0+2}{2}+1\right)} \\ &\times \exp\left\{-\frac{1}{2\sigma^2}(\underline{\beta} - \hat{\underline{\beta}}_*)'(X'X + \Sigma_0^{-1})(\underline{\beta} - \hat{\underline{\beta}}_*)\right\} \times \exp\left\{-\frac{1}{2\sigma^2}b_{(y_0)}\right\}, \end{aligned} \quad (10)$$

where

$$b_{(y_0)} = (y - X\hat{\beta})'(y - X\hat{\beta}) + (\hat{\beta} - \beta_0)'X'X(X'X + \Sigma_0^{-1})^{-1}\Sigma_0^{-1}(\hat{\beta} - \beta_0) + b_0.$$

We describe the estimation of (10) via a random sampler in Appendix H (supplement).

3.2.2. Multiple time series (MTS) model

We now consider a multiple time series (MTS) model, where a separate AR(1) process is fitted for each area $i = 1, \dots, \ell$, while introducing hierarchical priors that share hyperparameters across units, thereby borrowing strength through the prior distribution. The model accounts for heterogeneity in intercepts and autoregressive slopes across units by placing hierarchical priors on the parameters. Specifically, let $i = 1, \dots, \ell$ index area and $t = 1, \dots, T$ index time. The observation model is a series-specific AR(1) regression with random intercept, μ_i and slope, b_i ,

$$y_{it} \mid \mu_i, b_i, \sigma^2, y_{i,t-1} \sim \mathcal{N}(\mu_i + b_i y_{i,t-1}, \sigma^2), \quad t = 1, \dots, T, \quad i = 1, \dots, \ell.$$

$$\begin{aligned} \mu_i &\sim \mathcal{N}(\theta_1, \sigma_\mu^2), \\ b_i \mid \mu_i &\sim \mathcal{N}\left(\theta_2 + \rho \frac{\sigma_b}{\sigma_\mu}(\mu_i - \theta_1), (1 - \rho^2)\sigma_b^2\right). \end{aligned}$$

where θ_1 and θ_2 are global mean parameters, $\sigma_\mu^2 > 0$ and $\sigma_b^2 > 0$ are variance hyperparameters, and $\rho \in (-1, 1)$ models correlation between μ_i and b_i . The factor σ_b/σ_μ ensures that ρ is indeed the correlation between μ_i and b_i . Let $\tilde{\beta}_i = (\mu_i, b_i)$ and $\tilde{m}_0 = (\theta_1, \theta_2)'$. The random effects follow a bivariate Normal distribution

$$\tilde{\beta}_i \mid \tilde{m}_0, \sigma_\mu^2, \sigma_b^2, \rho \sim \mathcal{N}_2(\tilde{m}_0, \hat{\Sigma}), \quad \hat{\Sigma} = \begin{pmatrix} \sigma_\mu^2 & \rho\sigma_\mu\sigma_b \\ \rho\sigma_\mu\sigma_b & \sigma_b^2 \end{pmatrix},$$

The initial latent condition is unknown, so we assign a prior:

$$y_{i0} \sim \mathcal{N}(\mu_0, \sigma_0^2), \quad \mu_0, \sigma_0^2 \text{ are obtained from a pre-sample.}$$

Weakly informative or diffuse priors are assumed for the remaining hyperparameters:

$$\pi(\theta_1, \theta_2) \propto 1, \quad \pi(\sigma^2) \propto \frac{1}{\sigma^2}, \quad \pi(\sigma_\mu^2) \propto \frac{1}{(1 + \sigma_\mu^2)^2}, \quad \pi(\sigma_b^2) \propto \frac{1}{(1 + \sigma_b^2)^2}, \quad \rho \sim \text{Unif}(-1, 1).$$

To enable grid sampling on a compact domain, $(0, 1)$, we define and induced marginal priors for τ_μ, τ_b as

$$\tau_\mu := \frac{1}{1 + \sigma_\mu^2}, \quad \tau_b := \frac{1}{1 + \sigma_b^2}, \quad \tau_\mu, \tau_b \in (0, 1).$$

The inverse mapping is

$$\sigma_\mu^2 = \frac{1 - \tau_\mu}{\tau_\mu}, \quad \sigma_b^2 = \frac{1 - \tau_b}{\tau_b}, \quad |J| = \left| \frac{d\sigma^2}{d\tau} \right| = \frac{1}{\tau^2}.$$

Hence,

$$\pi(\tau) \propto \pi(\sigma^2(\tau)) \left| \frac{d\sigma^2}{d\tau} \right| \propto \left(\frac{1}{1 + \sigma^2(\tau)} \right)^2 \cdot \frac{1}{\tau^2} = \tau^2 \cdot \frac{1}{\tau^2} = 1,$$

so $\tau_\mu \sim \text{Unif}(0, 1)$, $\tau_b \sim \text{Unif}(0, 1)$. To write the model in matrix form, we define the $T \times 2$ design matrix $X_i = [\mathbf{1}_T, \mathbf{y}_{i,-T}]$ and write the area-specific observation vector as $\underline{y}_i = (y_{i1}, \dots, y_{iT})'$. Then, the likelihood model can be expressed compactly as

$$\underline{y}_i = X_i \underline{\beta}_i + \underline{\varepsilon}_i, \quad \underline{\varepsilon}_i \sim \mathcal{N}_T(\mathbf{0}, \sigma^2 \mathbf{I}_T).$$

Using Bayes' theorem, the joint posterior density is

$$\begin{aligned} \pi(m_0, \underline{\beta}_i, \sigma^2, \tau_\mu, \tau_b, \rho, y_{0i} \mid \underline{y}_i) &\propto (\sigma^2)^{-(\ell T/2+1)} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{\ell} \|\underline{y}_i - X_i \underline{\beta}_i\|^2\right) \\ &\times |\hat{\Sigma}_{(\tau_\mu, \tau_b, \rho)}|^{-\ell/2} \exp\left(-\frac{1}{2} \sum_{i=1}^{\ell} (\underline{\beta}_i - m_0)' \hat{\Sigma}_{(\tau_\mu, \tau_b, \rho)}^{-1} (\underline{\beta}_i - m_0)\right) \\ &\times \mathbf{1}\{0 < \tau_\mu < 1\} \mathbf{1}\{0 < \tau_b < 1\} \mathbf{1}\{-1 < \rho < 1\}. \end{aligned} \quad (11)$$

We describe the estimation of (11) via a Gibbs sampler in Appendix I (supplement)..

Although the model includes unit-specific parameters μ_i and b_i , it uses exchangeable hierarchical priors to share information across series through common hyperparameters. We regard this MTS specification as an intermediate benchmark between the single-series AR(1) baseline and the MTS-CAR model, which retains the same per-series likelihood but adds a CAR prior to encode cross-area dependence given the observed adjacency graph. This MTS model is the closest to the one described in Nandram and Petrucci (1997).

3.2.3. Empirical results

Here, we report empirical results for the baseline SS and MTS models and the proposed MTS-CAR model. We summarize posterior diagnostics, then compare one-step-ahead predictive accuracy and calibration across models using RMSE, PCV, PRMSE, bias, HPDI coverage, and relative error summaries. We analyze the data for 20 MSAs, but because of space, results for only 10 MSAs are shown in the tables for readability. The results are similar across all MSAs.

In Table 2, we report the posterior predictive summaries for the 10 MSAs under the three models: SS, MTS, and MTS-CAR. For comparison, the table also lists the actual value alongside the posterior predictive metrics. Across MSAs, the MTS-CAR and MTS models generally produce PM values close to the true value, with narrower HPDIs and smaller PSD and PCV, reflecting lower posterior variability and tighter uncertainty bands. Notably, MTS-CAR consistently attains the lowest PSD, BSTD, and PCV, indicating effective use of spatial structure, cross-series pooling, and limited between-area heterogeneity.

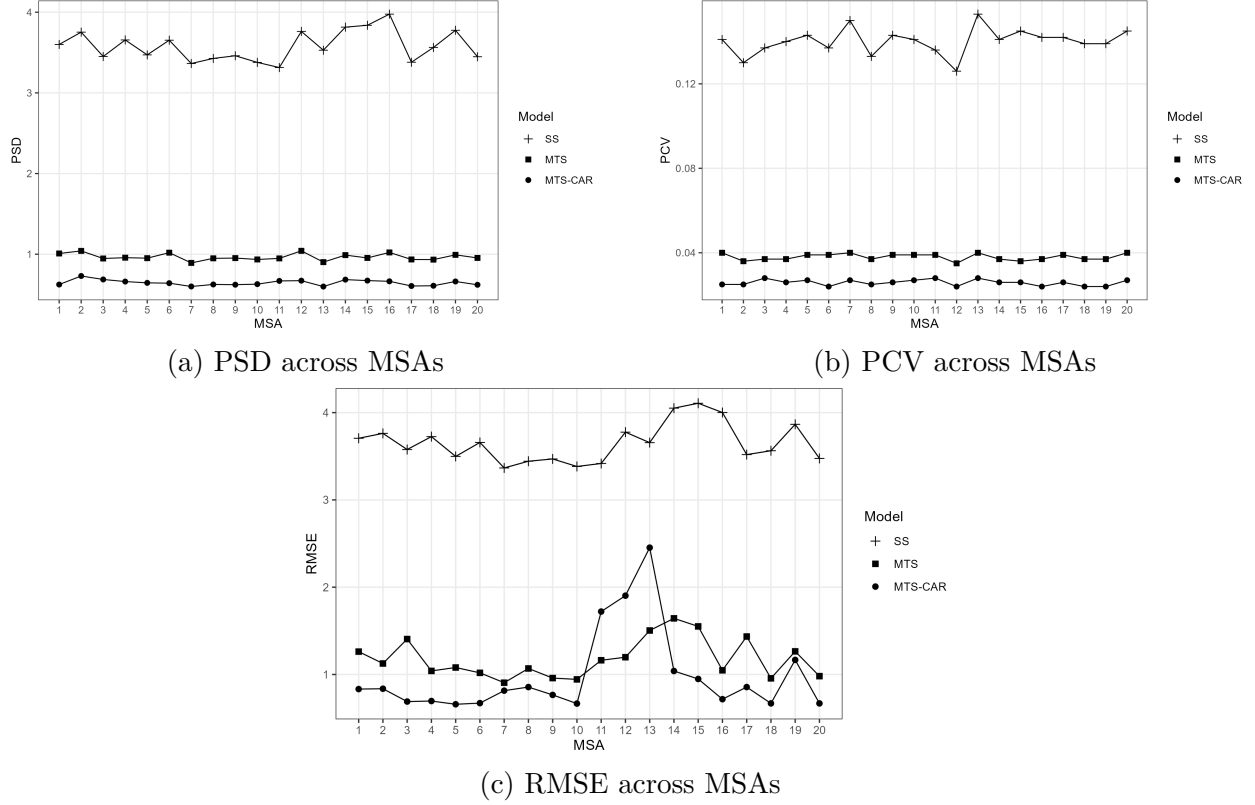


Figure 1: Dispersion, variability, and error across MSAs.

Figure 1(a) and 1(b) illustrate the predictive dispersion and variability (PSD and PCV) across the three models for the 20 MSAs. Both MTS-CAR and MTS show low, stable dispersion, with MTS-CAR model achieving the lowest PSD and PCV across all the MSAs. The MTS model performs well, consistently better than SS, but typically trails the MTS-CAR model. These plots visually corroborate Table 2.

We present the RMSE, PB, and PARB as posterior diagnostic measures in Table 3. The MTS-CAR once again demonstrates strong predictive performance, outperforming the other models in a majority of the MSAs. Specifically, MTS-CAR achieves the lowest RMSE in 10 out of 10 MSAs reported, indicating that our proposed model consistently yields the most accurate forecasts. Additionally, MTS-CAR attains the lowest ARB in 5 out of 10 MSAs reported, with particularly low values like 0.002 in MSA 2, and 0.006 in MSA 4, 0.009 in MSA 3, and 0.027 in MSAs 7 and 8, demonstrating its ability to minimize relative prediction errors. The MTS model occasionally shows competitive ARB (*e.g.*, ARB of 0.003 in MSA 4, 0.004 in MSA 2, and 0.007 in MSA 3), attaining the lowest ARB in 4 out of the 10 MSAs reported. These results reaffirm the robustness of the MTS-CAR model in achieving both low prediction error and bias across a broad range of spatial units.

Figure 1(c) shows that our MTS-CAR model yields lower RMSE than MTS model and is far below SS for most MSAs, with especially large gains where SS errors are highest. The MTS-CAR model exhibits higher RMSE in a few units (*e.g.*, around MSAs 11/13) than the MTS model, indicating heterogeneity in benefits. In MSAs where MTS-CAR does not uniformly outperform the baseline MTS model, the gains from spatial borrowing appear more

Table 2: Posterior predictive summary statistics

MSA	Model	True value	PM	PSD	BSTD	PCV	95% HPDI
1	SS	24.720	25.600	3.600	0.114	0.141	(18.177, 32.294)
	MTS		25.310	0.864	0.027	0.034	(23.631, 26.988)
	MTS-CAR		25.272	0.624	0.020	0.025	(24.104, 26.510)
2	SS	24.340	25.280	3.451	0.109	0.137	(19.062, 32.338)
	MTS		24.430	0.886	0.031	0.036	(22.749, 26.188)
	MTS-CAR		24.399	0.687	0.022	0.028	(23.074, 25.620)
3	SS	25.400	26.110	3.656	0.116	0.140	(19.066, 33.122)
	MTS		25.220	0.866	0.028	0.034	(23.725, 27.093)
	MTS-CAR		25.184	0.661	0.021	0.026	(23.901, 26.477)
4	SS	23.790	24.190	3.471	0.110	0.144	(17.692, 31.006)
	MTS		23.690	0.829	0.027	0.035	(22.018, 25.272)
	MTS-CAR		23.625	0.645	0.020	0.027	(22.208, 24.780)
5	SS	26.440	26.620	3.653	0.116	0.137	(20.224, 34.454)
	MTS		26.170	0.892	0.028	0.034	(24.121, 27.679)
	MTS-CAR		26.239	0.641	0.020	0.024	(25.014, 27.564)
6	SS	25.770	27.130	3.815	0.121	0.141	(19.898, 35.141)
	MTS		26.430	0.901	0.028	0.034	(24.638, 28.068)
	MTS-CAR		26.553	0.685	0.022	0.026	(25.329, 27.993)
7	SS	24.930	26.390	3.838	0.121	0.145	(19.167, 33.748)
	MTS		25.580	0.867	0.027	0.034	(24.024, 27.494)
	MTS-CAR		25.598	0.673	0.021	0.026	(24.355, 26.959)
8	SS	22.760	23.740	3.380	0.107	0.142	(17.586, 31.356)
	MTS		23.420	0.807	0.026	0.034	(21.787, 24.898)
	MTS-CAR		23.365	0.606	0.019	0.026	(22.015, 24.420)
9	SS	26.380	27.210	3.776	0.119	0.139	(20.417, 35.046)
	MTS		27.320	0.902	0.029	0.033	(25.719, 29.224)
	MTS-CAR		27.341	0.662	0.021	0.024	(26.019, 28.599)
10	SS	23.410	23.840	3.448	0.109	0.145	(17.336, 30.793)
	MTS		23.230	0.810	0.026	0.035	(21.634, 24.805)
	MTS-CAR		23.162	0.621	0.020	0.027	(21.964, 24.358)

Note: SS = Single-series, MTS = Multiple time series, MTS-CAR = Multiple time series with CAR prior, PM = Posterior Mean, PSD = Posterior Standard Deviation, BSTD = Bayesian Standard Deviation, PCV = Posterior Coefficient of Variation, HPDI = Highest Posterior Density Interval; Bolded figures denote the smallest values.

heterogeneous, likely reflecting weaker spatial coherence or stronger idiosyncratic dynamics in those regions. This behavior is expected and underscores that the benefits of adjacency-based pooling depend on the strength of underlying spatial similarity. Overall, the pattern corroborates the results in Table 3. Our MTS-CAR model generally improves accuracy relative to MTS model and substantially outperforms the SS model.

In Table 4, the MTS-CAR model frequently exhibited the lowest relative posterior error mean (PEM) and relative posterior error standard deviation (PESD) across almost all the MSAs, highlighting its capacity for precise and stable predictions. For instance, MTS-CAR achieved the lowest PEM in 7 out of the 10 MSAs reported, while MTS had the

Table 3: Prediction diagnostic

MSA	Model	RMSE	PB	ARB
1	SS	3.706	0.882	0.036
	MTS	1.262	0.757	0.024
	MTS-CAR	0.833	0.552	0.022
2	SS	3.577	0.941	0.039
	MTS	1.406	1.039	0.004
	MTS-CAR	0.690	0.059	0.002
3	SS	3.725	0.715	0.028
	MTS	1.041	0.409	0.007
	MTS-CAR	0.696	0.216	0.009
4	SS	3.498	0.430	0.018
	MTS	1.080	0.511	0.003
	MTS-CAR	0.659	0.135	0.006
5	SS	3.658	0.185	0.007
	MTS	0.931	0.267	0.010
	MTS-CAR	0.672	0.201	0.008
6	SS	4.051	1.360	0.053
	MTS	1.118	1.312	0.026
	MTS-CAR	1.040	0.783	0.030
7	SS	4.107	1.462	0.059
	MTS	1.086	0.653	0.026
	MTS-CAR	0.949	0.668	0.027
8	SS	3.519	0.980	0.043
	MTS	1.039	0.656	0.029
	MTS-CAR	0.856	0.605	0.027
9	SS	3.866	0.829	0.031
	MTS	1.301	0.938	0.036
	MTS-CAR	1.167	0.921	0.029
10	SS	3.474	0.429	0.018
	MTS	0.981	0.279	0.012
	MTS-CAR	0.669	0.248	0.010

Note: SS = Single series, MTS = Multiple time series, MTS-CAR = Multiple time series with CAR prior, RMSE = Posterior Root Mean Square Error; PB = Posterior Bias; ARB = Posterior Absolute Relative Bias; Bolded figures denote the smallest value.

lowest PEM in 3. MTS-CAR achieved the lowest PESD in all the 10 MSAs reported. The domination of our MTS-CAR model

suggest robust uncertainty quantification. Across the 20 MSAs, the MTS model offered moderate performance but was consistently outperformed by the MTS-CAR model. Overall, our proposed model provided stable and least dispersed prediction errors, and reliable interval estimates with relatively low error spread.

From a decision-theoretic perspective, tighter highest posterior density intervals and

Table 4: Posterior error summary statistics

MSA	Model	PEM	PESD	BSTD	95% HPDI
1	SS	0.031	0.152	0.005	(-0.242, 0.345)
	MTS	0.023	0.034	0.001	(-0.044, 0.088)
	MTS-CAR	0.022	0.025	0.001	(-0.024, 0.072)
2	SS	0.043	0.146	0.005	(-0.240, 0.330)
	MTS	0.004	0.035	0.001	(-0.063, 0.075)
	MTS-CAR	0.002	0.027	0.001	(-0.050, 0.054)
3	SS	0.019	0.145	0.005	(-0.280, 0.293)
	MTS	0.010	0.034	0.001	(-0.069, 0.066)
	MTS-CAR	0.009	0.027	0.001	(-0.054, 0.053)
4	SS	0.023	0.147	0.005	(-0.280, 0.293)
	MTS	0.003	0.033	0.001	(-0.074, 0.058)
	MTS-CAR	0.005	0.026	0.001	(-0.054, 0.048)
5	SS	0.005	0.139	0.004	(-0.236, 0.323)
	MTS	0.011	0.036	0.001	(-0.081, 0.058)
	MTS-CAR	0.008	0.025	0.001	(-0.057, 0.042)
6	SS	0.054	0.147	0.005	(-0.231, 0.333)
	MTS	0.036	0.036	0.001	(-0.051, 0.090)
	MTS-CAR	0.031	0.027	0.001	(-0.021, 0.085)
7	SS	0.057	0.146	0.005	(-0.234, 0.339)
	MTS	0.028	0.035	0.001	(-0.050, 0.089)
	MTS-CAR	0.027	0.027	0.001	(-0.021, 0.081)
8	SS	0.053	0.148	0.005	(-0.257, 0.317)
	MTS	0.026	0.032	0.001	(-0.038, 0.087)
	MTS-CAR	0.024	0.024	0.001	(-0.025, 0.070)
9	SS	0.041	0.136	0.004	(-0.212, 0.318)
	MTS	0.037	0.036	0.001	(-0.025, 0.114)
	MTS-CAR	0.038	0.027	0.001	(-0.015, 0.090)
10	SS	0.018	0.150	0.005	(-0.312, 0.296)
	MTS	0.007	0.032	0.001	(-0.073, 0.054)
	MTS-CAR	0.010	0.025	0.001	(-0.059, 0.037)

Note: SS = Single-series, MTS = Multiple time series; MTS-CAR = Multiple time series with CAR prior; PEM = relative Posterior error mean; PESD = relative Posterior error standard deviation; BSTD = Bayesian standard deviation; CV = Coefficient of variation; HPDI = Highest Posterior Density Interval; Bolded figures denote the smallest value.

reduced posterior dispersion are especially important in small-area forecasting, where decisions are often made under limited data and uncertainty directly affects resource allocation and planning. By borrowing strength across neighboring units through the prior, the MTS-CAR framework yields more concentrated posterior distributions without sacrificing interpretability. In addition, the collapsed Gibbs sampling strategy reduces posterior dependence among parameters, leading to improved mixing and effective sample sizes, consistent with modern Bayesian workflow principles that emphasize inferential efficiency alongside predictive performance.

Across multiple metrics, the results consistently demonstrate the advantages of our MTS-CAR model. The model repeatedly reduces prediction bias and relative error, reflecting the gains from pooling information across spatial units. The approach balances flexibility with efficiency and, relative to the SS and the MTS baselines, delivers more stable and accurate forecasts. Overall, the findings reinforce the value of hierarchical Bayesian modeling for small-area estimation, highlighting the complementary benefits of spatially adaptive pooled coefficient structures.

4. Concluding remark

This paper tackles forecasting in spatiotemporal settings - data evolving over time and across locations - where each location often has limited observations, as is common in local economic analyses. We propose a new multiple series model (MTS-CAR) that embeds spatially informed priors, enabling it to capture local earnings dynamics while accounting for spillover effects from neighboring regions.

Our MTS-CAR specification places a conditional autoregressive (CAR) prior on the unit-specific intercepts and extends this structure to the coefficient blocks via a shared spatial precision ($D - \rho W$), while the autoregressive coefficients are hierarchically pooled around global means. This setup yields flexible, region-specific dynamics that borrow strength across neighboring areas without over-parameterization. We build on the hierarchical panel AR framework of Nandram and Petrucci (1997) by retaining per-unit AR likelihoods and hierarchical pooling, but replacing exchangeable (global) pooling with adjacency-aware CAR priors on the unit-specific intercept and AR-slope deviations, thereby introducing explicit spatial dependence and coefficient-level regularization (spatially adaptive shrinkage). In contrast to spatial-econometric specifications with a lagged dependent variable in the likelihood (*e.g.*, SAR/Spatial autoregressive with autoregressive disturbances (SARAR); see LeSage and Pace (2009), building on Cliff and Ord (1973), and Anselin (1988)), our model introduces spatial heterogeneity through the prior, achieving adjacency-aware pooling while retaining per-series univariate likelihoods.

To test the effectiveness of our model, we applied it to panel data on earnings across multiple Metropolitan Statistical Areas (MSAs), and compared the results to the baseline models. The empirical results demonstrate that the MTS-CAR framework reliably improves predictive performance in small-area, multi-series settings by borrowing strength across spatially related units. Relative to the autoregressive single series (SS) and multiple time series (MTS) baselines, MTS-CAR delivers tighter uncertainty quantification, lower dispersion, and reduced error, indicating both greater stability and accuracy. The model's out-of-sample performance was impressive. We saw a median reduction of 30.8% (maximum 50.9%) in the root mean square forecast errors relative to the MTS baseline across the 20 areas. These gains arise from explicit spatial pooling that preserves unit-level heterogeneity while exploiting shared structure. Beyond point accuracy, the model concentrates posterior mass more effectively, yielding sharper, decision-relevant intervals. Robustness checks suggest conclusions are not unduly sensitive to reasonable choices of the spatial weights matrix W and prior hyperparameters. Taken together, the evidence supports hierarchical Bayesian modeling with spatial priors as a practical and principled approach for small-area estimation and forecasting.

Conceptually, the proposed MTS-CAR framework differs from likelihood-based spatial autoregressive models by introducing dependence entirely through the prior, rather than through the observation equation. By operating at the coefficient level and leveraging collapsed posterior updates, the model delivers interpretable dynamics, efficient inference, and decision-relevant uncertainty quantification for small-area forecasting.

While our sensitivity checks show robustness to reasonable choices of the spatial weights W and prior hyperparameters, the current specification relies on a fixed W and a time-invariant ρ ; fully dynamic spatial dependence and alternative proximity measures are left for future work. Crucially, the tighter intervals and lower dispersion we obtained are decision-useful. They support clearer budgeting ranges, more credible risk bounds, and sharper targeting for regional labor interventions. The framework is readily transferable to other small-area panels, such as health, education, or local price dynamics, where spillovers and sparse unit-level samples coexist without sacrificing interpretability. Natural extensions include time-varying spatial strength ρ_t , domain-specific W (*e.g.*, commuting, trade, or contiguity hybrids), inclusion of exogenous covariates, principled missing-data handling, and scalable inference for larger ℓ and longer horizons.

Moreover, while we focus on multiple univariate series with adjacency-aware pooling, one avenue for future work is to allow higher-order temporal dynamics while retaining adjacency-aware pooling. Any univariate $AR(p)$ (and, more generally, $VAR(p)$) can be written in companion form as an $AR(1)$ in an expanded state, so stationarity reduces to an eigenvalue check: the spectral radius of the companion (first-order) coefficient matrix must be strictly less than one. This makes it straightforward to monitor or enforce stationarity during posterior simulation (*e.g.*, reject/transform draws that violate the constraint) without altering the CAR prior structure. See Lütkepohl (2005) for the standard treatment of companion representations and stationarity conditions. We leave this multivariate extension to future research.

More broadly, our evidence supports hierarchical Bayesian small-area methods Rao and Molina (2015), Bayesian shrinkage for univariate time-varying parameter models (Belmonte *et al.*, 2014), and Bayesian pooling across many univariate AR processes (Rao and Yu, 1994; Nandram and Petrucci, 1997) as practical tools for forecasting in spatially interconnected, data-sparse settings.

Acknowledgments

The authors are grateful for the invitation to present a paper in the special issue, Recent Advances in Bayesian Statistics and Machine Learning, under the topic, Bayesian methods for time series, spatial statistics, and complex networks.

Conflict of interest

The authors declare no financial or non-financial conflicts of interest related to this work.

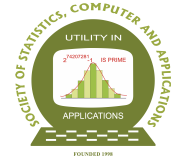
Supplementary materials

All R codes and notes for reproducing the analyses of the MTS and MTS-CAR models in this paper and the supplementary material are available at: <https://github.com/haolayinka/MTS-CAR-Model>

References

- Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Kluwer Academic Publishers.
- Belmonte, M. A., Koop, G., and Korobilis, D. (2014). Hierarchical shrinkage in time-varying parameter models. *Journal of Forecasting*, **33**, 80–94.
- Binder, D. A. and Dick, J. P. (1989). Modelling and estimation for repeated surveys. *Survey Methodology*, **15**, 29–45.
- Cadonna, A., Kottas, A., and Prado, R. (2019). Bayesian spectral modeling for multiple time series. *Journal of the American Statistical Association*, **114**, 1838–1853.
- Cliff, A. D. and Ord, J. K. (1973). *Spatial Autocorrelation*. Pion Limited, London.
- Desjardins, M. R., Hohl, A., and Delmelle, E. M. (2020). Space-time conditional autoregressive modeling to estimate COVID-19 risk. *International Journal of Environmental Research and Public Health*, **17**, 545.
- Fay, R. E. and Herriot, R. A. (1979). Estimates of income for small places: an application of James-Stein procedures to census data. *Journal of the American Statistical Association*, **74**, 269–277.
- Gefang, D., Hall, S. G., and Tavlás, G. S. (2025). Fast two-stage variational Bayesian approach to estimating panel spatial autoregressive models with unrestricted spatial weights matrices. *Spatial Economic Analysis*, 1–26.
- Gelfand, A. E. and Vounatsou, P. (2003). Proper multivariate conditional autoregressive models for spatial data analysis. *Biostatistics*, **4**, 11–15.
- Girolimetto, D. and Di Fonzo, T. (2024). Insights into regression-based cross-temporal forecast reconciliation. In *Scientific Meeting of the Italian Statistical Society*, pages 119–125, Cham. Springer Nature Switzerland.
- Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins University Press, 4 edition.
- Henderson, H. V. and Searle, S. R. (1981). On deriving the inverse of a sum of matrices. *SIAM Review*, **23**, 53–60.
- Jin, X., Carlin, B. P., and Banerjee, S. (2005). Generalized hierarchical multivariate CAR models for areal data. *Biometrics*, **61**, 950–961.
- Keefe, M. J., Ferreira, M. A. R., and Frank, C. T. (2019). Objective Bayesian analysis for Gaussian hierarchical models with intrinsic conditional autoregressive priors. *Bayesian Analysis*, **14**, 181–209.
- Lawson, A. B. (2018). *Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology*. Chapman and Hall/CRC, 3 edition.
- Lee, D., Mitchell, R., and Sahu, S. K. (2014). A Bayesian localized conditional autoregressive model for estimating the health effects of air pollution. *Biometrics*, **70**, 419–429.

- LeSage, J. and Pace, R. K. (2009). *Introduction to Spatial Econometrics*. Chapman and Hall/CRC.
- Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer.
- Manandhar, B. and Nandram, B. (2021). Hierarchical Bayesian models for continuous and positively skewed data from small areas. *Communications in Statistics-Theory and Methods*, **50**, 944–962.
- Manandhar, B. and Nandram, B. (2025). Hierarchical Bayesian models for small area estimation with GB2 distribution. *Journal of Applied Statistics*, **52**, 1–30.
- Mozdzen, A., Cremaschi, A., Cadonna, A., Guglielmi, A., and Kastner, G. (2022). Bayesian modeling and clustering for spatio-temporal areal data: An application to Italian unemployment. *Spatial Statistics*, **52**, 100715.
- Müller, U. K. and Watson, M. W. (2025). Forecasting related time series. Technical report, Department of Economics, Princeton University. Working paper, draft dated October 22, 2025.
- Nandram, B. and Petrucci, J. D. (1997). A Bayesian analysis of autoregressive time series panel data. *Journal of Business & Economic Statistics*, **15**, 328–334.
- Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.
- Nieto-Barajas, L. E. and Quintana, F. A. (2016). A bayesian non-parametric dynamic AR model for multiple time series analysis. *Journal of Time Series Analysis*, **37**, 675–689.
- Orozco-Acosta, E., Arab, A., Cruz, M., and Finley, A. (2023). A scalable approach for short-term disease forecasting in space and time. *Biometrical Journal*, **65**, 2200096.
- Prado, R., Ferreira, M. A. R., and West, M. (2021). *Time Series: Modeling, Computation, and Inference*. Chapman and Hall/CRC, 2 edition.
- Rao, J. N. K. and Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons, 2nd edition.
- Rao, J. N. K. and Yu, M. (1994). Small-area estimation by combining time-series and cross-sectional data. *Canadian Journal of Statistics*, **22**, 511–528.
- Rushworth, A., Lee, D., and Mitchell, R. (2014). A spatio-temporal model for estimating the long-term effects of air pollution on health. *Spatial and Spatio-temporal Epidemiology*, **10**, 29–38.
- Sahu, S. K. (2021). *Bayesian Modeling of Spatio-Temporal Data with R*. Chapman and Hall/CRC.
- Sherman, J. and Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, **21**, 124–127.
- Shoesmith, G. L. (2013). Space–time autoregressive models and forecasting national, regional and state crime rates. *International Journal of Forecasting*, **29**, 191–201.
- Sims, C. A. (1980). Macroeconomics and reality. *Econometrica*, **48**, 1–48.
- West, M. and Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models*. Springer, New York, 2nd edition.
- Woodbury, M. A. (1950). Inverting modified matrices. Technical report, Department of Statistics, Princeton University.
- Yu, B., Yin, H., and Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.



Hamiltonian Monte Carlo for (Physics) Dummies

Arghya Mukherjee and Dootika Vats

*Department of Mathematics and Statistics
Indian Institute of Technology Kanpur*

Received: 04 January 2026; Revised: 25 May 2026; Accepted: 27 May 2026

Abstract

Sampling-based inference has seen a surge of interest in recent years. Hamiltonian Monte Carlo (HMC) has emerged as a powerful algorithm that leverages concepts from Hamiltonian dynamics to efficiently explore complex target distributions. Variants of HMC are available in popular software packages, enabling off-the-shelf implementations that have greatly benefited the statistics and machine learning communities. At the same time, the availability of such black-box implementations has made it challenging for users to understand the inner workings of HMC, especially when they are unfamiliar with the underlying physical principles. We provide a pedagogical overview of HMC that aims to bridge the gap between its theoretical foundations and practical applicability. This review article seeks to make HMC more accessible to applied researchers by highlighting its advantages, limitations, and role in enabling scalable and exact Bayesian inference for complex models.

Key words: Gradient-based sampling; Hamiltonian dynamics; No U-turn sampler; Machine learning; Bayesian computation.

AMS Subject Classifications: 65C05, 62F15, 65C40, 65L06

1. Introduction

Markov chain Monte Carlo (MCMC) methods lie at the heart of Bayesian computation, providing a general mechanism to sample from complex posterior distributions that are analytically intractable. Yet, standard algorithms such as the Metropolis–Hastings (Hastings, 1970; Metropolis *et al.*, 1953) or the Gibbs sampler (Geman and Geman, 1984) often struggle in high dimensions or when model parameters exhibit strong correlations, leading to slow mixing and poor scalability in modern applications. Hamiltonian Monte Carlo (HMC), introduced by Duane *et al.* (1987) and made popular by Neal (2011), has been a breakthrough algorithm providing a sampling solution that overcomes many of these hurdles. A standard HMC algorithm introduces auxiliary variables and employs Hamiltonian dynamics to obtain a next candidate draw. Despite its elegance, the practical use of HMC entails many challenges. Effective deployment requires careful tuning of several hyperparameters that

govern both numerical stability and sampling efficiency. Recent investment in off-the-shelf adapted implementations of HMC, like Stan (Carpenter *et al.*, 2017) and PyMC3 (Salvatier *et al.*, 2016), have made it significantly easier for practitioners to employ adaptive HMC algorithms. On the one hand, this has led to renewed interest in Bayesian modeling strategies across disciplines, while on the other hand, HMC and its variants are often treated as black-box algorithms, limiting the opportunities for practitioners to understand and utilize them effectively.

Recently, a few efforts have been made to exposit the details of an HMC algorithm. Neal (2011) presents a great first introduction, Betancourt (2017) provides elegant explanations providing detailed heuristics and pedagogical motivations. Granados and Bañales (2025) focus on explaining HMC using physics fundamentals, and Thomas and Tu (2021) walk a reader through critical implementation details in R. However, arguably, all works assume a certain amount of knowledge of physics. Despite these excellent works, the authors of this paper still found it challenging to intuitively understand the fine interplay between Hamiltonian dynamics and Monte Carlo sampling. Further, it remained unclear how HMC fits into a traditional Metropolis-Hastings setup. This work presents a pedagogical explanation of HMC, introducing Hamiltonian dynamics in a way that, hopefully, does not require much beyond elementary physics. The intended audience is researchers in statistics and machine learning who are interested in sampling methods, understand the fundamentals of Metropolis-Hastings, but, like the authors of this paper, are “physics dummies”.

The rest of the paper is organized as follows. In Section 2, we present the Metropolis-Hastings algorithm and its Jacobian-based extension, the Metropolis-Hastings-Green method, from which we lay the foundation of HMC in Section 3. An effective HMC implementation requires careful choice of tuning parameters, heuristics and theory for which are outlined in Section 4. Section 5 demonstrates the implementation of HMC for a Bayesian model applied to a real dataset. Our approach in these sections is to keep the discussion fairly pedagogical to ensure clarity. In Section 6 we assemble and present recent developments in HMC and its variants, and we end with a discussion in Section 7.

2. Markov chain Monte Carlo

A typical MCMC problem may be posed in the following way. Interest is in obtaining samples from a distribution with density $\pi(x)$, defined on a state space $\mathcal{X} \subseteq \mathbb{R}^d$. (We employ the common abuse of notation, referring to π as both the density and the distribution.) In many Bayesian applications, π represents the posterior density of the model parameters given the observed data. However, the functional form of π often does not correspond to any known distribution, and it is typically available only up to a normalizing constant. That is,

$$\pi(x) \propto \tilde{\pi}(x),$$

such that $\tilde{\pi}(x)$ is completely known and the normalizing constant is possibly unknown. Consequently, obtaining independent samples from π can be difficult even for moderately large d . MCMC provides a general framework for sampling from such intractable target densities by constructing a Markov chain whose stationary and limiting distribution is π . An appropriately constructed MCMC algorithm would ideally be *ergodic*, in that it is guaranteed to converge to π in total variation distance, irrespective of the starting value of the Markov

chain. Although the construction of such algorithms might seem daunting, due to the path-breaking work of Metropolis *et al.* (1953), this is not so challenging. The challenge is to construct algorithms that exhibit this convergence rapidly.

2.1. Metropolis-Hastings algorithm

The Metropolis algorithm originated from a physics application in the 1950s (Metropolis *et al.*, 1953), and was further extended nearly two decades later by Hastings (1970), thus giving rise to the Metropolis–Hastings (MH) algorithm¹. The MH algorithm constructs a Markov chain using a user-chosen proposal distribution that generates a candidate sample based on the current state. This candidate is then accepted or rejected based on a certain probability that guarantees π as the stationary distribution. The specifics are provided in Algorithm 1, which employs a proposal distribution Q with conditional density $q(\cdot | x)$ to generate a candidate sample x^* . The acceptance function

$$\alpha_{\text{MH}}(x^{(t)}, x^*) = \min \left\{ 1, \frac{\pi(x^*)}{\pi(x^{(t)})} \frac{q(x^{(t)} | x^*)}{q(x^* | x^{(t)})} \right\}, \quad (1)$$

is the MH acceptance function that keeps π invariant for this algorithm.

Algorithm 1 Metropolis–Hastings algorithm

```

1: Initialize: Set initial value  $x^{(0)}$ 
2: for  $t = 0, 1, 2, \dots, T - 1$  do
3:   Draw a proposal  $x^* \sim q(\cdot | x^{(t)})$ , where  $q$  is a proposal density.
4:   Draw  $W \sim \text{Uniform}(0, 1)$  independently.
5:   Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \frac{\pi(x^*)}{\pi(x^{(t)})} \frac{q(x^{(t)} | x^*)}{q(x^* | x^{(t)})}.$$

6:   if  $W \leq \min\{1, r(x^{(t)}, x^*)\}$  then
7:     Set  $x^{(t+1)} = x^*$ 
8:   else
9:     Set  $x^{(t+1)} = x^{(t)}$ 
10:  end if
11: end for
12: Return:  $\{x^{(t)}\}_{t=0}^{T-1}$ .
```

The effectiveness of an MH algorithm relies almost entirely on the choice of the proposal distribution, Q . A “good” proposal should often propose values in high probability areas of π and should also sufficiently explore the tails. Consequently, a “good” proposal

¹More recently, it has been highlighted that much of the laborious programming implementation in the original paper was carried out by Arianna Rosenbluth, and the name should possibly be changed to the Metropolis-Rosenbluth-Teller-Hastings Algorithm. Although we keep the common name in this paper, we direct the reader to Chen (2022), a wonderful article on the contributions of Arianna Rosenbluth.

makes large moves in the state-space, $\mathcal{X}$. A “bad” proposal is one that proposes values close to the current value, yielding low movement of the sampler.

Significant effort has gone into constructing effective sampling strategies. Perhaps the easiest one to implement is the original random walk Metropolis (RWM) of Metropolis *et al.* (1953) where $Q \equiv N(x^{(t)}, h)$ for a tunable $h > 0$. RWM algorithms are often a practitioner’s go-to choice, requiring very little overhead and guaranteeing quick implementation time. However, the proposal struggles to move rapidly in the state-space, particularly in high dimensions, as it utilizes no information from π itself on where high-probability areas might be. This led to the development of powerful gradient-based proposals like the Metropolis Adjusted Langevin Algorithm (MALA) (Roberts and Tweedie, 1996)

$$Q \equiv N\left(x^{(t)} + \frac{h}{2} \nabla \log \pi(x), h \mathbb{I}_d\right), \quad (2)$$

where $h > 0$ is tunable. The MALA proposal moves intentionally towards areas of high probability, learning from the local geometry. The result is an improved exploration of the state-space in high dimensions. HMC is also a gradient-based algorithm that cleverly uses multiple gradients to propose the next move of the sampler. However, HMC does not very elegantly fit into a typical MH scheme and can be viewed as an instance of the general Metropolis-Hastings-Green algorithm.

2.2. Metropolis-Hastings-Green algorithm

Consider a situation when we might want to employ a proposal that is *deterministic*. That is, there is nothing random about the proposal. Certainly, this will be extremely ineffective as the purpose of MCMC sampling is to obtain *random* draws from π . Green (1995), however, identified that one may augment the state-space in such a way that a deterministic proposal in an augmented state-space is in fact random in the original state-space. This is done through the Metropolis-Hastings-Green algorithm with Jacobians (MHGJ)².

Consider a conditional distribution $S(\cdot | x)$ with density $s(\cdot | x)$ defined on $\mathcal{Y}$ such that together with π , this yields the joint density

$$\pi_{\text{Aug}}(x, y) = \pi(x)s(y | x). \quad (3)$$

Since s is a valid density, integrating out y from π_{Aug} will yield our π of interest. In an MHGJ algorithm, the sampler will attempt to keep π_{Aug} invariant, however, we will only be retaining the x samples (in effect, integrating out y). The state-space for the augmented distribution is $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Typically, $S(\cdot | x)$ is chosen in such a way that sampling directly from it is straightforward.

The key idea in MHGJ is then the following: given a state $(x^{(t)}, y^{(t)})$, we first draw a fresh y from the true conditional distribution $S(\cdot | x^{(t)})$, and then apply a *deterministic* mapping on the *random* $(x^{(t)}, y)$, yielding (x^*, y^*) . This new state, (x^*, y^*) , is then accepted

²The Metropolis-Hastings-Green algorithm has found incredible use in transdimensional MCMC algorithms, and its connections to HMC are somewhat less known. Here, we present only what is essential for their use in HMC.

or rejected based on some acceptance probability that is designed to keep π_{Aug} invariant. Since we ignore the y samples, the marginal sampler will be π -invariant. The deterministic mapping here is a central piece and must satisfy certain properties. We denote this function by g and require that $g : \mathcal{Z} \rightarrow \mathcal{Z}$ satisfies $g(g(z)) = z$ for all $z \in \mathcal{Z}$ ³ and the Jacobian of g is everywhere non-singular. We present the MHGJ algorithm in Algorithm 2.

Algorithm 2 Metropolis–Hastings–Green with Jacobians (MHGJ) Algorithm

- 1: **Initialize:** Set initial value $x^{(0)}$
- 2: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
- 3: Draw the auxiliary variable $y \sim S(\cdot | x^{(t)})$ with density $s(y | x^{(t)})$.
- 4: Apply the involution g , and obtain the proposed state

$$(x^*, y^*) = g(x^{(t)}, y).$$

- 5: Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \frac{\pi(x^*) s(y^* | x^*)}{\pi(x^{(t)}) s(y | x^{(t)})} \left| \det \left(\nabla g(x^{(t)}, y) \right) \right|.$$

- 6: Draw $W \sim \text{Uniform}(0, 1)$ independently.
 - 7: **if** $W \leq \min\{1, r(x^{(t)}, x^*)\}$ **then**
 - 8: Set $x^{(t+1)} = x^*$
 - 9: **else**
 - 10: Set $x^{(t+1)} = x^{(t)}$
 - 11: **end if**
 - 12: **end for**
 - 13: **Return:** $\{x^{(t)}\}_{t=0}^{T-1}$ as the Markov chain samples.
-

Algorithm 2 seems to employ an acceptance function noticeably different from standard MH. This is not entirely true, as we will now see. First, we note that Algorithm 2 should be viewed as a composition of two Markov kernels. The first kernel does a Gibbs step in Step 3, which keeps π_{Aug} invariant like any Gibbs full conditional update. The second kernel is comprised of Steps 4–11 for which the input is $(x^{(t)}, y)$. Now, given this input, Step 4 acts as a proposal step where the proposal is indeed *deterministic*. For Step 5, see that,

$$\begin{aligned} r(x^{(t)}, x^*) &= \frac{\pi(x^*) s(y^* | x^*)}{\pi(x^{(t)}) s(y | x^{(t)})} \left| \det \left(\nabla g(x^{(t)}, y) \right) \right| \\ &= \frac{\pi_{\text{Aug}}(x^*, y^*)}{\pi_{\text{Aug}}(x^{(t)}, y)} \left| \det \left(\nabla g(x^{(t)}, y) \right) \right|. \end{aligned} \tag{4}$$

This restructuring makes the acceptance function somewhat similar to (1), where the ratio of the target density at the proposed and the current points also appears. Since the proposal is a Dirac mass, only the Jacobian of the transformation remains in place of the ratio of the proposal densities in (1). See Tierney (1998) for more details on this last comment.

³Such a function is called an involution.

For the algorithm to be correct, g must be involutive, acting as its own inverse, ensuring reversibility of the deterministic proposal mechanism within the MHGJ framework. Glatt-Holtz *et al.* (2026); Neklyudov *et al.* (2020) present a holistic “involutive MCMC” paradigm, which shows that a wide class of MCMC kernels can be expressed via a self-inverse map g acting on an augmented state-space. Subsequent work has extended this idea to non-parametric models (Mak *et al.*, 2022), adaptive involutive map (Liu *et al.*, 2025), and to learned volume-preserving neural involutions (Spanbauer *et al.*, 2020).

Implementing the MHGJ algorithm thus entails specifying a suitable auxiliary distribution S and a deterministic map g . HMC does exactly this by choosing S and g by drawing connections with Hamiltonian dynamics. For most of Section 3, we assume the dimension $d = 1$ for ease of explanation. A general algorithm for a d -dimensional target is discussed in Section 3.3.2.

3. Hamiltonian Monte Carlo

To address computational challenges in lattice quantum chromodynamics, Duane *et al.* (1987) stitched molecular dynamics with MCMC, introducing the “Hybrid Monte Carlo” method. Since its introduction to statistics by Neal (1996) in neural network modeling, “Hybrid Monte Carlo” has become very popular and was later adopted as “Hamiltonian Monte Carlo” as a generic tool for sampling smooth distributions in Bayesian statistical inference. The Hamiltonian approach offers an alternative methodology for describing motion in particle systems and has served as the foundation for the development of HMC in the statistics community. We begin by first describing motion via Hamiltonian dynamics.

3.1. Hamiltonian dynamics

Imagine a ball in a large, frictionless one-dimensional bowl or valley. We may place the ball at any point in the bowl and give it a “push” to initiate its motion. Because the bowl is frictionless, the ball continues to move down and up the bowl according to its initial momentum. After descending into the bowl, it travels up the opposite side, gradually slowing as it climbs. At a sufficiently high point, the ball momentarily comes to rest before sliding back down the side of the bowl. The act of sliding back down corresponds to a change in the sign of the momentum. If the initial “push” is smaller, the highest point the ball can reach in the bowl is correspondingly lower. This behavior arises because the ball moves while conserving its total energy, which is determined jointly by the initial position and the momentum. As the ball moves up the sides of the bowl, its kinetic energy decreases (as its momentum reduces) while its potential energy increases (as its height increases), maintaining a constant total energy. A larger initial push gives the ball greater kinetic energy, allowing it to travel farther up the sides of the bowl.

The reader may further imagine that this system is simple enough that if a physicist were to close their eyes at some point while observing the ball, they would still be able to accurately predict the location of the ball after some time s . This is because the motion of the ball in the bowl follows well-defined laws that can be described using Hamiltonian dynamics. As we proceed in this section, we invite the reader to keep this ball-and-bowl analogy in mind to help build intuition.

Consider, now, the motion of an object on any frictionless surface. At any time t , let x_t denote the position of the particle and let p_t denote its momentum. At any given moment, an object possesses both potential energy and kinetic energy. The potential energy, $U(x_t)$, is determined by the position, and the kinetic energy, $K(p_t)$, is determined by its momentum. In the absence of external forces or friction, the object evolves in such a way that its total energy remains conserved. That is,

$$H(x_t, p_t) := U(x_t) + K(p_t)$$

remains constant as the object moves and is called the Hamiltonian. The actual value of the Hamiltonian is then dictated by the initial condition (x_0, p_0) , where the object started from and with what momentum. The conservation of the Hamiltonian is incredibly useful in determining how the object will move in a given system. Given U and K for a system, the temporal evolution of the object is governed by Hamilton's equations:

$$\frac{dx_t}{dt} = \frac{\partial H(x_t, p_t)}{\partial p_t} \quad \text{and} \quad \frac{dp_t}{dt} = -\frac{\partial H(x_t, p_t)}{\partial x_t}. \quad (5)$$

That is, (5) informs the change in the object's position and momentum with time. Given an (x_0, p_0) , if an object moves according to Hamilton's equations, then the Hamiltonian is conserved and $H(x_0, p_0) = H(x_t, p_t)$ for all t . Further, the solution to the partial differential equations in (5) allows for a functional relationship of the state of the object at any time t .

Running example 3.1. Consider a system where the potential, kinetic, and total (Hamiltonian) energies are expressed as

$$U(x) = \frac{x^2}{2}, \quad K(p) = \frac{p^2}{2}, \quad H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}.$$

Applying Hamilton's equations, we obtain the system of differential equations

$$\frac{dx_t}{dt} = \frac{\partial H(x_t, p_t)}{\partial p_t} = p_t, \quad \frac{dp_t}{dt} = -\frac{\partial H(x_t, p_t)}{\partial x_t} = -x_t. \quad (6)$$

For initial condition (x_0, p_0) , the system of partial differential equations in (6) admits the analytical solution

$$x_t = x_0 \cos t + p_0 \sin t \quad \text{and} \quad p_t = -x_0 \sin t + p_0 \cos t. \quad (7)$$

An analytical solution, as in (7), allows the knowledge of the state of the object at any given time point t . This is eventually what will be used in the HMC sampler.

3.2. Ideal HMC

A key idea in HMC is to assume that the target density corresponds to a potential energy function. Specifically, we set

$$U(x) = -\log \pi(x).$$

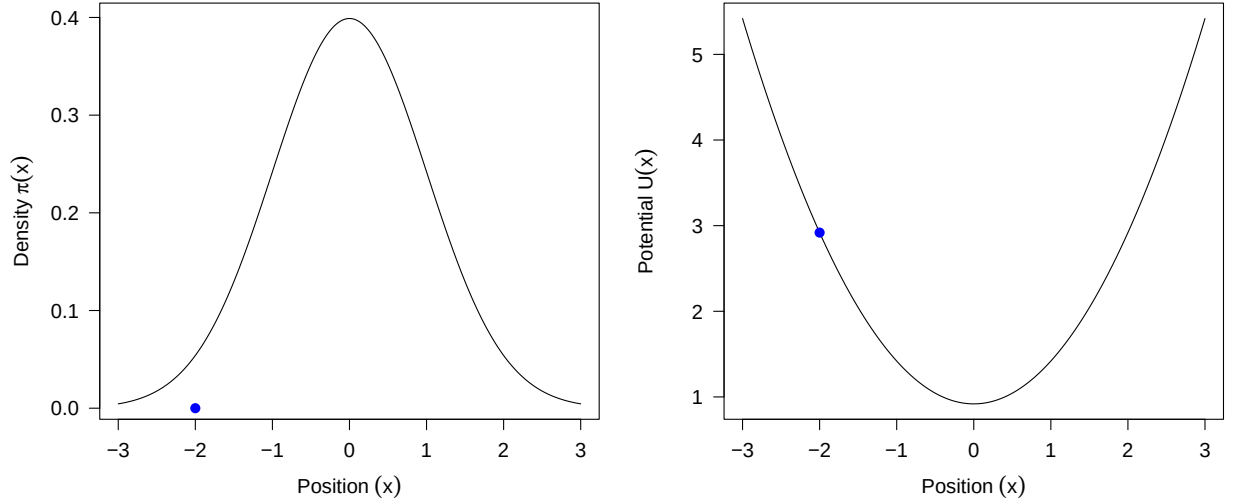


Figure 1: An equivalent representation of Gaussian potential energy.

Then a draw $X = x \sim \pi$ can be interpreted as a state possessing potential energy $U(x)$. Points located in the tails of the distribution correspond to regions of high potential energy, and points in the center of the distribution correspond to having low potential energy; see Figure 1. This might seem counterintuitive, but note that due to conservation of energy, high potential energy in the tails implies low kinetic energy. At a far enough point in the tail, the kinetic energy will be zero, at which point the direction of momentum will flip, making the object travel back to areas of high probability. If the object is assumed to have mass m , the kinetic energy function is

$$K(p) = \frac{p^2}{2m}.$$

Since we are setting up an imaginary system (there is no real object in motion), we may assume any form of kinetic energy; however, the above choice is standard.

Running example 3.2 (Normal distribution). Suppose $\pi \equiv N(0, 1)$, then $U(x) = x^2/2$ and Figure 1 plots this potential energy function. We may assume an associated kinetic energy of $K(p) = p^2/2$ so that $m = 1$. This yields the same Hamiltonian as in Example 3.1:

$$H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}.$$

Given an initial condition (x_0, p_0) , the object will evolve according to the Hamiltonian dynamics (6) conserving energy in its path. Since an analytical solution is available in (7), we can draw the path of the object as time evolves, as a function of x and p . In Figure 2, the trajectory illustrates how the system evolves when initialized at $x_0 = 2$ and $p_0 = -2$. The particle begins its motion from this state and moves along a circular trajectory in the (x, p) plane, conserving its total energy throughout. Because the total energy is preserved for a given initial condition, different starting values of (x_0, p_0) correspond to distinct energy contours. In Figure 3, we consider a second trajectory with the same starting position, x_0 , but a different initial momentum, $p_0 = -1$. This can be thought of as putting an object at the same initial position but with a different initial momentum. In this case, the system

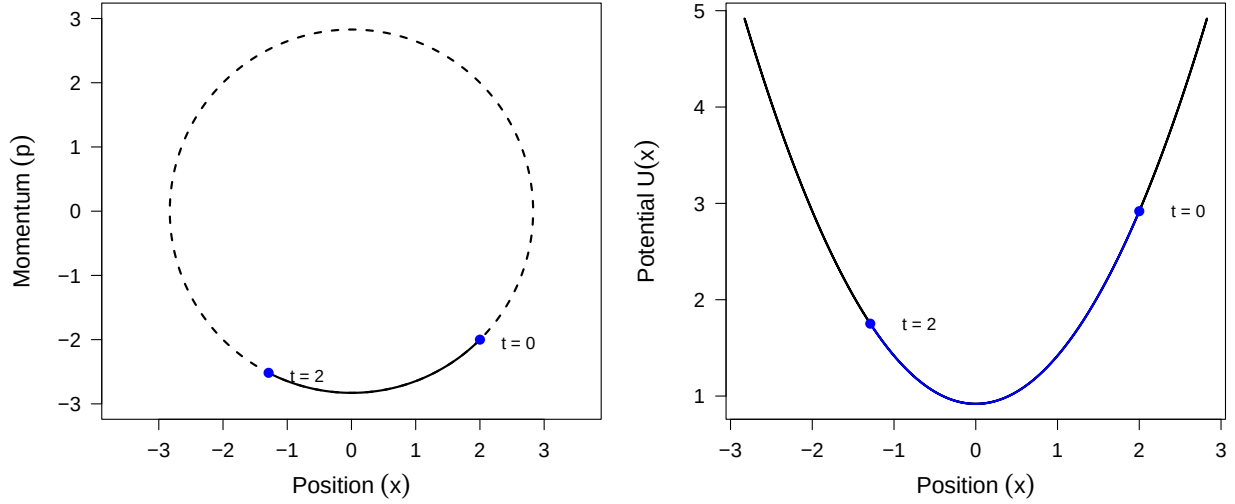


Figure 2: Hamiltonian dynamics under a Gaussian potential. The left panel depicts the trajectory of the particle in the (x, p) phase space, starting from $(x_0, p_0) = (2, -2)$ and its state after $t = 2$ time units. The right panel shows the potential energy function $U(x) = -\log \pi(x)$, illustrating the quadratic bowl corresponding to $\pi \equiv N(0, 1)$.

evolves along a separate energy contour.

We now present the ideal HMC algorithm, building upon the theoretical foundation of Hamiltonian dynamics established so far. The ideal HMC is essentially an MHGJ algorithm with a specific choice of auxiliary variable and involution. Here, the joint target density π_{Aug} is defined as

$$\pi_{\text{Aug}}(x, p) \propto e^{-H(x, p)} = e^{-U(x)} e^{-K(p)} \propto \pi(x) e^{-K(p)}. \quad (8)$$

The Hamiltonian curves in the (x, p) phase-space are the level curves of H , indicating points of equal value of the Hamiltonian; see the left plot of Figure 3. The ideal HMC sampler technically will sample from π_{Aug} , except the momentum samples are typically not saved. Notice here that by assumption, the marginal distribution of the momentum variable is $N(0, m)$, which is independent of the position.

The next ingredient to build an MHGJ sampler is to identify the involution; this is where the Hamiltonian dynamics are employed. Let $T_s(x_t, p_t)$ denote the progression of the object after s time units, starting from (x_t, p_t) . That is, let

$$(x_{t+s}, p_{t+s}) = T_s(x_t, p_t).$$

The map T_s has some nice properties. Namely,

1. If $s = 0$, the particle stays where it is. That is, $T_0(x_t, p_t) = (x_t, p_t)$.
2. The mapping is additive in time. That is, $T_{s_1+s_2}(x_t, p_t) = T_{s_1}(T_{s_2}(x_t, p_t)) = T_{s_2}(T_{s_1}(x_t, p_t))$.
3. The inverse of T_s is the mapping that goes back in time. That is, $T_s^{-1}(x_t, p_t) = T_{-s}(x_t, p_t)$.

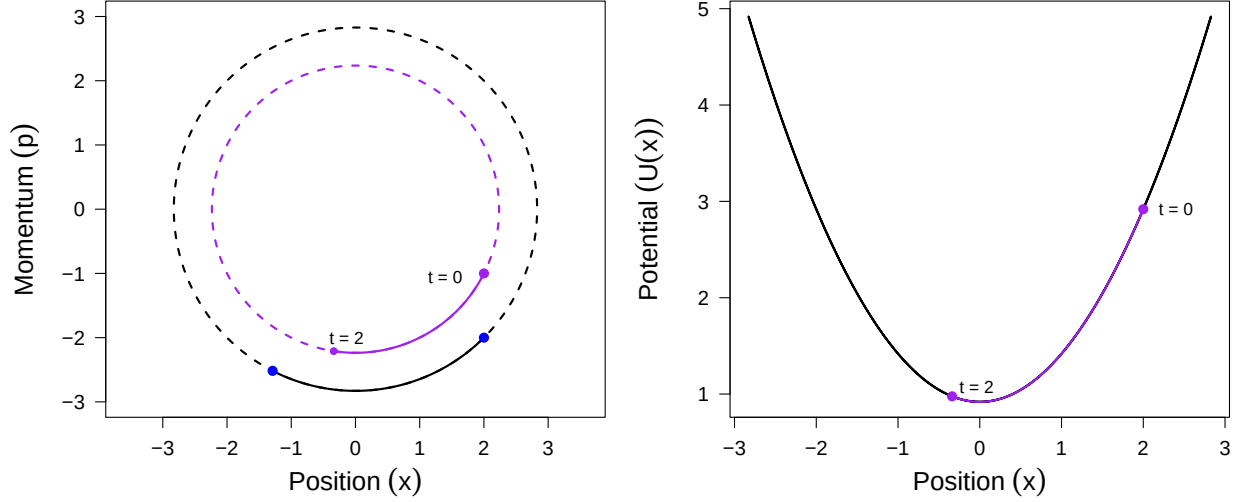


Figure 3: The left panel shows two distinct energy contours in the (x, p) space corresponding to different initial momenta. The right panel illustrates the particle's position after two time units for the second trajectory.

Despite these nice properties, the mapping T_s itself is not an involution and must be used in conjunction with another map. Define the map that reverses the momentum of the object as

$$i(x_t, p_t) := (x_t, -p_t),$$

called the momentum reversal map. Clearly, the mapping i is an involution. When the kinetic energy function is chosen to be symmetric about zero, that is $K(p) = K(-p)$, then together, T_s and i satisfy time-reversal symmetry which states that flipping the momentum of a system, moving the object s time units, and then flipping the momentum again is equivalent to going back in time s units. That is,

$$i \circ T_s \circ i(x_t, p_t) = T_{-s}(x_t, p_t).$$

Consider now the following composition map that pushes the object forward s time units and then reverses the momentum:

$$P_s(x_t, p_t) := i \circ T_s(x_t, p_t) = i(T_s(x_t, p_t)) = (x_{t+s}, -p_{t+s}). \quad (9)$$

The mapping P_s will be the final mapping that we will consider in the MHGJ algorithm. As it turns out, even though T_s is not an involution, P_s is. To see this, note that

$$\begin{aligned} P_s(P_s(x_t, p_t)) &= i(T_s(i(T_s(x_t, p_t)))) \\ &= T_{-s} \circ T_s(x_t, p_t) \\ &= T_s^{-1} \circ T_s(x_t, p_t) \\ &= (x_t, p_t). \end{aligned}$$

Thus, P_s is an involution. Moreover, since T_s is volume-preserving (Jordan, 2004, Liouville's theorem) and the Jacobian of the momentum flip i is a block diagonal matrix with

determinant ± 1 , therefore,

$$\left| \det \nabla P_s(x, p) \right| = 1, \quad (10)$$

showing that P_s is volume-preserving as well. Intuitively, volume-preservation implies that the volume of a region in the (x, p) space remains the same after P_s acts on it. Finally, since $K(p)$ is an even function of p , we have

$$H(x, -p) = U(x) + K(-p) = U(x) + K(p) = H(x, p),$$

ensuring energy invariance under momentum reversal. Using these ingredients, namely

- P_s is an involution,
- P_s is volume-preserving, and
- $K(p)$ is symmetric around zero,

we can run an MHGJ algorithm, yielding an ideal HMC algorithm in Algorithm 3. Specifically, Step 4 draws the auxiliary variable from its conditional distribution given the current iterate x ; in this instance, the conditional distribution is independent of x . In Step 5, we apply the involution P_s on (x, p) . Note here that this is akin to evolving the system using the Hamiltonian mapping T_s and then reversing the momentum. Since $K(p) = K(-p)$, reversing the momentum does not change the outcome of the algorithm and thus is not necessary during implementation. Step 6 calculates the acceptance ratio, which is deterministically 1, since in Step 5 (x^*, p^*) was obtained so as to conserve the Hamiltonian. Thus, the ideal HMC algorithm yields perfect acceptance and no rejections! Finally, since the momentum is resampled in Step 4, there is no need to save the momentum variables.

Algorithm 3 Ideal Hamiltonian Monte Carlo (HMC)

- 1: **Initialize:** Set initial value $x^{(0)}$.
- 2: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
- 3: Given current state $x^{(t)}$.
- 4: Draw momentum $p \sim N(0, m)$.
- 5: Evolve the system for time s to obtain $(x^*, -p^*) = T_s(x^{(t)}, p)$.
- 6: Compute the Metropolis-Hastings ratio:

$$r(x^{(t)}, x^*) = e^{-H(x^*, p^*) + H(x^{(t)}, p)} = 1.$$

- 7: Set $x^{(t+1)} = x^*$.
 - 8: **end for**
 - 9: **Return:** $\{x^{(t)}\}_{t=0}^{T-1}$.
-

Perfect acceptance does not necessarily imply an excellent Markov chain. This is known for Gibbs samplers, which, although also having perfect acceptance, can exhibit slow mixing in various situations (Roberts and Sahu, 1997). The following running example will explain this point a bit more.

Running example 3.3. We continue with the example of $\pi \equiv N(0, 1)$ and $m = 1$ where recall

$$H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}.$$

From (7), we have

$$T_s(x, p) = \begin{pmatrix} x \cos s + p \sin s \\ -x \sin s + p \cos s \end{pmatrix} \quad \text{and} \quad P_s(x, p) = \begin{pmatrix} x \cos s + p \sin s \\ x \sin s - p \cos s \end{pmatrix}.$$

Using the above involution P_s , the ideal HMC in Algorithm 3 can be implemented. Below is an R function that implements the ideal HMC for the standard Gaussian target.

```

1 # ideal HMC function
2 normal_idealHMC <- function(s = 1, n = 1e4)
3 {
4   x <- numeric(length = n)
5   x[1] <- 0 # starting at zero position
6   # don't need a starting value of p
7
8   for(k in 2:n)
9   {
10    p <- rnorm(1) # new momentum
11
12    # simulating Hamiltonian dynamics forward s time units
13    # no practical need to flip momentum
14    x[k] <- x[k-1]*cos(s) + p*sin(s)
15    p <- -x[k-1]*sin(s) + p*cos(s)
16  }
17  return(x) # not returning momentum
18 }
```

Figure 4 presents density estimates, trace plots, and autocorrelation function plots for three different choices of s . Noticeably, some choices of s are better than others, and perfect acceptance does not guarantee excellent mixing of the chain. In this implementation, $s = 1, 5$ yield well-behaved Markov chains, whereas $s = .1$ yields a significantly slower mixing Markov chain. This is because with $s = .1$, the object has not yet had the opportunity to travel far away from the current location, yielding smaller jump sizes, and thus higher autocorrelation.

The ideal HMC algorithm is called as such since, for most realistic examples, an analytical expression for T_s is not available. Thus, the involution cannot be applied directly. This roadblock leads to the widely accepted HMC algorithm, where the Hamiltonian dynamics are time-discretized to yield an approximation to T_s .

3.3. Hamiltonian Monte Carlo algorithm

In practical implementations, the map T_s is not available in closed-form, and Hamilton's equations must be approximated through numerical integration by discretizing time.

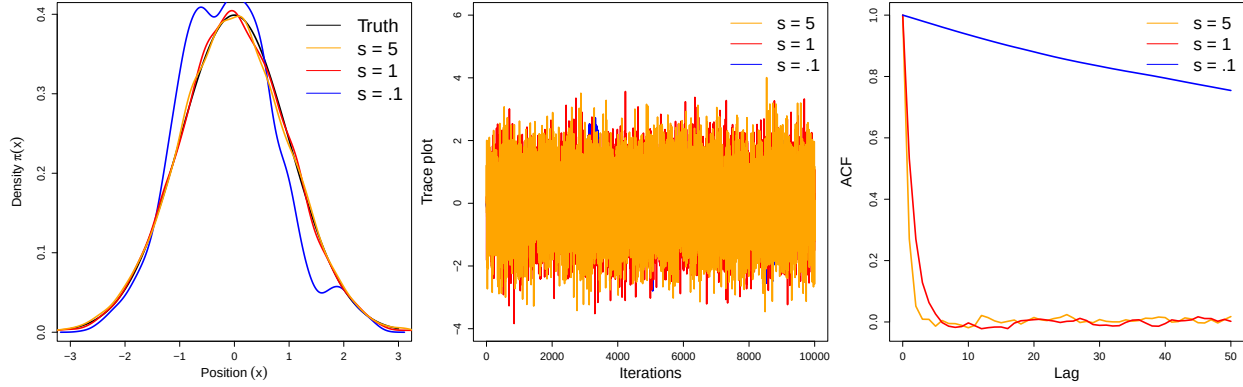


Figure 4: Density plots (left panel), trace plots (middle panel), and ACF plots (right panel) for ideal HMC on Gaussian target density with $s = 0.1, 1, 5$.

This yields an approximation to T_s that is used instead of T_s in Algorithm 3 to obtain the proposal x^* . Recall Hamilton's equations

$$\frac{dp_t}{dt} = -\frac{\partial H(x_t, p_t)}{\partial x_t} = -\nabla U(x_t), \quad \frac{dx_t}{dt} = \frac{\partial H(x_t, p_t)}{\partial p_t} = \nabla K(p_t) = \frac{p_t}{m}.$$

When a solution is analytically unavailable, a numerical integrator approximates dt with an $\varepsilon > 0$. Such a discretization moves the object approximately according to Hamilton's equations for $s = \varepsilon$ time units. As was seen in Figure 4, too small an s can yield a slow mixing Markov chain. Thus, such ε -step discretizations are applied L times so as to get $s = L\varepsilon$. However, any discretizer would only approximately be able to satisfy Hamilton's equations and thus would not be able to conserve the Hamiltonian exactly. Consider a standard Euler discretization:

$$p_{t+\varepsilon} = p_t - \varepsilon \nabla U(x_t) \quad \text{and} \quad x_{t+\varepsilon} = x_t + \varepsilon \nabla K(p_{t+\varepsilon}) = x_t + \varepsilon \frac{p_t}{m}. \quad (11)$$

As $\varepsilon \rightarrow 0$, consistent discretizers like the Euler discretization in (11) will yield improved conservation of the Hamiltonian. However, this does not guarantee long-time stability under repeated composition. That is, repeated application of ε -steps L times worsens the quality of the approximation. This has many consequences. First, any integrator will replace T_s with an approximation, $T_{L,\varepsilon}$ in Algorithm 3. In order for the resulting algorithm to still yield a π -invariant Markov chain, $T_{L,\varepsilon}$ and the resulting $P_{L,\varepsilon}$ must satisfy critical properties required by the MHGJ algorithm. That is, we require that

- $P_{L,\varepsilon}$ is also an involution, and
- $P_{L,\varepsilon}$ is also volume preserving.

The standard Euler discretizer violates both these properties. However, a popular discretizer that guarantees both these properties is the leapfrog integrator.

3.3.1. Leapfrog integrator

The leapfrog integrator introduces a staggered update sequence in which the momentum p is offset by $\varepsilon/2$ time units relative to the position x , yielding a scheme that forms the foundation of practical HMC implementations. Specifically, one jump of the leapfrog integrator first does an $\varepsilon/2$ -Euler jump for p , followed by an ε -Euler jump of x using the most updated state of p , with one last $\varepsilon/2$ -Euler jump of p . We can write a single leapfrog update over one step of size ε as:

1. Half-step update for momentum:

$$p_{t+\frac{\varepsilon}{2}} = p_t - \frac{\varepsilon}{2} \nabla U(x_t). \quad (12)$$

2. Full-step update for position:

$$x_{t+\varepsilon} = x_t + \varepsilon \frac{p_{t+\varepsilon/2}}{m}. \quad (13)$$

3. Final half-step update for momentum:

$$p_{t+\varepsilon} = p_{t+\frac{\varepsilon}{2}} - \frac{\varepsilon}{2} \nabla U(x_{t+\varepsilon}). \quad (14)$$

This procedure approximately advances the system by $s = \varepsilon$ time units. To simulate a longer trajectory of duration $s = L\varepsilon$, one simply performs L successive leapfrog steps, repeating (12)–(14). The final (ε, L) leapfrog integrator steps are

$$\begin{aligned} p_{t+\frac{\varepsilon}{2}} &= p_t - \frac{\varepsilon}{2} \nabla U(x_t), \\ x_{t+\varepsilon} &= x_t + \varepsilon \frac{p_{t+\varepsilon/2}}{m}, \\ p_{t+\frac{3\varepsilon}{2}} &= p_{t+\varepsilon/2} - \varepsilon \nabla U(x_{t+\varepsilon}), \\ &\vdots \\ x_{t+L\varepsilon} &= x_{t+(L-1)\varepsilon} + \varepsilon \frac{p_{t+(2L-1)\varepsilon/2}}{m}, \\ p_{t+L\varepsilon} &= p_{t+(2L-1)\varepsilon/2} - \frac{\varepsilon}{2} \nabla U(x_{t+L\varepsilon}). \end{aligned} \quad (15)$$

The leapfrog integrator is a *symplectic* integrator, implying both volume-preservation and the fact that repeated ε steps do not yield poorer approximations. This is evidenced in Figure 5 where $L = 20$ repeated ε -steps do not yield a systematic propagation of error. Further, it remains true that for $\varepsilon \approx 0$, the approximate path of the object stays closer to the true Hamiltonian than that for larger ε .

The leapfrog scheme in (15) thereby defines a mapping $T_{L,\varepsilon}(x, p)$ that approximates the exact Hamiltonian flow $T_s(x, p)$, where $s = L\varepsilon$. As before, we define

$$P_{L,\varepsilon}(x, p) = i(T_{L,\varepsilon}(x, p)), \quad \text{where recall } i(x, p) = (x, -p). \quad (16)$$

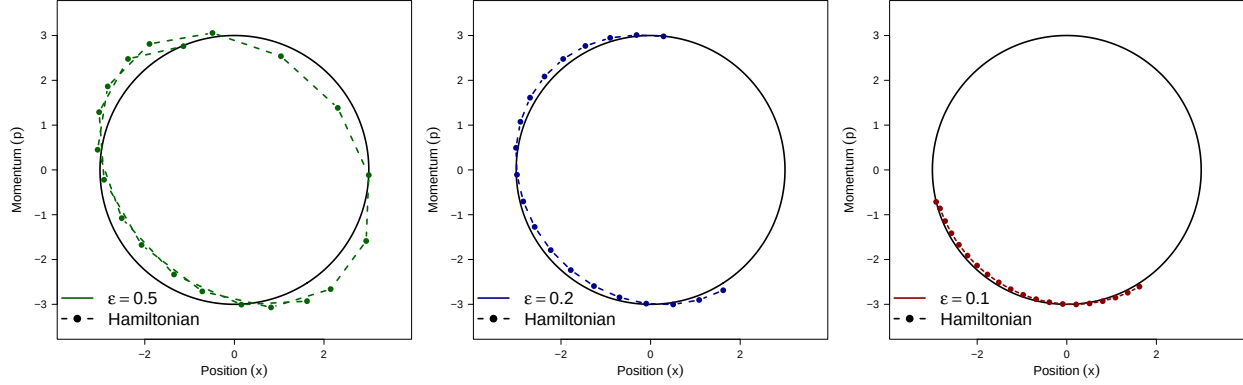


Figure 5: Approximated Hamiltonian paths with leapfrog discretization for different $\varepsilon = 0.5, 0.2, 0.1$ (from left to right) for standard Gaussian target distribution.

The function $P_{L,\varepsilon}$ in (16) is both an involution and volume-preserving due to the nice properties of the leapfrog integrator. However, the resulting $P_{L,\varepsilon}(x, p)$ does not conserve energy exactly, and this implies that the acceptance ratio need not be 1. As before, flipping the sign is not important in practice. The resulting implementable version of HMC using the leapfrog integrator is provided in Algorithm 4. This is the HMC algorithm, known and used widely in practice.

Algorithm 4 Hamiltonian Monte Carlo for 1-dimensional targets

- 1: **Initialize:** Set initial value $x^{(0)}$.
 - 2: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
 - 3: Given current state $x^{(t)}$.
 - 4: Draw momentum $p \sim N(0, m)$.
 - 5: Compute $(x^*, -p^*) = T_{L,\varepsilon}(x^{(t)}, p)$ using L leapfrog steps of size ε .
 - 6: Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \exp\{-H(x^*, p^*) + H(x^{(t)}, p)\}.$$
 - 7: Draw $W \sim \text{Uniform}(0, 1)$, independently.
 - 8: **if** $W \leq \min\{1, r(x^{(t)}, x^*)\}$ **then**
 - 9: Set $x^{(t+1)} = x^*$.
 - 10: **else**
 - 11: Set $x^{(t+1)} = x^{(t)}$.
 - 12: **end if**
 - 13: **end for**
 - 14: **Return:** $\{x^{(t)}\}_{t=0}^{T-1}$.
-

Running example 3.4. For $\pi \equiv N(0, 1)$ and $m = 1$,

$$\nabla K(p) = p \text{ and } \nabla U(x) = x.$$

We implement Algorithm 4 in the R code below.

```

1 normal_HMC <- function(L = 10, eps = .1, n = 1e4)
2 {
3   x <- numeric(length = n)
4   x[1] <- 0 # starting from x = 0
5   # vectorizing this outside to save time in R
6   momentums <- rnorm(n)
7
8   for(k in 2:n)
9   {
10    p <- momentums[k] # new momentum
11    x_prop <- x[k-1]
12
13    # one half-Euler step
14    p_prop <- p - eps/2 * x_prop
15    # let the frog leap!
16    for(l in 1:L)
17    {
18      x_prop <- x_prop + eps * p_prop
19      if(l != L) p_prop <- p_prop - eps*x_prop
20    }
21    # one last half-Euler step
22    p_prop <- p_prop - eps/2 * x_prop
23
24    # Accept-reject on log scale
25    log.ratio <- (-x_prop^2 - p_prop^2 + x[k-1]^2 + p^2)/2
26    if(log(runif(1)) < log.ratio)
27    {
28      x[k] <- x_prop
29    } else{
30      x[k] <- x[k-1]
31    }
32  }
33  return(x)
34 }

```

We call the function with $s = 1$ for various choices of ε and present the results in Figure 6. The quality of the samples is fairly close to that of the ones obtained from the ideal HMC.

3.3.2. HMC in higher dimensions

So far, we have inherently assumed that π is defined on a 1-dimensional space. Naturally, this is rarely the case for the problems in which MCMC is employed. The extension to a general d -dimensional space is fairly straightforward. The potential $U(x)$ remains $U(x) = -\log \pi(x)$. However, the auxiliary variable (momentum) is also d -dimensional and for an appropriately chosen $p \times p$ mass matrix, M , the kinetic energy considered is

$$K(p) = \frac{1}{2}p^\top M^{-1}p.$$

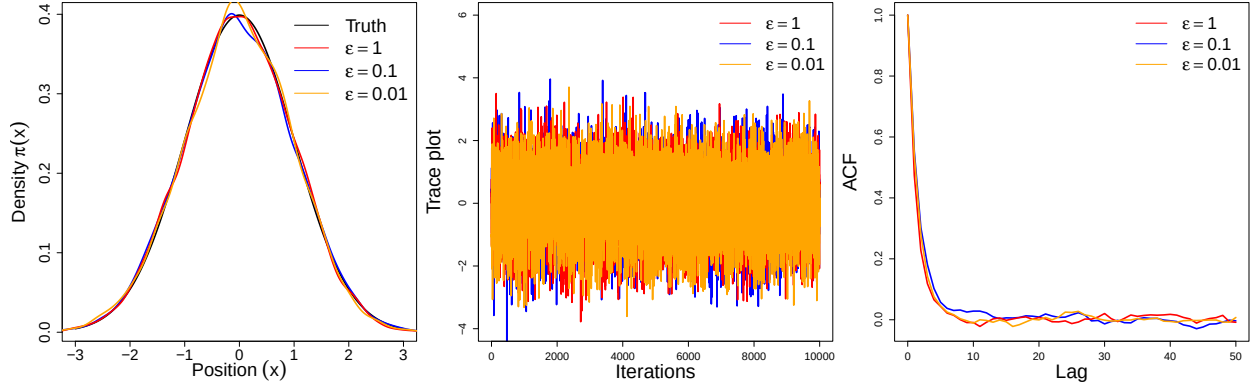


Figure 6: Density plots (left panel), trace plots (middle panel), and ACF plots (right panel) of HMC on Gaussian target density with different $\varepsilon = 0.01, 0.1, 1$ with $s = L\varepsilon = 1$.

Thus, the Hamiltonian on $\mathbb{R}^{2d}$ takes the form

$$H(x, p) = U(x) + K(p) = -\log \pi(x) + \frac{1}{2}p^\top M^{-1}p.$$

Consequently, $\nabla K(p) = M^{-1}p$. Given a step size $\varepsilon > 0$ and a number of steps L , the leapfrog in general dimensions can be written down as an extension of (12)–(14) as follows:

$$\begin{aligned} p_{t+\frac{\varepsilon}{2}} &= p_t - \frac{\varepsilon}{2}\nabla U(x_t), \\ x_{t+\varepsilon} &= x_t + \varepsilon M^{-1}p_{t+\varepsilon/2}, \\ p_{t+\frac{3\varepsilon}{2}} &= p_{t+\varepsilon/2} - \varepsilon\nabla U(x_{t+\varepsilon}), \\ &\vdots \\ x_{t+L\varepsilon} &= x_{t+(L-1)\varepsilon} + \varepsilon M^{-1}p_{t+(2L-1)\varepsilon/2}, \\ p_{t+L\varepsilon} &= p_{t+(2L-1)\varepsilon/2} - \frac{\varepsilon}{2}\nabla U(x_{t+L\varepsilon}). \end{aligned} \tag{17}$$

The final HMC algorithm in general dimensions is provided in Algorithm 5.

4. Choosing tuning parameters

A successful implementation of an HMC algorithm requires careful tuning of s, ε, L , and M . In this section, we provide recommendations for these parameters based on both heuristics and theory. Typically, a good strategy is to first choose M according to Section 4.1. Next, fix a small L and choose ε according to the guidelines in Section 4.2. Finally, using M and ε thus chosen, choose L and consequently $s = L\varepsilon$ based on the guidelines in Section 4.3.

Algorithm 5 Hamiltonian Monte Carlo for d -dimensional targets

```

1: Initialize: Set initial value  $x^{(0)}$ .
2: for  $t = 0, 1, 2, \dots, T - 1$  do
3:   Given current state  $x^{(t)}$ .
4:   Draw momentum  $p \sim N_d(0, M)$ .
5:   Compute  $(x^*, -p^*) = T_{L,\varepsilon}(x^{(t)}, p)$  using  $L$  leapfrog steps of size  $\varepsilon$  using (17).
6:   Compute the acceptance ratio:

$$r(x^{(t)}, x^*) = \exp\{-H(x^*, p^*) + H(x^{(t)}, p)\}.$$

7:   Draw  $W \sim \text{Uniform}(0, 1)$ , independently.
8:   if  $W \leq \min\{1, r(x^{(t)}, x^*)\}$  then
9:     Set  $x^{(t+1)} = x^*$ .
10:  else
11:    Set  $x^{(t+1)} = x^{(t)}$ .
12:  end if
13: end for
14: Return:  $\{x^{(t)}\}_{t=0}^{T-1}$ .
```

4.1. Mass matrix M

A first practical and natural choice for M is setting $M = \mathbb{I}_d$. Usually, a practitioner first tries this choice, and if the sampler is ineffective, a more informed choice of M is made. A practical workflow is demonstrated in Section 5. When a mass matrix different from $\mathbb{I}_d$ is employed, this is akin to linear preconditioning.

The choice of the mass matrix controls how the algorithm moves through the augmented parameter space. Intuitively, the mass matrix determines the relative speed at which the d different directions are explored during the simulated Hamiltonian trajectories. If these speeds are poorly matched to the scale of the target distribution, the resulting trajectories can become inefficient, requiring very small step sizes and leading to poor mixing. From a geometric perspective, a linear transformation of the position variables is equivalent to the inverse linear transformation of the momentum variables, and the mass matrix provides a convenient way to implement this re-scaling without explicitly reparametrizing the model. Considering this, an ideal choice of M is such that

$$M^{-1} = \text{Cov}_\pi(X) =: \Sigma.$$

However, Σ will typically be unavailable in practice, in which case, a warm-up chain is run from which Σ is estimated using a sample covariance matrix employing the sample mean, $\bar{x} = T^{-1} \sum_{t=0}^{T-1} x^{(t)}$:

$$\hat{\Sigma} = \frac{1}{T-1} \sum_{t=0}^{T-1} (x^{(t)} - \bar{x})(x^{(t)} - \bar{x})^\top. \quad (18)$$

Consequently, $M = \hat{\Sigma}^{-1}$ is chosen. There are two disadvantages of this method, (i) the estimator in (18) may be poor due to poor warm-up samples, and (ii) the sampler can be

expensive to implement due to the need of employing a matrix-vector multiplication for every leapfrog step.

A reasonable alternative is to employ a diagonal M with elements $1/\text{diag}(\hat{\Sigma})$. That is, the diagonals of M are set to be the marginal inverse sample variances for all components. This is a computationally cheaper alternative that can often work well, which we also employ in Section 5. However, recently, Hird and Livingstone (2025) show that it is in fact possible for this type of preconditioning to yield worse results than no preconditioning at all. They also propose a computationally viable alternate preconditioning method that proves to be more effective than diagonal preconditioning.

While an appropriate mass matrix can dramatically improve performance, it is important to recognize the limitations of linear preconditioning. Mass matrix adaptation is most effective when the target is well approximated by an elliptically contoured distribution, such as a multivariate Gaussian. For more complex targets exhibiting strong nonlinearity, varying curvature, or pronounced non-elliptical structure, no fixed linear transformation can simultaneously regularize the geometry everywhere. In such cases, Girolami and Calderhead (2011) propose Riemannian manifold HMC methods that incorporate position-dependent geometry where the mass matrix M is allowed to also depend on x . We expand on this a bit more in Section 6.3.

4.2. Discretization size ε

Having found an appropriate M , we move on to choosing ε . This parameter is more easily tunable due to the measurable effects and consequences of choosing a particular ε .

Recall that the acceptance probability for the ideal HMC in Algorithm 3 is exactly 1 due to the exact conservation of the Hamiltonian. In most situations, the solution to Hamilton's equations is unavailable, and the leapfrog integrator in Section 3.3.1 is used to simulate the Hamiltonian dynamics. This approximation leads to a change in the Hamiltonian, no longer ensuring an acceptance probability of exactly 1. Recall, however, that the Hamiltonian is increasingly better conserved as $\varepsilon \rightarrow 0$. Thus, the choice of the discretization step, ε , dictates the acceptance probability of the algorithm. A smaller value of ε will yield higher acceptance, and a larger value of ε will yield lower acceptance. Further, due to the characteristics of the leapfrog integrator, for a fixed ε , the acceptance probability is minimally affected by the choice of L .

Running example 4.1. For our running example, we implement the HMC algorithm for $s = L\varepsilon = 1$ and $s = L\varepsilon = 10$ in Table 1. We make two observations: first, the same value of ε for different values of L yields similar acceptances. This indicates that indeed the acceptance probability is primarily affected by ε . Second, acceptances decrease as ε increases. As always, large acceptance is not necessarily an indicator of a good Markov chain. In Figure 7, we present visual summaries of the HMC algorithm with $s = L\varepsilon = 10$. This figure is in stark contrast to Figure 6, where all choices of ε yielded well-behaved Markov chains. Here, when ε is too large, the chain gets stuck at the current position for a while since the Hamiltonian of the proposal deviates significantly from the Hamiltonian of the current value.

As discussed previously, a larger acceptance probability need not imply an excellent Markov chain. Using techniques from the optimal scaling literature (Roberts and Rosenthal,

Table 1: Acceptance rates for Leapfrog-based HMC algorithm with varying (L, ε) settings. Total trajectory length is $s = L\varepsilon$.

$s = L\varepsilon = 1$				$s = L\varepsilon = 10$			
Chain	ε	L	Acceptance	Chain	ε	L	Acceptance
1	0.01	100	1.0000	1	0.1	100	0.9995
2	0.1	10	0.9996	2	1	10	0.9155
3	1	1	0.9181	3	10	1	0.0037

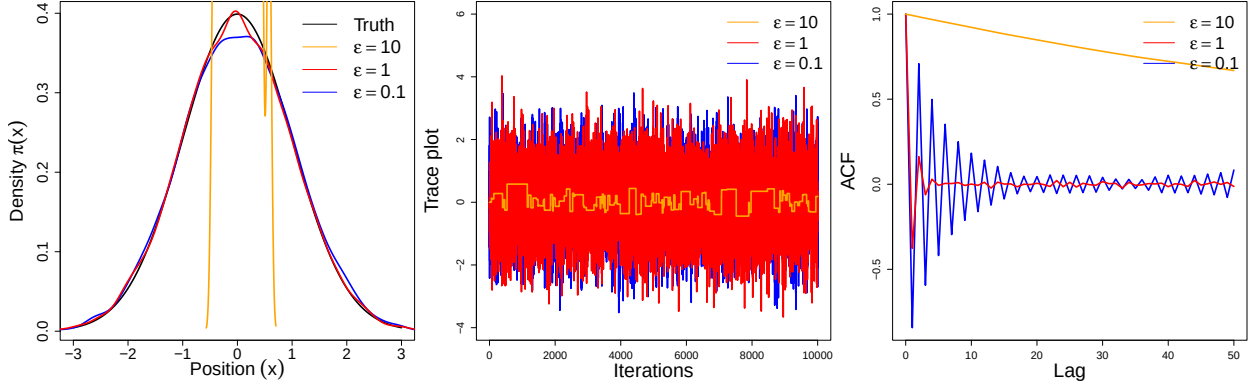


Figure 7: Density plots (left panel) and trace plots (right panel) for exact HMC on Gaussian target density with different $\varepsilon = 0.1, 1, 10$ with $s = L\varepsilon = 10$.

2001), Beskos *et al.* (2013) showed that as dimension $d \rightarrow \infty$, a user should tune ε to yield an acceptance rate of 0.651. However, for large but finite d , their simulations showed that higher acceptance rates up to 0.85 can yield better results. In general, users often choose ε aiming for 65% acceptance for complicated and high-dimensional target distributions and tune for acceptances closer to 85% for simpler targets. The tuning of ε can then be done using shorter length Markov chains than the final run, employing a starting value that is generally in an area of high probability.

4.3. Number of leapfrog steps and s

Having chosen M and ε , we move on to choosing the number of leapfrog steps, L . The performance of HMC depends critically on the number of leapfrog steps. Too few steps and the chain will explore the space like a slow random walk; too many, and we waste computation. A reasonable way then to choose L is to assess the computational budget. Each step of the leapfrog requires the calculation of a gradient vector, which is often the leading cost in the algorithm. Thus, L can be chosen based on the computation available. Note that for a given ε , a large L is unlikely to affect the acceptance probability of the HMC algorithm. Thus, if we have the computational bandwidth to increase L , it is often preferable.

There is, however, a limit to how large we might want L to be. It is possible for L to be large enough that a push of $s = L\varepsilon$ time units brings the object back close to the original position. This depends on the period of the Hamiltonian. Ideally, we would like to choose L so that $s = L\varepsilon$ is close to the half period of the underlying Hamiltonian. Then, the proposal

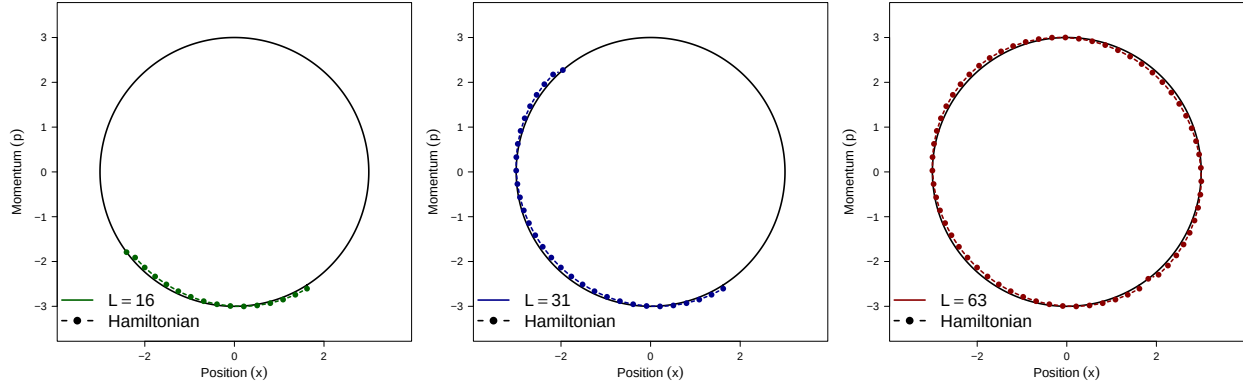


Figure 8: Approximated Hamiltonian paths with leapfrog discretization for fixed $\varepsilon = 0.1$ and with different L with time-trajectories $s = L\varepsilon$ at quarter period (left), half-period (middle), full-period (right) for standard Gaussian target distribution.

x^* is in general far from the current state of the object. This is challenging to do in practice since different parts of the state-space can have different periodicity. We discuss a popular adaptive procedure that attempts to solve this problem in Section 6.1.

Running example 4.2. In Figure 8, we demonstrate the quarter, half, and full period of the Gaussian Hamiltonian. For the same choice of ε (and thus same acceptance), each choice of L will yield different quality Markov chains. Increasing L beyond a value of 31 will be both wasteful and statistically inefficient.

In the next section, we discuss the implementation of HMC for a Bayesian logistic regression model on a real and popular dataset, demonstrating some key learnings from this section.

5. Example implementation

We present an HMC implementation for Bayesian logistic regression on the popular Pima Indian diabetes dataset (Chopin and Ridgway, 2017). This dataset was originally collected by the National Institute of Diabetes and Digestive and Kidney Diseases and contains information on adult female individuals of Pima Indian heritage, all aged at least 21 years and living near Phoenix, USA. Each response is a binary variable indicating the presence or absence of diabetes, together with covariates derived from medical examinations. We analyze a subset of the data available in the **MASS** package in **R**, consisting of $n = 200$ observations with no missing values. There are $d = 8$ covariates, including the intercept, number of pregnancies (**pregnant**), plasma glucose concentration measured two hours after an oral glucose tolerance test (**glucose**), diastolic blood pressure in mmHg (**pressure**), triceps skin-fold thickness in mm (**triceps**), body mass index (**mass**), defined as weight in kilograms divided by the square of height in meters, and the diabetes pedigree function (**pedigree**), which summarizes the genetic predisposition based on family history.

To study the relationship between these covariates and diabetes status, we employ a Bayesian logistic regression model of the form

$$Y_i \stackrel{\text{ind}}{\sim} \text{Bernoulli}(p_i) \text{ for all } i = 1, 2, \dots, n.$$

Denoting z_i as the covariate for the i th observation and the corresponding regression coefficient vector β , we model each p_i with the logit linked linear covariates as

$$\log\left(\frac{p_i}{1-p_i}\right) = z_i^\top \beta,$$

with independent priors on the regression coefficient $\beta \sim N_d(0, \sigma^2 \mathbb{I}_d)$. We choose the hyperparameter $\sigma^2 = 100$ to impose weak prior information. The resulting posterior is proper but intractable, and thus MCMC is used to obtain samples for posterior summaries.

First, we illustrate the importance of employing a suitable mass matrix as described in Section 4.1. We begin by running a baseline HMC algorithm using the identity mass matrix $M = \mathbb{I}_d$, following Algorithm 5. The sampler is run for 10^5 iterations with tuning parameters $L = 20$ and $\varepsilon = 2.1 \times 10^{-5}$, chosen in accordance with the guidelines in Section 4. The resulting trace plots shown in Figure 9 reveal that while most components of β exhibit satisfactory mixing, two components `Intercept` and `pedigree` mix slowly due to the variability of different scales present in the posterior. This lack of mixing leads to unreliable posterior density estimates, as displayed in Figure 10.

To mitigate the scale imbalance, we employ the diagonal preconditioning strategy described in Section 4.1. Specifically, we use a computationally cheaper diagonal mass matrix M , comprised of the inverse marginal sample variances estimated from warm-up iterations of a naive HMC run. We then rerun the HMC sampler for 10^5 iterations using the same leapfrog length $L = 20$, but with a substantially larger step size $\varepsilon = 0.11$, which maintains an acceptance rate consistent with the tuning considerations of Section 4. Figure 11 displays trace plots for the final 10^4 iterations of this preconditioned sampler, and the corresponding posterior density estimates are shown in Figure 12. Together, these results demonstrate a significant improvement in mixing across all components of β , as well as substantially more reliable posterior density estimation. The R codes to reproduce the simulation studies and real data analysis are available on GitHub⁴.

6. HMC variants

The potential power of HMC algorithms, as well as the need for its careful tuning, has led to several extensions and variations. We provide a short review of popular variants of the HMC algorithm proposed in the literature.

6.1. NUTS

Perhaps the most famous and powerful extension of the HMC algorithm is the No-U-Turn Sampler (NUTS). As seen in our previous sections, the efficiency of HMC critically depends on the tuning of the step size ε and the number of leapfrog steps L . If ε is too large relative to the local curvature of π , the numerical trajectory can diverge, while an excessively small ε leads to insufficient exploration. The NUTS (Hoffman and Gelman, 2014) eliminates the need to pre-specify the integration time $s = L\varepsilon$ by adaptively determining how long to simulate Hamiltonian dynamics. NUTS recursively constructs a trajectory by simulating Hamiltonian dynamics in both forward and backward time directions uniformly at random.

⁴Available at the repository https://github.com/ArghyaStat/HMC_for_dummies.

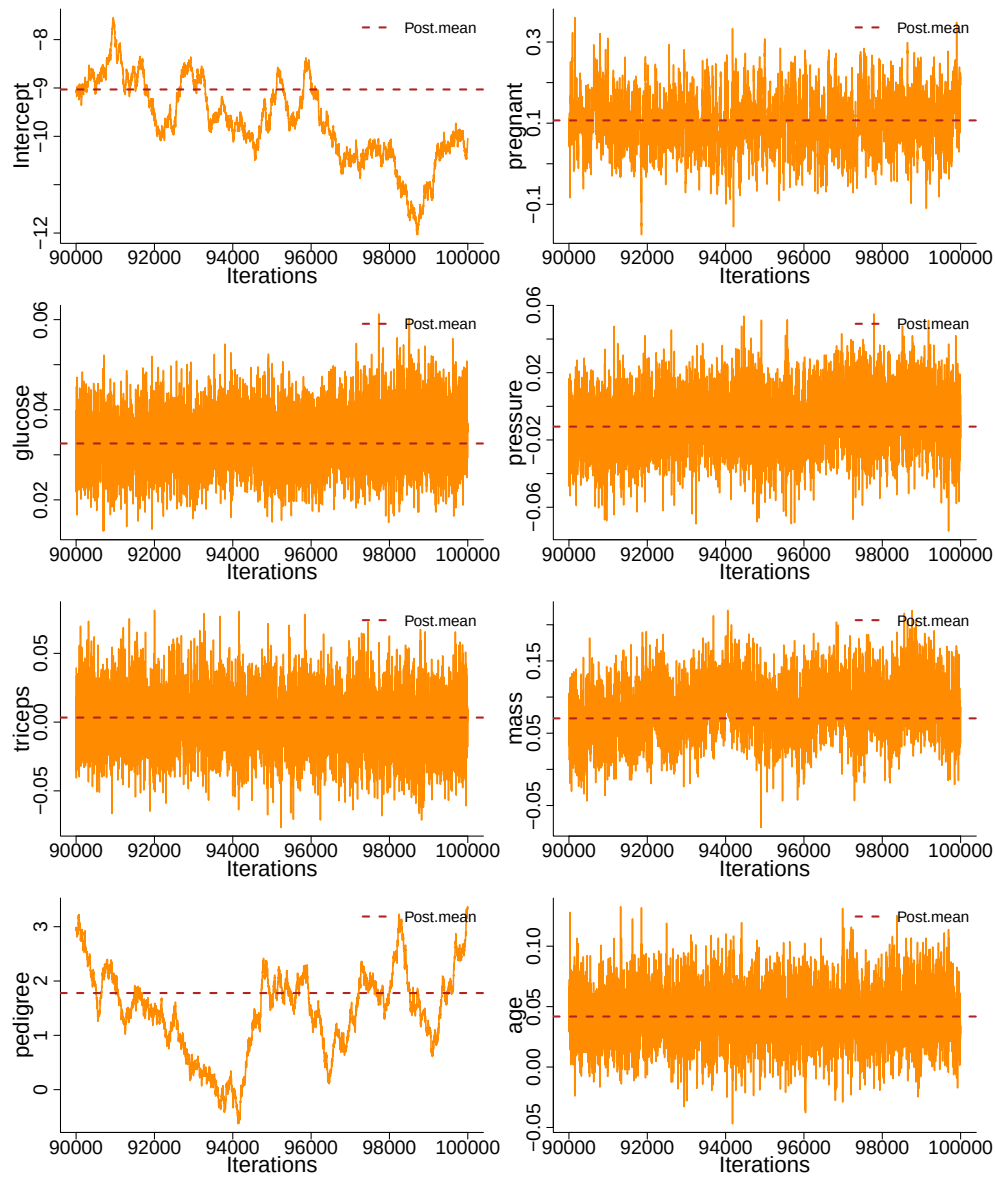


Figure 9: Traceplots of components of β with posterior mean (in dotted line) of the resulting Markov chain from the naive HMC sampler. The figure indicates that, except for Intercept and pedigree, all components are fast mixing.

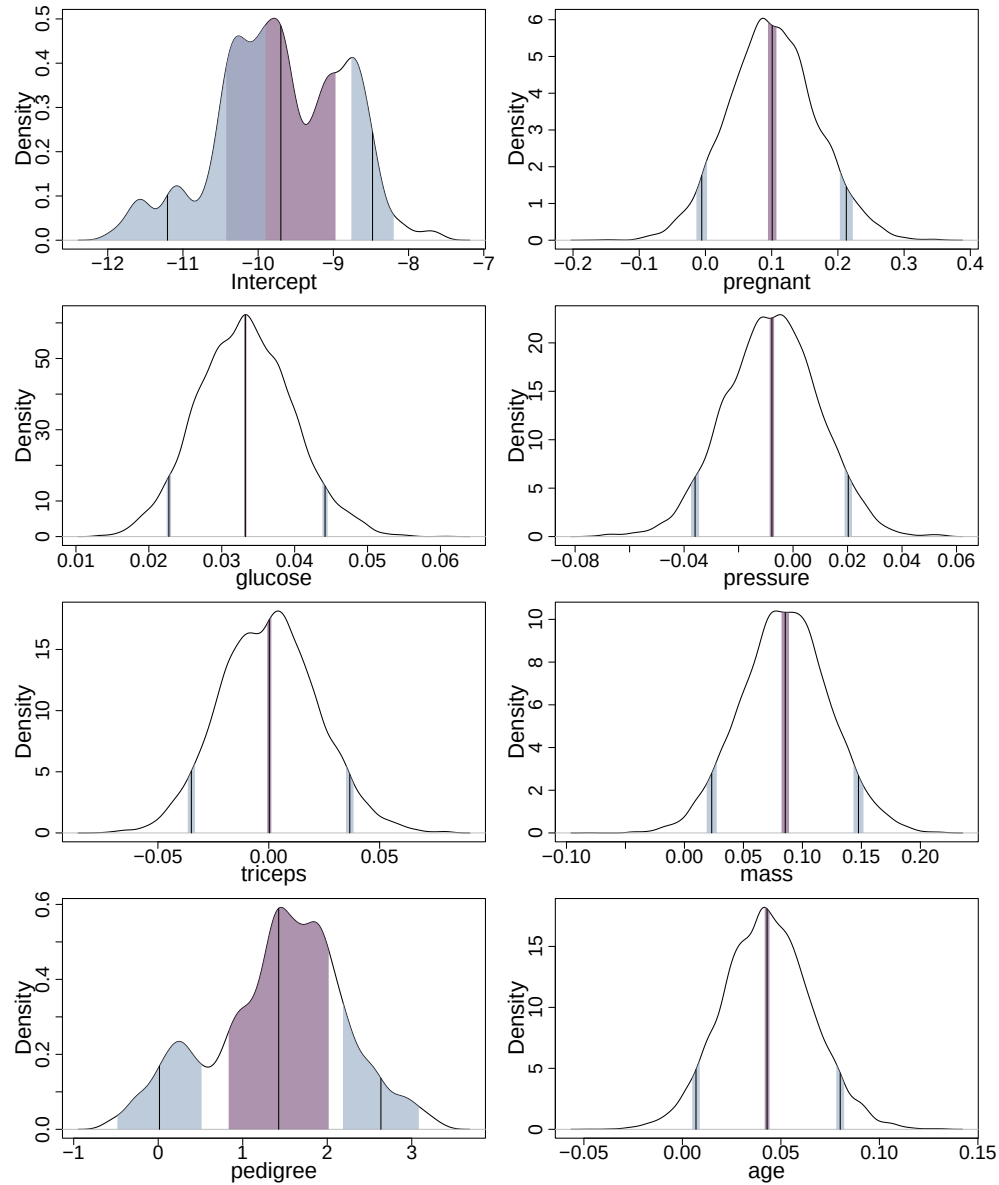


Figure 10: Density estimates for the components of β , with posterior means and the 5% and 95% estimated posterior quantiles from the naive HMC sampler. Bands around vertical lines show Monte Carlo uncertainty. Plot is made using R package SimTools.

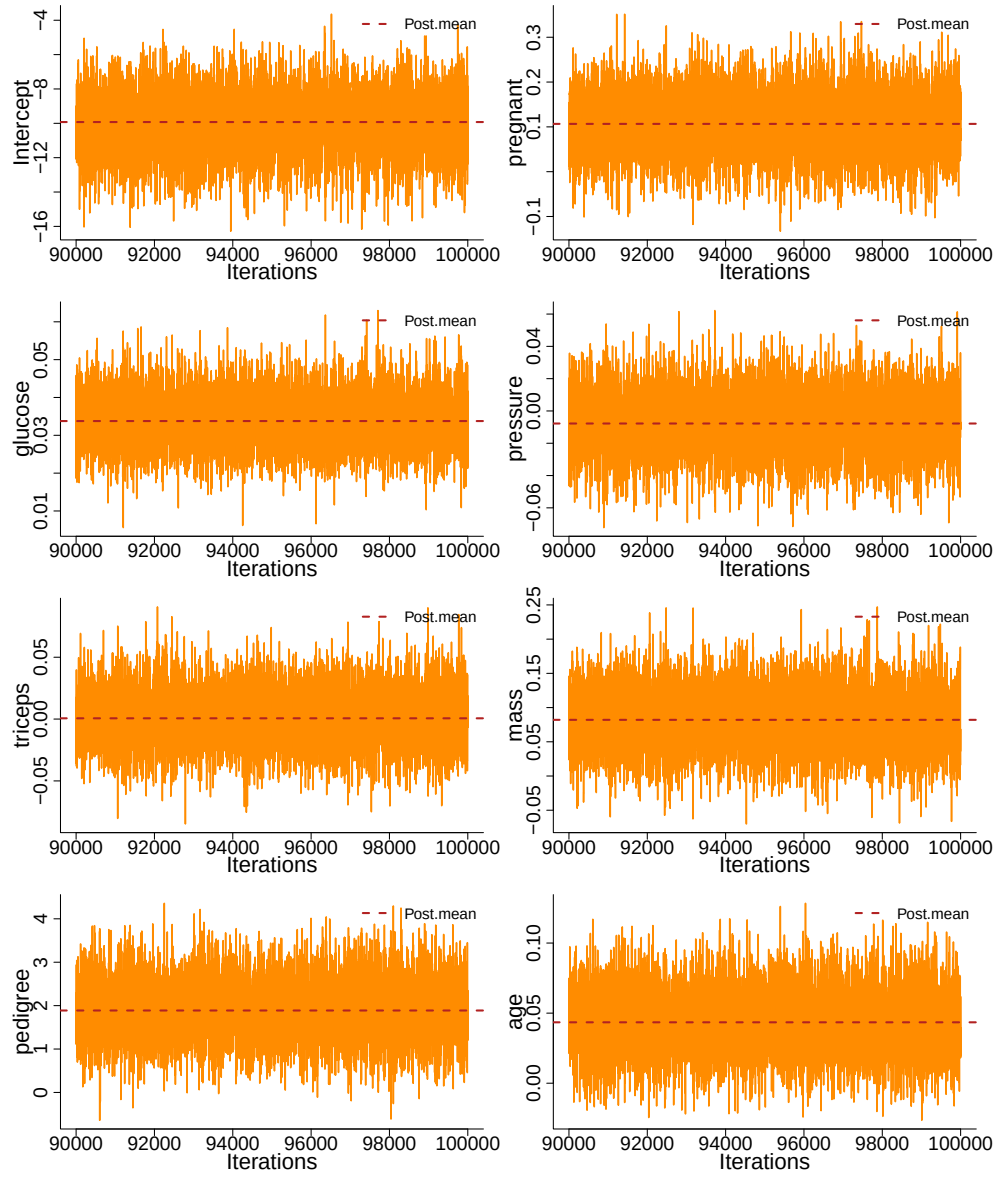


Figure 11: Trace plots of components of β with posterior mean (in dotted line).

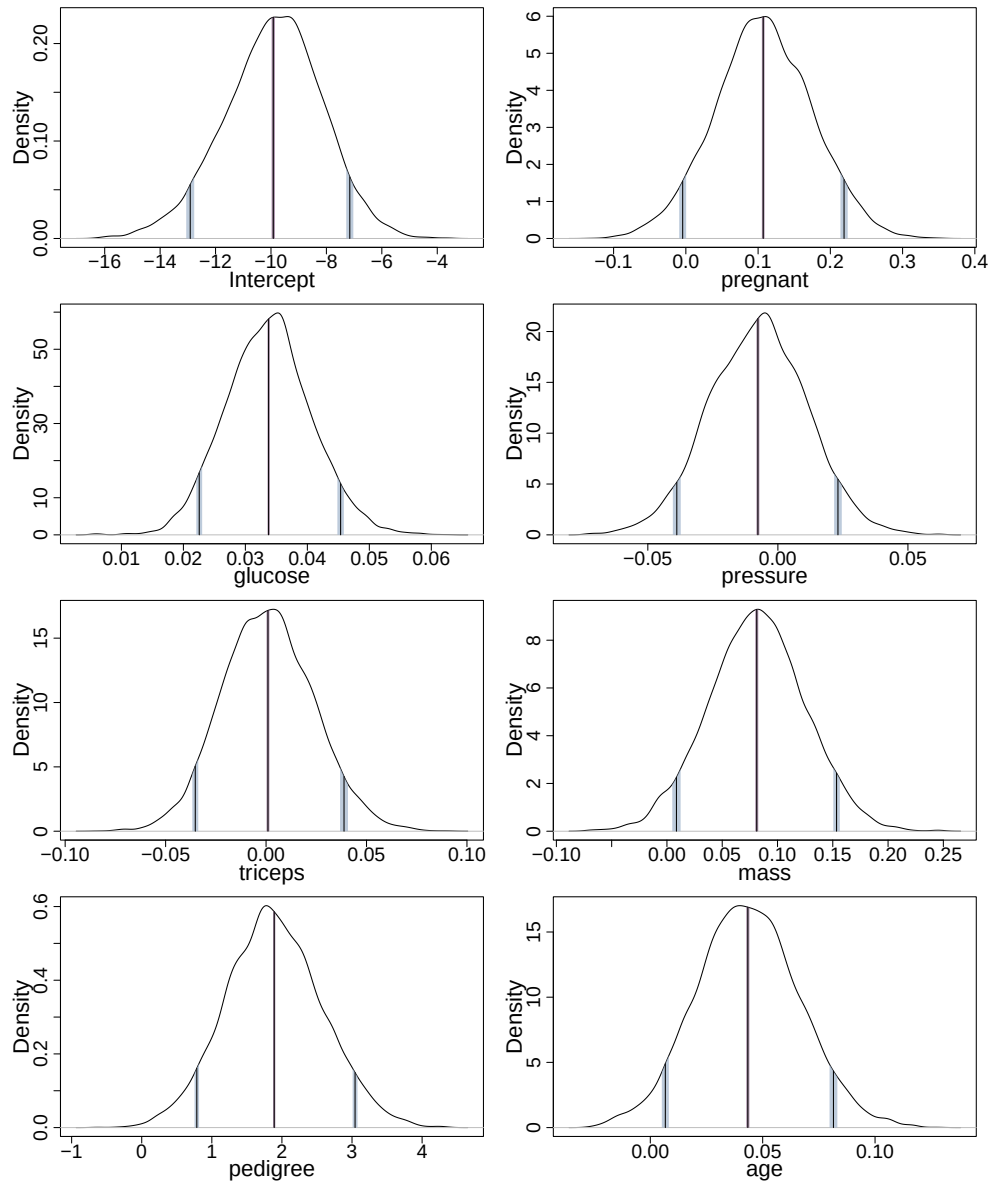


Figure 12: Density estimates for the components of β , with posterior means and the 5% and 95% estimated posterior quantiles for pre-conditioned sampler. The figure indicates that all components are well estimated.

The orbit is thus extended in that direction by doubling its length. The specifics of this ensure that the trajectory does not double back on itself. The resulting set of candidate states forms an orbit that is both symmetric and volume-preserving, thereby maintaining the reversibility of the Markov transition. From the constructed orbit, a single proposal (x^*, p^*) is drawn using a sampling scheme that preserves invariance. This adaptive construction enables NUTS to automatically adjust the integration time according to the local geometric features of the target distribution, substantially improving sampling efficiency without requiring manual tuning of L . Stan (Carpenter *et al.*, 2017) provides a widely used implementation of NUTS.

While NUTS adaptively addresses the integration length, it still employs a single global step size ε across the entire sampling process. In Stan, this choice of ε is determined in the warm-up phase using dual averaging (Nesterov, 2009) targeting desired acceptance rate. In multiscale problems, regions of high curvature in $U(x)$ may demand smaller step sizes for stability, whereas flatter regions permit larger ones. A fixed ε , therefore, presents a trade-off between numerical stability and efficiency. Bou-Rabee *et al.* (2025b) recently introduce orbit-level adaptive step-size variants of NUTS, which select ε separately for each orbit while maintaining detailed balance. Building further on this idea, Bou-Rabee *et al.* (2025a) propose the WALNUTS (Within-orbit Adaptive Leapfrog No-U-Turn Sampler), where the step size ε_t is allowed to vary within each orbit as a function of the local curvature, thereby ensuring stability in regions of high curvature while preserving efficiency in flatter regions. This localized adaptation improves robustness and energy control without violating reversibility or detailed balance.

6.2. Stochastic gradient HMC

One of the crucial bottlenecks of HMC is the need to compute the gradient of the log target repeatedly within a single iteration. Modern models frequently deal with datasets having a massive number of observations. In these ever-more-common scenarios of massive data, full gradient computation is infeasible. For example, suppose we want to sample from the posterior distribution of θ , the model parameter given a set of independent observations $\mathcal{D} = \{y_1, \dots, y_n\}$, denoted by

$$\pi(\theta|\mathcal{D}) = \left\{ \prod_{j=1}^n \log p(y_j|\theta) \right\} p(\theta) \propto \exp(-U(\theta)), \quad (19)$$

where $\log p(y_j|\theta)$ is the likelihood of θ based on y_j and $p(\theta)$ is the prior density of θ . The gradient of the potential energy function is thus given by

$$\nabla U(\theta) = - \sum_{j=1}^n \nabla \log p(y_j|\theta) - \nabla p(\theta). \quad (20)$$

Traditional stochastic methods (Welling and Teh, 2011) enjoy success by approximating $\nabla U(\theta)$ by $\nabla \tilde{U}(\theta)$ using minibatches of data, say $\tilde{\mathcal{D}}$, of size $k \ll n$ to scale algorithms for large n as

$$\nabla \tilde{U}(\theta) = -\frac{k}{n} \sum_{y_j: y_j \in \tilde{\mathcal{D}}} \nabla \log p(y_j|\theta) - \nabla p(\theta). \quad (21)$$

Chen *et al.* (2014) propose variants of stochastic gradient HMC with replacing ∇U with $\nabla \tilde{U}$. But HMC with stochastic gradient $\nabla \tilde{U}$ can remain costly due to the accept-reject step requiring the need to evaluate the full Hamiltonian. Chen *et al.* (2014) introduces a friction term in the Hamiltonian dynamics which allows the proposed sample path to be used without an accept-reject step. Note that the use of stochastic gradients comes at the cost of exactness of the algorithm.

Ma *et al.* (2017) provide a complete framework for stochastic gradient MCMC methods. They discussed a unified underlying stochastic dynamical system where HMC and SGHMC are special cases. Zou and Gu (2021) propose a stochastic variance-reduced HMC method for sampling from a smooth and strongly log-concave distribution. Their algorithm exhibits a faster rate of convergence and better gradient complexity than vanilla HMC and stochastic gradient HMC methods across a wide range of regimes. For a complete review of stochastic gradient MCMC methods, see Nemeth and Fearnhead (2021).

6.3. Riemannian manifold HMC

Sampling efficiency of HMC highly depends on the local geometric structure of the target distribution. Several extensions of HMC have sought to exploit this structure by modifying the kinetic energy function $K(p)$ through the introduction of global preconditioning matrices. A major advancement in this direction was proposed by Girolami and Calderhead (2011), who extended HMC from Euclidean manifolds to Riemannian manifolds by defining a position-dependent Gaussian momentum distribution. This is referred to as Riemannian manifold HMC (RHMC). In RHMC, the kinetic energy incorporates a second-order position-dependent metric tensor, typically the local Hessian of the potential function. Consequently, the Hamiltonian becomes non-separable in position (x) and momentum (p), rendering the generalized leapfrog updates computationally more expensive and often unstable compared to the true Hamiltonian trajectory.

Numerical instability arises because regions of high curvature in the parameter space can induce excessively large momentum realizations, leading to unstable trajectories unless a sufficiently small step size is employed. This restriction often necessitates conservative integration settings that limit efficiency across the parameter space. Inspired by the wide use of gradient clipping in deep learning, Lu *et al.* (2017) proposed a modified kinetic energy formulation that effectively bounds particle velocities during Hamiltonian evolution, mitigating the risk of instability. More recently, Whalley *et al.* (2024) investigated the use of randomized integration times (Bou-Rabee and Sanz-Serna, 2017) to enhance the numerical stability and mixing performance of RHMC. We discuss various numerical integrators and their stability properties in Section 6.6.

6.4. HMC for non-differentiable targets

Differentiability of the target density is a fundamental requirement for HMC. Nevertheless, Bayesian models formulated to induce structured sparsity frequently employ non-differentiable priors, thereby presenting substantial challenges in designing efficient sampling algorithms. Chaari *et al.* (2016) were among the first to address this issue by introducing a framework for sampling from probability distributions with non-differentiable energy functions, referred to as non-smooth HMC (ns-HMC). Their approach approximates the non-

differentiable potential using a smooth envelope. Although ns-HMC provides a principled solution to handle non-smooth potentials, its computational efficiency is limited when the proximal mapping of the potential is not available in closed form and must be evaluated through iterative optimization procedures.

More recently, Shukla *et al.* (2025) critically examined the limitations of ns-HMC and proposed the proximal HMC (p-HMC) algorithm, which integrates proximal mapping techniques within the Hamiltonian framework. They established theoretical guarantees of geometric ergodicity and offered practical guidelines for selecting tuning parameters associated with the proximal operator. Both ns-HMC and p-HMC essentially rely on smooth approximations of the non-smooth potential function via Moreau-Yosida envelopes for the purpose of gradient calculation. It remains to be seen whether there exist enveloping techniques that are particularly suited for Hamiltonian dynamics.

6.5. HMC for multi-modal distributions

Almost all standard MCMC methods struggle with crossing low-probability barriers that separate modes, and consequently take a long time to search for new modes or traverse from one mode to another, even in low dimensions. Łatuszyński *et al.* (2026) mention three major challenges of multi-modal sampling, namely: a) moving between the modes, b) finding the modes, and c) sampling efficiently within the modes. Gradient-based schemes, like HMC, particularly struggle with moving between modes since local gradients tend to push the sampler towards the current local basin of attraction.

The inability of HMC to sample effectively from multi-modal distributions has been widely acknowledged in the literature (Celeux *et al.*, 2000; Sminchisescu and Welling, 2007). For example, Mangoubi *et al.* (2018) show that for multi-modal target distributions, the ability of HMC to transition between modes is provably worse than that of random-walk Metropolis.

Tripuraneni *et al.* (2017) propose a non-canonical Hamiltonian dynamics called Magnetic HMC (MHMC), where the mechanics of the particle is coupled to a magnetic field. Magnetic dynamics accelerate the particle’s transition from one mode to another. Mongwe *et al.* (2021) proposed perturbed Hamiltonians designed with importance sampling. A more recent work in this context is Repelling-Attracting Hamiltonian Monte Carlo (RAHMC) by Vishwanath and Tak (2024). In RAHMC, the modified dynamics involve a friction parameter that allows (i) mode-repelling, that encourages the sampler to move away from regions of high probability density, and (ii) mode-attracting, that facilitates the sampler’s ability to find and settle near alternative modes. Moreover, Vishwanath and Tak (2024) show that the RAHMC preserves various important properties of traditional Hamiltonian dynamics like volume-preservation, time-reversibility, *etc.*

6.6. Numerical integrators for HMC

The main challenge in implementing the HMC method is generating the Hamiltonian trajectories themselves. Aside from a few trivial examples, we cannot solve Hamilton’s equations exactly, and any implementation must instead solve them numerically. Numerical inaccuracies, however, can quickly compromise the utility of even the most well-tuned

Hamiltonian transition. An active area of research is on using alternative integrators other than leapfrog integrators (Leimkuhler and Reich, 2004; Blanes *et al.*, 2014) in HMC. For highly curved posteriors, stability requires a small step size ε , which may lead to increased computational cost. To improve accuracy without reducing ε , higher-order symplectic methods (Yoshida, 1990) have been proposed, which are based on a symmetric composition of multiple leapfrog steps, achieving fourth-order accuracy. Brofos and Lederman (2021) study the implicit midpoint integrator in the context of RMHMC. They demonstrate that the implicit midpoint integrator exhibits superior conservation of energy, better preservation of phase-space volume and time-reversibility, compared to the standard generalized leapfrog in non-separable Hamiltonians.

Meanwhile, Pourzanjani and Petzold (2019) examines HMC for multiscale distributions where the standard leapfrog step size must be extremely small. They demonstrate that an implicit integrator enables larger step sizes and improved sampling efficiency in the presence of severe geometric non-linearity in the target. On the other hand, each step requires solving fixed-point or Newton iterations, which increases the per-step cost and complexity in implementation and tuning.

7. Discussion

The introduction of HMC has brought somewhat of a resurgence to Bayesian computation. Although not without its flaws, HMC has still been successfully employed in a wide variety of applications. Concurrently with methodological developments, there have been considerable efforts in also studying theoretical properties of HMC samplers. Durmus *et al.* (2017) were one of the first to study the ergodicity of the HMC algorithm. Livingstone *et al.* (2019) establish sufficient conditions for geometric ergodicity of HMC algorithms, with conditions similar to those required for MALA. Chen *et al.* (2020) study non-asymptotic bounds on the mixing time of HMC, quantifying the improvement compared to MALA.

More recently, together with stochastic and approximate sampling schemes, HMC has found great success in computation for machine learning. The use of HMC algorithms within the machine learning community has become prevalent, particularly for sampling from high-dimensional and potentially complicated distributions. We provide a brief review of the existing machine learning literature, where HMC plays a crucial role in sampling-based inference. Havasi *et al.* (2018) applies Stochastic Gradient HMC (SGHMC) for inference in deep Gaussian processes, demonstrating superior posterior estimation accuracy compared to the state-of-the-art doubly stochastic variational inference framework of Salimbeni and Deisenroth (2017). Their results illustrate the potential of Hamiltonian dynamics to provide uncertainty quantification beyond variational approximations.

Extending this direction, Foreman *et al.* (2021) propose the deep-learning HMC algorithm, a generalized formulation of HMC wherein a stack of neural network layers replaces traditional leapfrog updates. This approach enables the integration of deep learning architectures into the HMC framework, allowing adaptive and learnable transition dynamics. In the context of adversarial learning, Wang *et al.* (2022) introduce HMC with accumulated momentum (HMC-AM), a method that adaptively controls step sizes across the trajectory. By allowing particles to evolve with variable step sizes along different directions, HMC-AM enhances flexibility in navigating complex loss surfaces common in adversarial training.

Robnik *et al.* (2023) propose microcanonical HMC (MCHMC), which follows fixed energy Hamiltonian dynamics rather than the canonical ensemble used in standard HMC. Most recently, Lockwood *et al.* (2024) extend the classical HMC framework into the quantum domain by proposing quantum dynamical HMC (QDHMC). This hybrid approach leverages quantum computation as a proposal mechanism, replacing the classical symplectic integration step with simulations of quantum-coherent continuous-space dynamics on digital or analog quantum hardware. The QDHMC framework exemplifies an emerging intersection between quantum simulation and Bayesian computation, suggesting promising directions for scalable probabilistic inference in the quantum era.

Acknowledgments

The authors are thankful to Andrew Holbrook for his wonder talk available at <https://www.youtube.com/watch?v=Byr9JVBI9cUv> and some useful discussions that helped highlight the relationship between HMC and the MHGJ Algorithm. The authors are also thankful to Sameer Deshpande and Amy Herring for their encouragement to write this article. Finally, DV thanks Arushi Agarwal and Sarthak Parikh for creating a safe space to ask all the questions about Hamiltonian dynamics a “physics dummy” would ask.

References

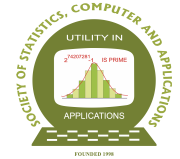
- Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, **19**, 1501–1534.
- Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.
- Blanes, S., Casas, F., and Sanz-Serna, J. M. (2014). Numerical integrators for the hybrid Monte Carlo method. *SIAM Journal on Scientific Computing*, **36**, A1556–A1580.
- Bou-Rabee, N., Carpenter, B., Kleppe, T. S., and Liu, S. (2025a). The within-orbit Adaptive Leapfrog No-U-Turn sampler. *arXiv preprint arXiv:2506.18746*.
- Bou-Rabee, N., Carpenter, B., Kleppe, T. S., and Marsden, M. (2025b). Incorporating local step-size adaptivity into the No-U-Turn Sampler using Gibbs self-tuning. *The Journal of Chemical Physics*, **163**, 084119.
- Bou-Rabee, N. and Sanz-Serna, J. M. (2017). Randomized Hamiltonian Monte Carlo. *The Annals of Applied Probability*, **27**, 2159–2194.
- Brofos, J. and Lederman, R. R. (2021). Evaluating the implicit midpoint integrator for Riemannian Hamiltonian Monte Carlo. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1072–1081. PMLR.
- Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, **76**, 1–32.
- Celeux, G., Hurn, M., and Robert, C. P. (2000). Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, **95**, 957–970.

- Chaari, L., Tournieret, J.-Y., Chaux, C., and Batatia, H. (2016). A Hamiltonian Monte Carlo method for non-smooth energy sampling. *IEEE Transactions on Signal Processing*, **64**, 5585–5594.
- Chen, S. (2022). Arianna Rosenbluth and the Metropolis Monte Carlo algorithm. *APS News*. <https://www.aps.org/apsnews/2022/03/rosenbluth-metropolis-monte-carlo-algorithm>.
- Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691. PMLR.
- Chen, Y., Dwivedi, R., Wainwright, M. J., and Yu, B. (2020). Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients. *Journal of Machine Learning Research*, **21**, 1–72.
- Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: Binary regression as a benchmark for Bayesian computation. *Statistical Science*, **32**, 64–87.
- Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, **195**, 216–222.
- Durmus, A., Moulines, E., and Saksman, E. (2017). On the convergence of Hamiltonian Monte Carlo. *arXiv preprint arXiv:1705.00166*.
- Foreman, S., Jin, X.-Y., and Osborn, J. C. (2021). Deep learning Hamiltonian Monte Carlo. *arXiv preprint arXiv:2105.03418*.
- Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**, 721–741.
- Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, **73**, 123–214.
- Glatt-Holtz, N. E., Holbrook, A. J., Krometis, J. A., Mondaini, C. F., and Sheth, A. (2026). Sacred and profane: from the involutive theory of MCMC to helpful Hamiltonian hacks. In Craiu, R. V., Vats, D., Jones, G. L., Brooks, S., Gelman, A., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, Boca Raton, 2nd edition.
- Granados, A. and Bañales, I. (2025). Understanding the Hamiltonian Monte Carlo through its physics fundamentals and examples. *arXiv preprint arXiv:2501.13932*.
- Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711–732.
- Hastings, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**, 97–109.
- Havasi, M., Hernández-Lobato, J. M., and Murillo-Fuentes, J. J. (2018). Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo. *Advances in Neural Information Processing Systems*, **31**.
- Hird, M. and Livingstone, S. (2025). Quantifying the effectiveness of linear preconditioning in Markov chain Monte Carlo. *Journal of Machine Learning Research*, **26**, 1–51.
- Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn Sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, **15**, 1593–1623.

- Jordan, T. F. (2004). Steppingstones in Hamiltonian dynamics. *American Journal of Physics*, **72**, 1095–1099.
- Latuszyński, K., Moores, M. T., and Stumpf-Fétizon, T. (2026). MCMC for multi-modal distributions. In Craiu, R. V., Vats, D., Jones, G. L., Brooks, S., Gelman, A., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, Boca Raton, 2nd edition.
- Leimkuhler, B. and Reich, S. (2004). *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.
- Liu, T., Surjanovic, N., Biron-Lattes, M., Bouchard-Côté, A., and Campbell, T. (2025). Autostep: Locally adaptive involutive MCMC. In *Forty-second International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 39624–39650. PMLR.
- Livingstone, S., Betancourt, M., Byrne, S., and Girolami, M. (2019). On the geometric ergodicity of Hamiltonian Monte Carlo. *Bernoulli*, **25**, 3109–3138.
- Lockwood, O., Weiss, P., Aronshtein, F., and Verdon, G. (2024). Quantum dynamical Hamiltonian Monte Carlo. *Physical Review Research*, **6**, 033142.
- Lu, X., Perrone, V., Hasenclever, L., Teh, Y. W., and Vollmer, S. (2017). Relativistic Monte Carlo. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1236–1245. PMLR.
- Ma, Y.-A., Foti, N. J., and Fox, E. B. (2017). Stochastic gradient MCMC methods for hidden Markov models. In *International Conference on Machine Learning*, pages 2265–2274. PMLR.
- Mak, C., Zaiser, F., and Ong, L. (2022). Nonparametric involutive Markov chain Monte Carlo. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14802–14859. PMLR.
- Mangoubi, O., Pillai, N. S., and Smith, A. (2018). Does Hamiltonian Monte Carlo mix faster than a random walk on multimodal densities? *arXiv preprint arXiv:1808.03230*.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, **21**, 1087–1092.
- Mongwe, W. T., Mbuva, R., and Marwala, T. (2021). Antithetic magnetic and shadow Hamiltonian Monte Carlo. *IEEE Access*, **9**, 49857–49867.
- Neal, R. M. (1996). Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, New York.
- Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC, Boca Raton, FL.
- Neklyudov, K., Welling, M., Egorov, E., and Vetrov, D. (2020). Involutive MCMC: a unifying framework. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7273–7282. PMLR.
- Nemeth, C. and Fearnhead, P. (2021). Stochastic gradient Markov chain Monte Carlo. *Journal of the American Statistical Association*, **116**, 433–450.
- Nesterov, Y. (2009). Primal-dual subgradient methods for convex problems. *Mathematical programming*, **120**, 221–259.

- Pourzanjani, A. A. and Petzold, L. R. (2019). Implicit Hamiltonian Monte Carlo for sampling multiscale distributions. *arXiv preprint arXiv:1911.05754*.
- Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, **16**, 351–367.
- Roberts, G. O. and Sahu, S. K. (1997). Updating schemes, correlation structure, blocking and parametrization for the Gibbs sampler. *Journal of the Royal Statistical Society, Series B*, **59**, 291–317.
- Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, **2**, 341–363.
- Robnik, J., De Luca, G. B., Silverstein, E., and Seljak, U. (2023). Microcanonical Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, **24**, 1–34.
- Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems*, **30**.
- Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, **2**, e55.
- Shukla, A., Vats, D., and Chi, E. C. (2025). Proximal Hamiltonian Monte Carlo. *arXiv preprint arXiv:2510.22252*.
- Sminchisescu, C. and Welling, M. (2007). Generalized darting Monte Carlo. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 516–523. PMLR.
- Spanbauer, S., Freer, C., and Mansinghka, V. (2020). Deep involutive generative models for neural MCMC. *arXiv preprint arXiv:2006.15167*.
- Thomas, S. and Tu, W. (2021). Learning Hamiltonian Monte carlo in R. *The American Statistician*, **75**, 403–413.
- Tierney, L. (1998). A note on Metropolis-Hastings kernels for general state spaces. *The Annals of Applied Probability*, **8**, 1–9.
- Tripuraneni, N., Rowland, M., Ghahramani, Z., and Turner, R. (2017). Magnetic Hamiltonian Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3453–3461. PMLR.
- Vishwanath, S. and Tak, H. (2024). Repelling-attracting Hamiltonian Monte Carlo. *arXiv preprint arXiv:2403.04607*.
- Wang, H., Li, G., Liu, X., and Lin, L. (2022). A Hamiltonian Monte Carlo method for probabilistic adversarial attack and learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**, 1725–1737.
- Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.
- Whalley, P. A., Paulin, D., and Leimkuhler, B. (2024). Randomized time Riemannian manifold Hamiltonian Monte Carlo. *Statistics and Computing*, **34**, 48.
- Yoshida, H. (1990). Construction of higher order symplectic integrators. *Physics letters A*, **150**, 262–268.

Zou, D. and Gu, Q. (2021). On the convergence of Hamiltonian Monte Carlo with stochastic gradients. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 13012–13022. PMLR.



On the Choice of Model Space Priors and Multiplicity Control in Bayesian Variable Selection: An Application to Streaming Logistic Regression

Joyee Ghosh

*Department of Statistics and Actuarial Science
The University of Iowa*

Received: 27 December 2025; Revised: 03 July 2026; Accepted: 08 July 2026

Abstract

Bayesian variable selection (BVS) depends critically on the specification of a prior distribution over the model space, particularly for controlling sparsity and multiplicity. This paper examines the practical consequences of different model space priors for BVS in logistic regression, with an emphasis on streaming data settings. We review some popular and well-known Beta–Binomial priors alongside the recently proposed matryoshka doll (MD) prior. We introduce a simple approximation to the MD prior that yields independent inclusion indicators and is convenient for scalable inference. Using BIC-based approximations to marginal likelihoods, we compare the effect of different model space priors on posterior inclusion probabilities and coefficient estimation at intermediate and final stages of the data stream via simulation studies. Overall, the results indicate that no single model space prior uniformly dominates across scenarios, and that the recently proposed MD prior provides a useful additional option that occupies an intermediate position between commonly used Beta–Binomial priors with differing degrees of sparsity. Representative R code is available at <https://github.com/joyeeghosh/model-space-priors-comparison>.

Key words: Bayesian model averaging; Beta–Binomial prior; High-dimensional; Inclusion probability; Matryoshka doll prior; Markov chain Monte Carlo.

AMS Subject Classifications: 62F15, 62J12

1. Introduction

BVS provides a principled framework for incorporating variable selection uncertainty by placing a prior distribution over a space of candidate models, corresponding to different combinations of predictor variables in a regression set up. In regression problems with many predictors, the choice of the model space prior can substantially influence posterior inference, particularly posterior inclusion probabilities and selected model sizes.

A widely adopted choice is the Beta–Binomial prior (Scott and Berger (2010)), which

is constructed by assigning a Beta prior to a common inclusion probability and assuming that, conditional on this probability, the predictor inclusion indicators are independent Bernoulli variables. One of the reasons for the popularity of this prior is better multiplicity control compared to the discrete uniform prior (Scott and Berger (2010)) when there are many predictors, and its associated computational convenience in posterior computation. Recently, the MD prior has been proposed by Womack *et al.* (2025) as an alternative construction motivated by theoretical considerations of sparsity and multiplicity adjustment for large model spaces.

The outline of this paper is as follows. In Section 2, we provide a brief review of several model space priors that are commonly used or have been recently proposed in the BVS literature. In Section 3, we propose a simple approximation to the MD prior that preserves its limiting sparsity behavior with independent inclusion indicators, making it attractive for posterior inference in scenarios where independence or conditional independence has been proven to be crucial, such as Expectation Maximization algorithms with spike-and-slab priors (Ročková and George (2014), Ročková and George (2018)). In Section 4, we compare these priors in the context of BVS for logistic regression with streaming data, focusing on posterior behavior at an intermediate batch and the final batch using BIC-based approximations to marginal likelihoods as in Ghosh *et al.* (2025), De *et al.* (2025). Finally, in Section 5 we summarize our main findings and discuss some avenues for future research.

2. Model space priors: A brief review

Let p denote the number of candidate predictors excluding the intercept, which is always included in the model and is not subject to selection. Let $\gamma = (\gamma_1, \dots, \gamma_p)^\top$ denote a vector of inclusion indicators, where $\gamma_j = 1$ indicates the inclusion of predictor j and $\gamma_j = 0$ indicates the exclusion of predictor j . Let $k_\gamma = \sum_{j=1}^p \gamma_j$ denote the model size. Model space priors can be described by putting priors on the inclusion indicators or through the induced prior distribution on model size.

2.1. Discrete uniform prior

The discrete uniform prior assigns equal probability to all 2^p possible models, and is given by

$$p(\gamma) = \frac{1}{2^p}.$$

Although simple and seemingly noninformative, this prior does not explicitly favor sparse models and can place substantial prior mass on large models even when p is moderate (Clyde and George (2004)). It was shown by Scott and Berger (2010) that this prior does not correct for multiplicity at all.

2.2. Beta–Binomial prior

Under a Beta–Binomial prior, the inclusion indicators are conditionally independent given an inclusion probability w ,

$$\gamma_j \mid w \stackrel{iid}{\sim} \text{Bernoulli}(w) \text{ for } j = 1, 2, \dots, p, \quad w \sim \text{Beta}(a, b),$$

where $a > 0, b > 0$. On integrating out w , this induces a Beta–Binomial distribution on model size (Wilson *et al.* (2010)). Common choices include Beta(1, 1) (Scott and Berger (2010)), which corresponds to a uniform prior on w , as well as Beta(1, p) and Beta(1, p^2), which increasingly favor sparse models as p grows. Alternative recommendations have included Beta(1, $\sqrt{p}$) by Li (2013), which was implemented by De and Ghosh (2026) because they found that it led to improved posterior behavior under heavy-tailed error distributions, compared to Beta(1, 1) when p was around a hundred. These priors are attractive due to their simplicity and ease of implementation, and they remain among the most commonly used choices in practice.

2.3. Matryoshka doll prior

The MD prior was recently proposed by Womack *et al.* (2025) as a model space prior based on a simple and principled construction. Its defining feature is that, for any model, the prior odds of that model against the collection of all models that nest it is fixed at a constant value, denoted by ξ , and does not depend on the total number of available predictors. This construction provides an explicit form of multiplicity control, particularly suited to settings in which the true model is assumed to have finite size while the number of candidate predictors may grow. While this principle is theoretically appealing, it does not yield a closed-form expression for the model space prior. Instead, the MD prior probabilities can be computed recursively, which is feasible for moderately large values of p . For very large model spaces, direct computation becomes impractical, but the authors show that the prior can be accurately approximated using its limiting distribution as p increases.

In particular, Womack *et al.* (2025) showed that as $p \rightarrow \infty$, the model size distribution induced by the MD prior converges to a Poisson(θ) distribution, where $\theta = \log(1 + (1/\xi))$. In particular,

$$\lim_{p \rightarrow \infty} p(k_\gamma = k) = e^{-\theta} \frac{\theta^k}{k!} \text{ for } k = 0, 1, 2, \dots,$$

and for finite p the induced distribution can be approximated by a Poisson(θ) distribution truncated to $\{0, 1, \dots, p\}$. From a practical standpoint, however, the MD prior induces dependence among inclusion indicators and cannot be naturally expressed as a collection of independent (or conditionally independent) Bernoulli variables. This dependence can complicate its implementation in algorithms in large-scale settings that benefit from conditional independence, such as EM algorithms with spike-and-slab priors or Gibbs samplers that exploit independence structure (Ghosh and Clyde (2011)). This is the motivation for us to consider an approximation to the MD prior that can be expressed in an independent Bernoulli form.

3. A Bernoulli approximation to the MD prior

In this section, we propose a simple and computationally convenient approximation to the MD prior with parameter $\theta > 0$, obtained by assuming independent inclusion indicators. Specifically, we propose

$$\gamma_j | \theta \stackrel{iid}{\sim} \text{Bernoulli}(\theta/p) \text{ for } j = 1, \dots, p,$$

which implies that $k_\gamma | \theta \sim \text{Binomial}(p, \theta/p)$.

We show below that this approximation has some of the asymptotic properties of the MD prior with parameter θ , while differing in its finite p properties.

Proposition 1: Fix $\theta > 0$. Suppose $\gamma_j | \theta \stackrel{iid}{\sim} \text{Bernoulli}(\theta/p)$ for $j = 1, \dots, p$, and $k_\gamma = \sum_{j=1}^p \gamma_j$. Then, for each fixed integer $k \geq 0$,

$$\lim_{p \rightarrow \infty} p(k_\gamma = k | \theta) = e^{-\theta} \frac{\theta^k}{k!}.$$

The proof is given in Appendix A.

Remark 1: Proposition 1 shows that the asymptotic distribution of k_γ is the same under the MD(θ) prior and the Bernoulli(θ/p) prior. Note that this is a special case of the standard Poisson approximation to a Binomial ($p, \theta/p$) distribution. Since this approximation is based on the limiting condition $p \rightarrow \infty$, its accuracy may deteriorate when p is very small.

Proposition 2: Fix $\theta > 0$. Suppose $\gamma_j | \theta \stackrel{iid}{\sim} \text{Bernoulli}(\theta/p)$ for $j = 1, \dots, p$, and let $k_\gamma = \sum_{j=1}^p \gamma_j$. Then, for each fixed integer $k \geq 0$,

$$\lim_{p \rightarrow \infty} \frac{p(k_\gamma = k + 1 | \theta)}{p(k_\gamma = k | \theta)} = \frac{\theta}{k + 1}.$$

The proof is provided in Appendix A.

Remark 2: Note that the limiting value of the model size ratio in Proposition 2 coincides with that induced by the MD prior with parameter θ , as shown by Womack *et al.* (2025). Here, model size refers to the number of predictors subject to variable selection that are included in the model; this matches the notion of model complexity in Womack *et al.* (2025), with our formulation corresponding to the special case in which only the intercept is treated as fixed, whereas their framework allows an arbitrary set of predictors to be always included. In particular, Proposition 7 of Womack *et al.* (2025) establishes that under a Beta–Binomial prior obtained by placing a Beta($1, b$) prior on the common inclusion probability, the limiting value of the same model size ratio as $p \rightarrow \infty$ equals (i) 1 when b is fixed, (ii) $1/(m+1)$ when $b = mp$, and (iii) p^{1-v} when $b = p^v$, $v > 1$.

Interpretation of limiting model size ratios: The above results provide insight about how different model space priors penalize increases in model size. The Beta($1, 1$) prior on the common inclusion probability does not impose any additional penalty on expanding the model, as the prior odds of adding a predictor remain constant. The Beta($1, p$) prior induces a constant penalty on model expansion that depends on the parameter m , leading to uniform shrinkage toward smaller models. In contrast, the Beta($1, p^2$) prior yields a penalty that grows with p , which can result in aggressive shrinkage toward very sparse models. The MD(θ) prior and the Bernoulli(θ/p) approximation induce an adaptive penalty that depends on the current model size, leading to a gradual increase in resistance to model expansion as the model becomes larger.

While this Bernoulli approximation does not preserve the full structural properties of the MD prior, it yields independent inclusion indicators and is therefore particularly

convenient for algorithms such as EM-type methods (Ročková and George (2014)). In what follows, we assess how well this approximation mimics the behavior of the MD prior in practice.

4. Bayesian variable selection for streaming logistic regression

In this section, we evaluate the model space priors described above in the context of BVS for logistic regression with streaming data. We focus on settings with $p = 10$ and $p = 15$ predictors (in addition to the intercept, which is always included), for which the model space with 2^p models is small enough to be enumerated, allowing Bayesian model averaging (BMA) without Markov chain Monte Carlo (MCMC).

4.1. Offline and online inference for streaming data

We assume that observations arrive sequentially in batches $D_1, D_2, \dots, D_b, \dots$, where batch b contains n_b observations. Let $D_b^* = \bigcup_{t=1}^b D_t$ denote the cumulative data up to batch b . We compare two approaches for posterior inference over the model space:

- **Offline** refers to recomputing posterior quantities at each batch using the entire cumulative dataset D_b^* . This serves as a gold standard but requires storing all past observations and refitting each candidate model as new data arrive.
- **Online** refers to the Online 2 method of Ghosh *et al.* (2025), which updates posterior quantities sequentially using only the current batch data and a small set of summary statistics from previous batches. This method is based on a second order Taylor expansion of the log-likelihood and provides a more accurate approximation than the first-order online method proposed in the same paper by Ghosh *et al.* (2025). In the rest of this paper, Online refers exclusively to this second-order method.

4.2. Marginal likelihood approximation and posterior computation

For a given model $\gamma = (\gamma_1, \dots, \gamma_p)^\top$, let X_γ denote the design matrix containing the intercept and predictors with $\gamma_j = 1$, and let β_γ be the corresponding vector of regression coefficients. The logistic regression model is

$$p(y_i = 1 \mid x_{i,\gamma}, \gamma, \beta_\gamma) = \frac{\exp(x_{i,\gamma}^\top \beta_\gamma)}{1 + \exp(x_{i,\gamma}^\top \beta_\gamma)},$$

where $x_{i,\gamma}^\top$ denotes the i th row of X_γ . Posterior model probabilities at batch b are given by

$$p(\gamma \mid D_b^*) \propto p(\gamma) m(D_b^* \mid \gamma),$$

where $p(\gamma)$ is the model space prior and $m(D_b^* \mid \gamma)$ is the marginal likelihood under model γ . To avoid specifying an explicit prior on β_γ and to facilitate repeated batch updates, we approximate the marginal likelihood using a BIC-based approximation as in Ghosh *et al.* (2025) as follows:

$$\log m(D_b^* \mid \gamma) \approx \ell_b(\hat{\beta}_{\gamma,b}) - \frac{k_\gamma + 1}{2} \log N_b,$$

where $\ell_b(\hat{\beta}_{\gamma,b})$ is the log-likelihood evaluated at the maximum likelihood estimate $\hat{\beta}_{\gamma,b}$ (or its online estimate under Online), k_γ is the number of included predictors (not including the intercept), and N_b is the aggregate sample size at batch b .

Posterior inclusion probabilities are computed as

$$p(\gamma_j = 1 \mid D_b^*) = \sum_{\gamma: \gamma_j=1} p(\gamma \mid D_b^*),$$

and BMA estimates of regression coefficients are obtained by taking weighted averages of $\hat{\beta}_{\gamma,b}$ over models, with coefficients set to zero when excluded from a model, and the weights being equal to the posterior probabilities of models.

4.3. Model space priors under comparison

The following model space priors are compared, ordered from relatively weak to relatively strong sparsity-inducing property, with the corresponding abbreviations provided in parentheses:

1. Discrete uniform prior on the model space (DU).
2. Beta(1, 1) prior on the common inclusion probability (B11).
3. Beta(1, p) prior on the common inclusion probability (B1p).
4. MD prior with $\theta = 1$ (MD).
5. Truncated Poisson approximation to the MD prior with $\theta = 1$ (PA).
6. Bernoulli(θ/p) approximation proposed in Section 3 with $\theta = 1$ (BA).
7. Beta(1, p^2) prior on the common inclusion probability (B1psq).

The choice $\theta = 1$ is adopted here, which is the default specification used by Womack *et al.* (2025) and corresponds to a Poisson(1) limiting distribution for model size. This value provides a natural baseline for sparsity and facilitates comparison between the MD prior and its approximations.

4.4. Simulation scenarios

We examine four simulation scenarios with two values of p (10 and 15) and two sparsity regimes (sparse versus non-sparse). In all cases, covariates are generated independently from a normal distribution with mean 0 and variance 9. The intercept term is always included in the model. For each scenario, we generate 25 independent datasets.

Scenarios with $p = 10$: These scenarios are implemented with $p = 10$ covariates, other than the intercept. Batch sizes are taken as $n_1 = 50$ and $n_b = 10$ for $b = 2, \dots, 21$, as the first batch size needs to be sufficiently large for the online method to work well (Luo and Song (2020); Ghosh *et al.* (2025), De *et al.* (2025)). Let $\beta = (\beta_0, \beta_1, \dots, \beta_{10})^\top$, where β_0 is the intercept term. We have the following two cases:

- **Sparse:** $\beta = (0.2, 0.3, -0.4, 0, \dots, 0)^\top$.
- **Non-sparse:** $\beta = (0.2, 0.2, 0.2, 0.2, 0.2, -0.2, -0.2, -0.2, 0, 0, 0)^\top$.

Scenarios with $p = 15$: Here we consider batch sizes $n_1 = 100$ and $n_b = 10$ for $b = 2, \dots, 21$ with the following two cases:

- **Sparse:** $\beta = (0.2, 0.3, 0.4, 0, \dots, 0)^\top$.
- **Non-sparse:** $\beta = (0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0, \dots, 0)^\top$.

4.5. Performance measures

For each scenario and model space prior, we summarize the performance at an intermediate batch ($b = 11$) and the final batch ($b = 21$). Specifically, we report:

- Root mean squared error (RMSE) in estimating the true regression coefficient vector β using BMA estimates of β .
- RMSE in estimating the true model γ using marginal posterior inclusion probabilities.

Box plots showing the values of RMSE across 25 replicates are used to summarize the variability for Offline and Online. We do not include running time comparisons in this study, as meaningful computational gains primarily arise for substantially larger batch sizes. In such regimes, however, posterior behavior under different model space priors becomes increasingly similar, making it difficult to isolate their individual effects.

4.6. Summary of simulation results

The box plots for the four simulation scenarios at two batches ($b = 11$ and $b = 21$) under each scenario are provided in Figures 1–8. We provide an overall summary of the results here. Across all scenarios, Offline and Online produce very similar posterior behavior, with Online closely tracking Offline even at intermediate batches. This agreement persists across different model space priors and both sparse and non-sparse regimes, indicating that the online procedure provides an accurate approximation to offline inference in the settings considered here. Differences among model space priors are evident in both sparse and non-sparse settings, though their nature differs substantially across regimes.

In sparse scenarios, priors that more strongly favor sparse models, such as the $\text{Beta}(1, p^2)$ prior and, to a smaller extent, the MD prior and its two approximations, tend to achieve lower RMSE for posterior inclusion probabilities by more effectively controlling false positives. In contrast, priors that induce little or no multiplicity control, such as the discrete uniform prior and the $\text{Beta}(1, 1)$ prior, exhibit higher RMSE for inclusion probabilities due to the inclusion of additional noise predictors. The $\text{Beta}(1, p)$ prior typically falls between these two extremes, reflecting its intermediate level of sparsity promotion.

In non-sparse scenarios, this pattern is largely reversed. Priors that strongly favor small models tend to incur higher RMSE for inclusion probabilities due to the exclusion of

truly active predictors, resulting in false negatives. In these settings, less sparsity-inducing priors, most notably the discrete uniform and Beta(1, 1) priors, perform better in terms of inclusion probability RMSE, while the Beta(1, p^2) prior performs the worst. The MD prior and its approximations again exhibit intermediate behavior, neither consistently dominating nor performing poorly across all cases.

Across all scenarios, differences among model space priors have a much larger impact on the estimation of posterior inclusion probabilities than on the estimation of regression coefficients. In the sparse settings, even when false positives occur under less sparsity-promoting priors, the corresponding regression coefficients are typically shrunk toward zero, leading to relatively modest differences in RMSE for coefficient estimation. In contrast, in the non-sparse scenarios, differences in coefficient RMSE become more pronounced: priors that aggressively favor sparsity tend to drop truly nonzero coefficients, resulting in higher RMSE due to false negatives, whereas priors that allow larger models achieve better coefficient recovery.

The two approximations to the MD prior: (1) the truncated Poisson approximation proposed by Womack *et al.* (2025), and (2) the Bernoulli(θ/p) approximation proposed by us in this paper, track the behavior of the MD prior reasonably closely across all scenarios considered, with the posterior under approximation (1) being almost indistinguishable from the posterior under the MD prior, whereas (2) shows a small difference. However, differences among these three priors are generally small relative to differences between broader classes of model space priors, suggesting that the proposed Bernoulli(θ/p) approximation provides a reasonable and computationally convenient surrogate for the MD prior in finite- p settings, where independence among inclusion indicators may be vital for computational reasons in an EM algorithm.

Finally, we note that this simulation study uses 25 replicates per scenario. Therefore, the observed ordering patterns among priors should be interpreted with some caution, particularly in cases where the corresponding box plots exhibit substantial overlap or spread. A more extensive simulation study could further assess the stability of these patterns, but this is beyond the scope of the present work.

5. Discussion

In this paper, we examined the role of the model space prior in BVS for logistic regression, with a particular emphasis on streaming data settings. We reviewed several commonly used priors, including the discrete uniform and Beta–Binomial families, alongside the recently proposed MD prior (Womack *et al.* (2025)). We also introduced a simple Bernoulli(θ/p) approximation to the MD prior that preserves its limiting sparsity behavior while yielding independent inclusion indicators, which is attractive for scalable inference in EM-type algorithms.

From a theoretical perspective, the MD prior provides an appealing framework for multiplicity control, inducing Poisson-type behavior in model size and an adaptive penalty for adding an additional predictor that depends on the current model size rather than directly on the total number of predictors. In contrast, for adding a predictor to a current model, Beta–Binomial priors impose penalties whose strength depends on how the Beta hyperparameters change with p , ranging from no additional penalty under the Beta(1, 1) prior

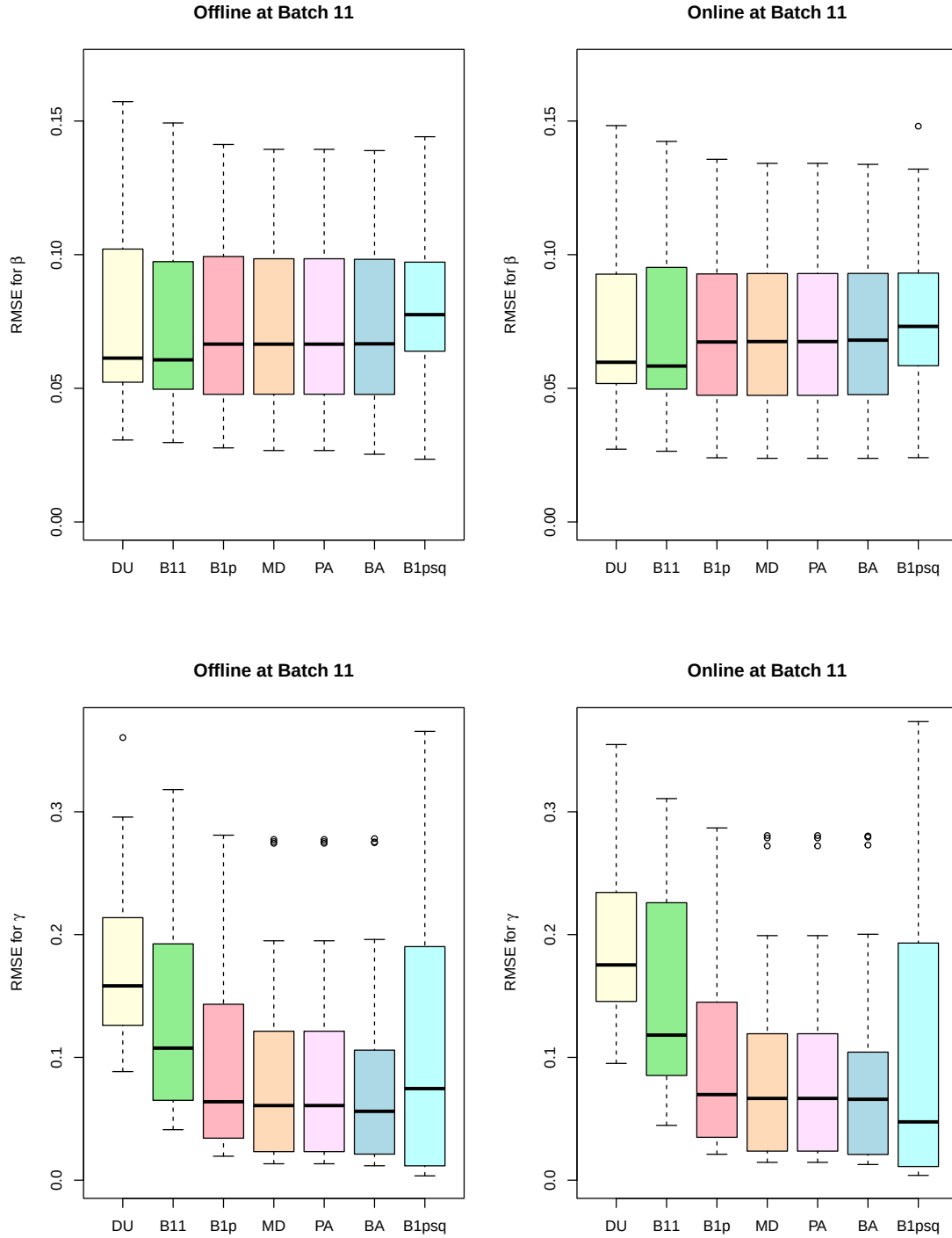


Figure 1: Box plots showing RMSE for estimating β and γ at batch 11, across 25 replicates for Offline and Online methods, for the *sparse simulation scenario with $p = 10$* under the 7 model space priors described in Section 4.3.

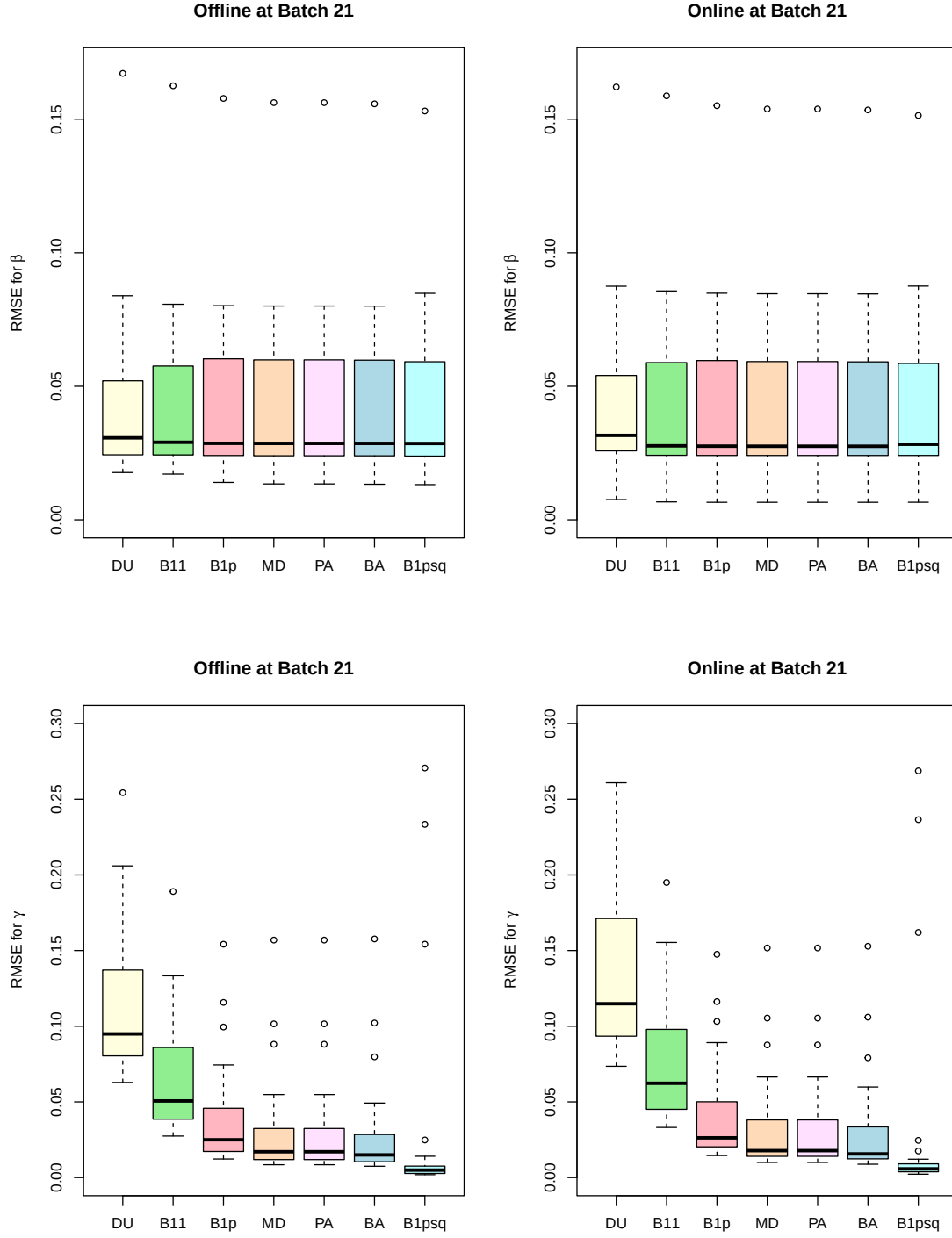


Figure 2: Box plots showing RMSE for estimating β and γ at batch 21, across 25 replicates for Offline and Online methods, for the *sparse simulation scenario with $p = 10$* under the 7 model space priors described in Section 4.3.

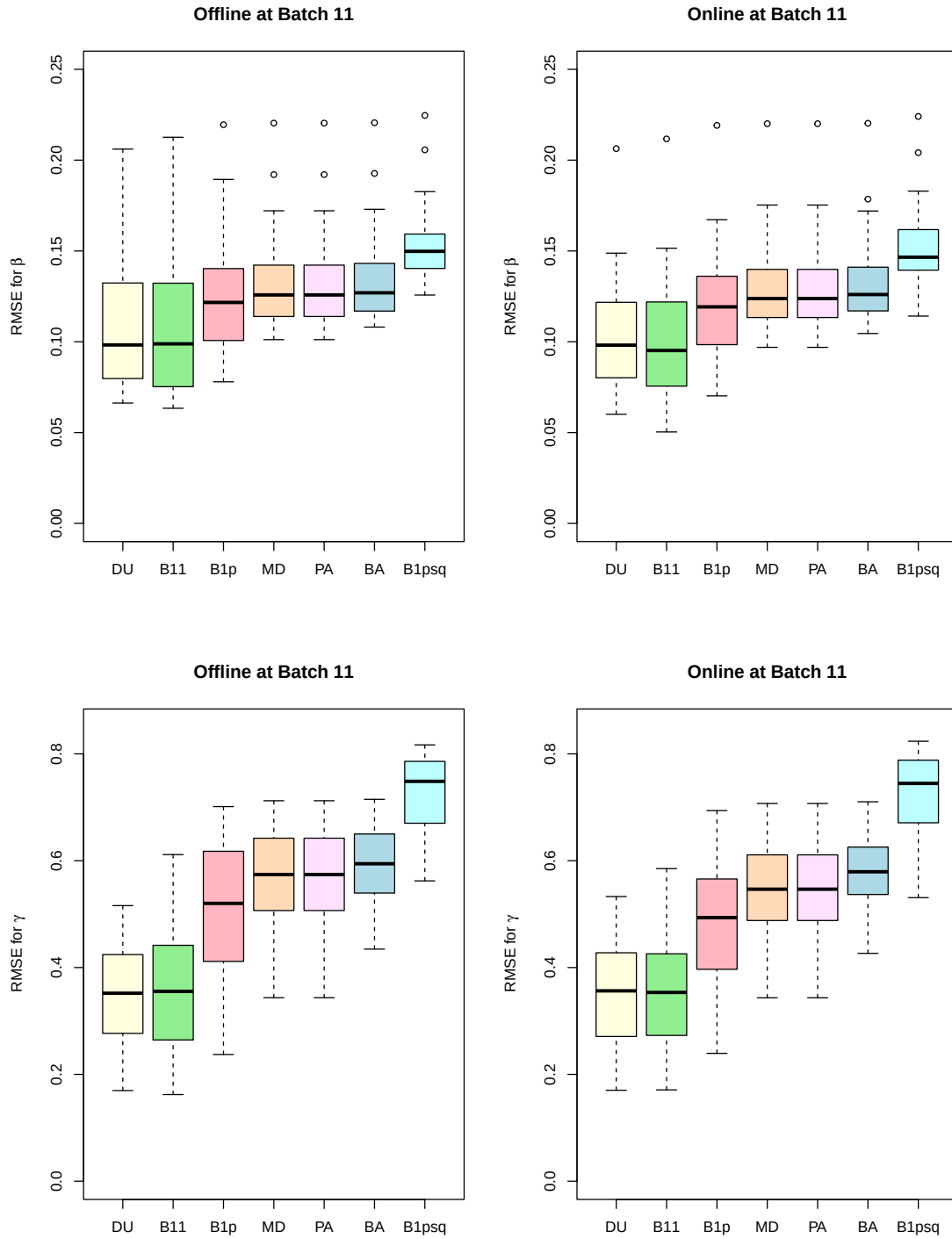


Figure 3: Box plots showing RMSE for estimating β and γ at batch 11, across 25 replicates for Offline and Online methods, for the *non-sparse simulation scenario with $p = 10$* under the 7 model space priors described in Section 4.3.

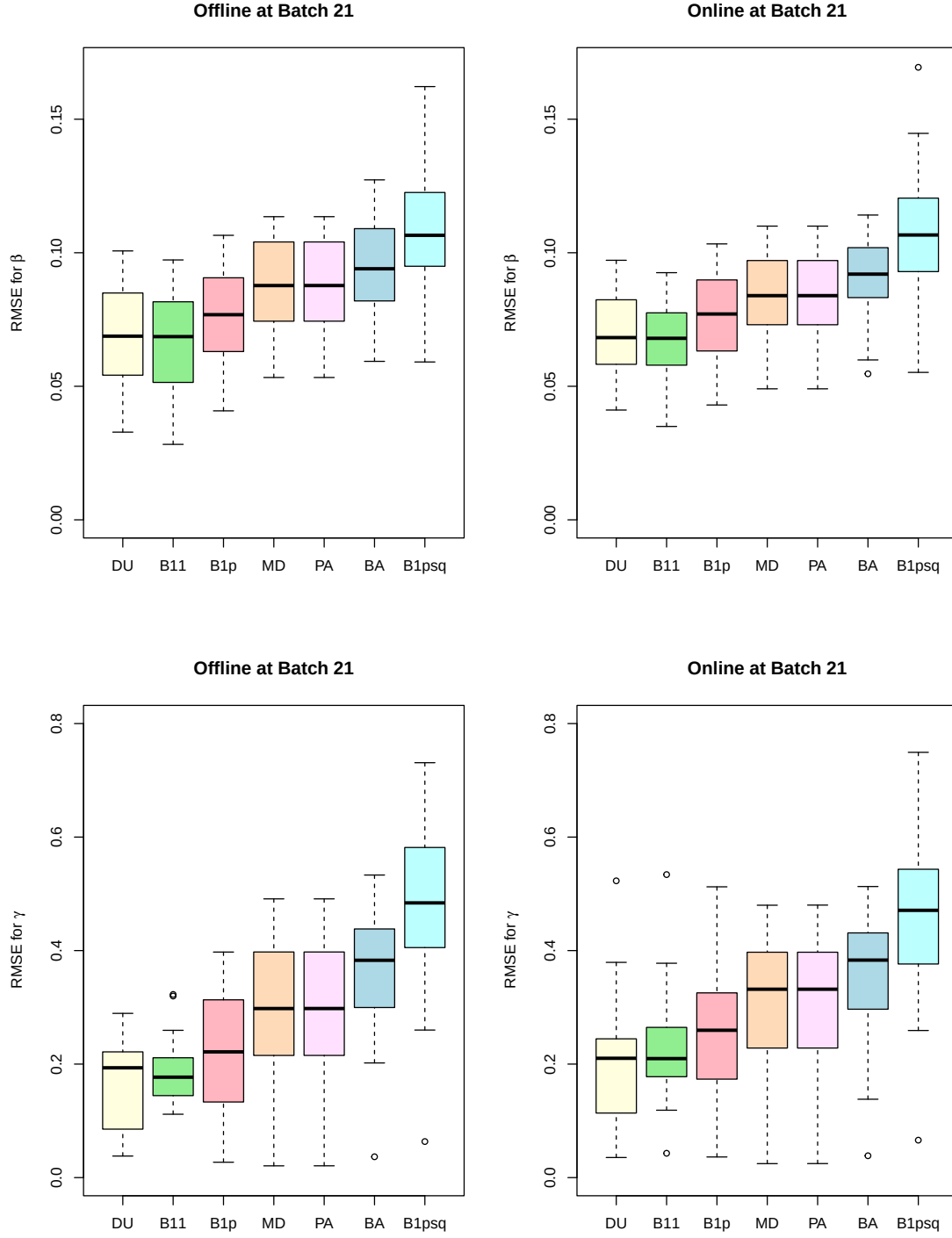


Figure 4: Box plots showing RMSE for estimating β and γ at batch 21, across 25 replicates for Offline and Online methods, for the non-sparse simulation scenario with $p = 10$ under the 7 model space priors described in Section 4.3.

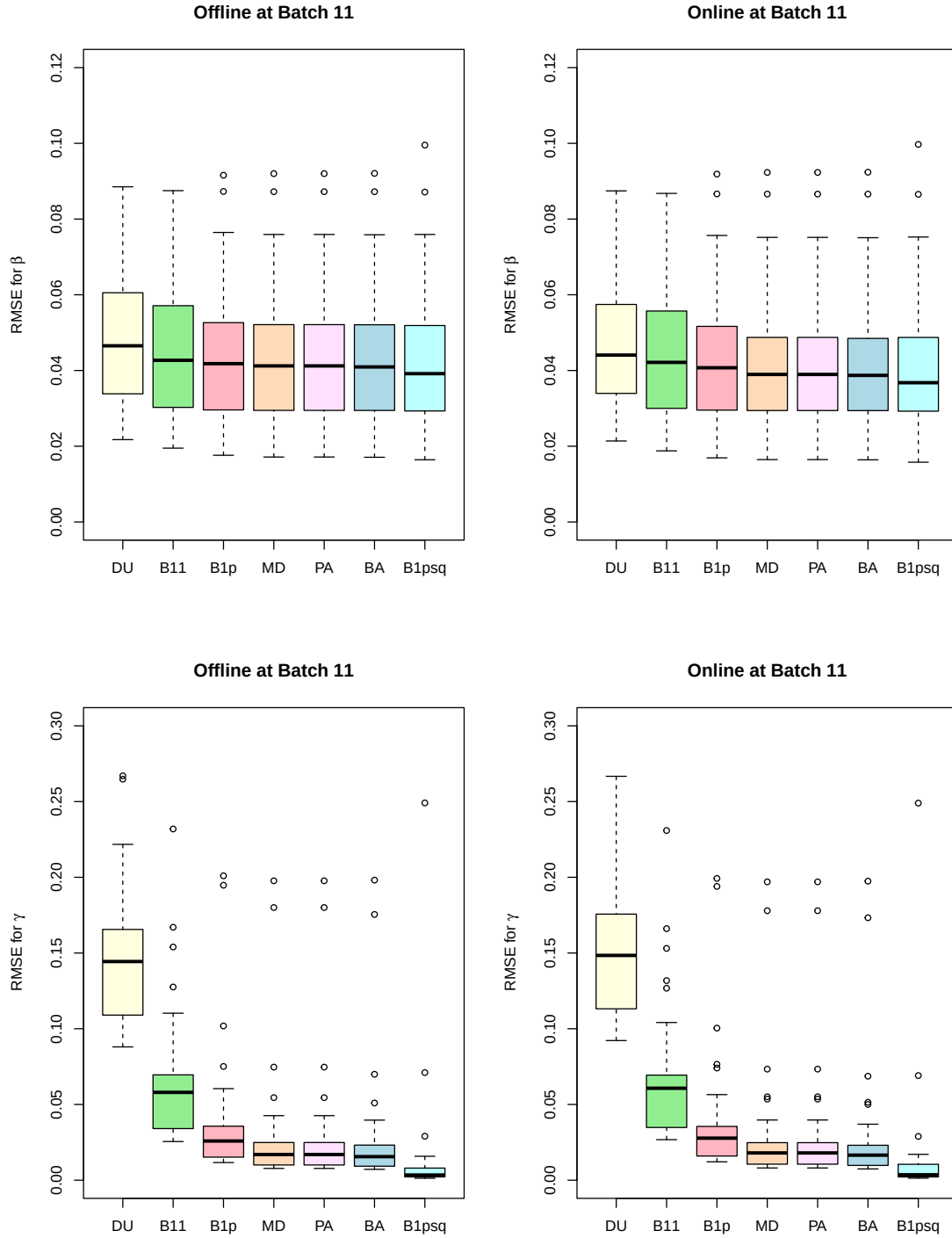


Figure 5: Box plots showing RMSE for estimating β and γ at batch 11, across 25 replicates for Offline and Online methods, for the *sparse simulation scenario with $p = 15$* under the 7 model space priors described in Section 4.3.

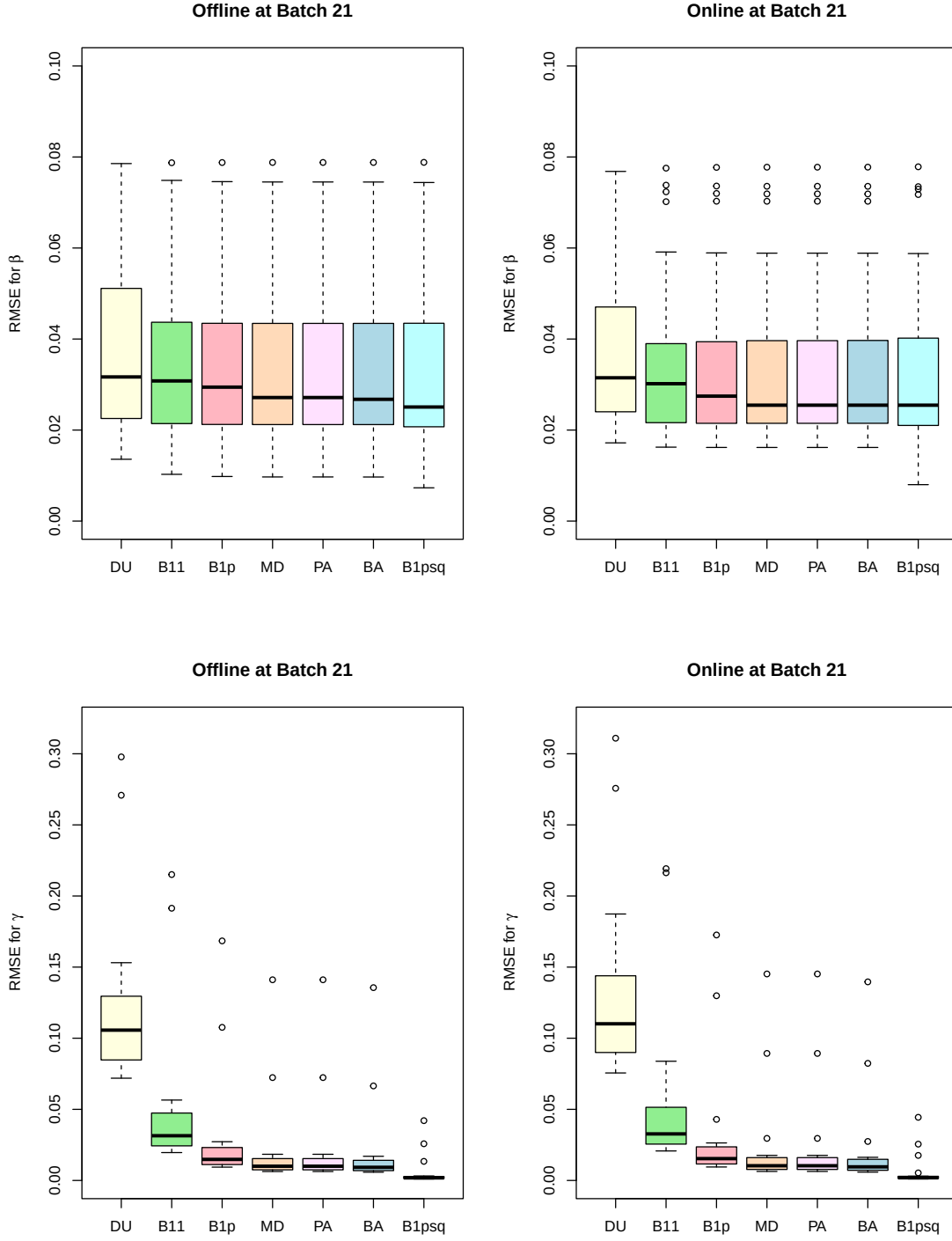


Figure 6: Box plots showing RMSE for estimating β and γ at batch 21, across 25 replicates for Offline and Online methods, for the *sparse simulation scenario with $p = 15$* under the 7 model space priors described in Section 4.3.

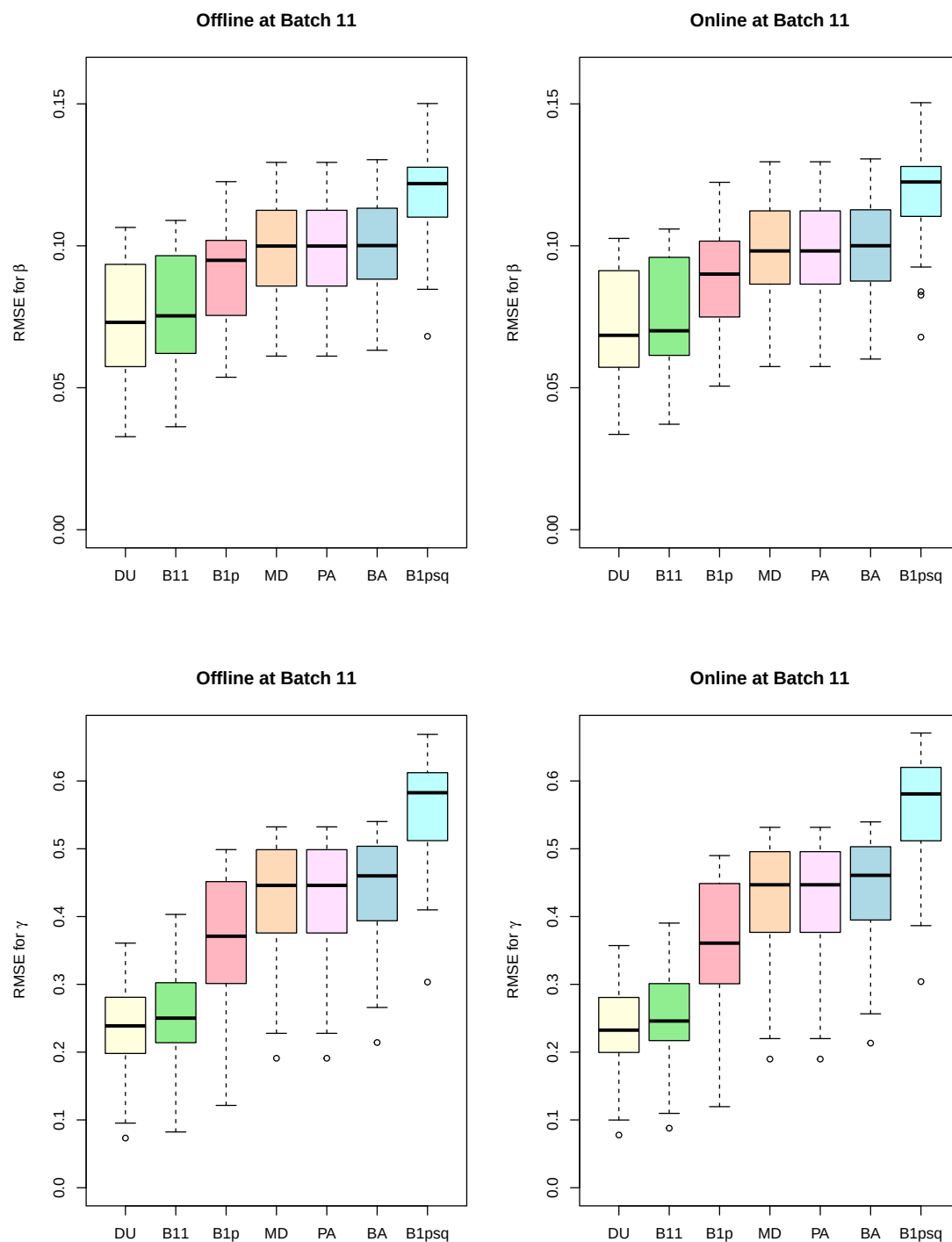


Figure 7: Box plots showing RMSE for estimating β and γ at batch 11, across 25 replicates for Offline and Online methods, for the *non-sparse simulation scenario with $p = 15$* under the 7 model space priors described in Section 4.3.

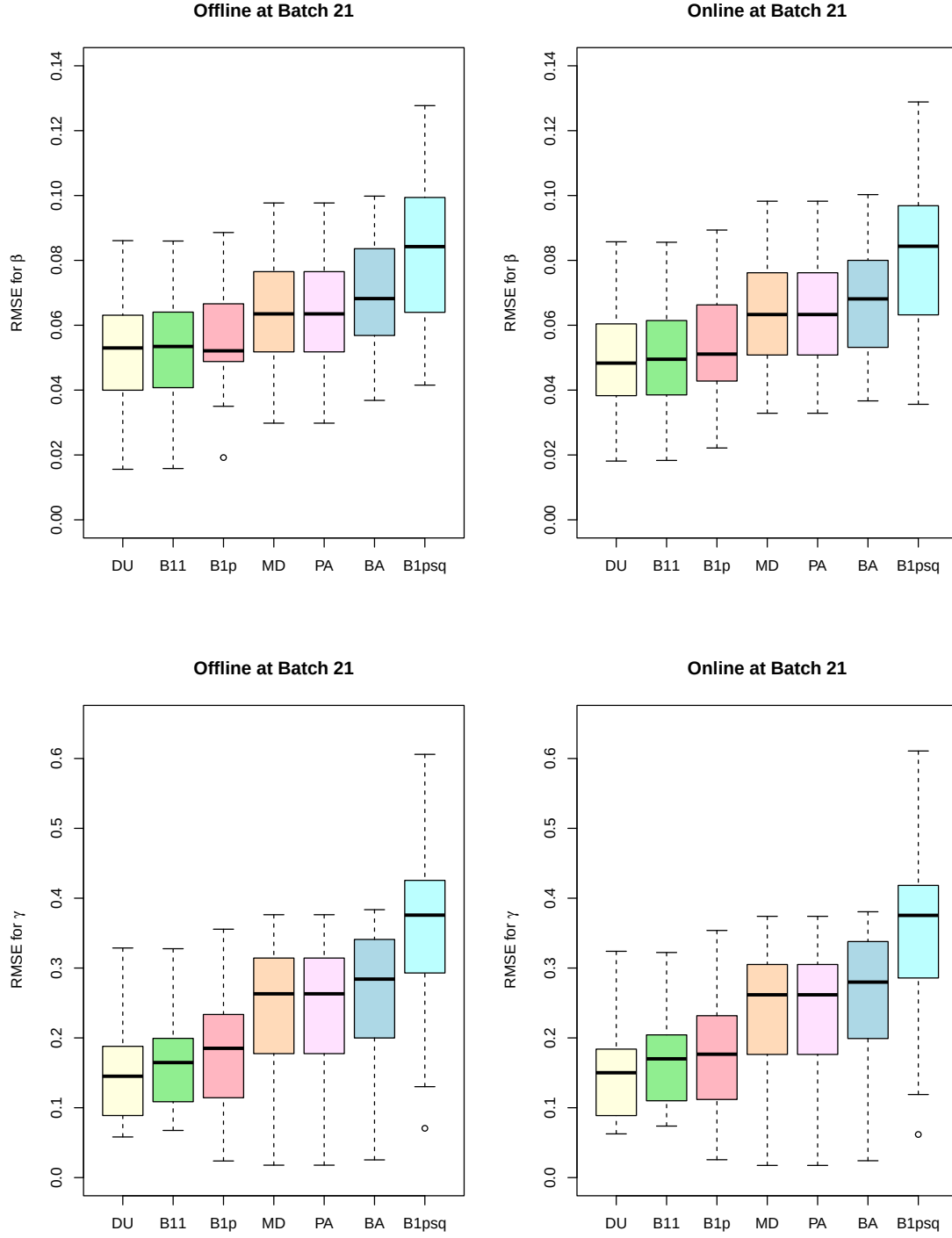


Figure 8: Box plots showing RMSE for estimating β and γ at batch 21, across 25 replicates for Offline and Online methods, for the *non-sparse simulation scenario with $p = 15$* under the 7 model space priors described in Section 4.3.

to very aggressive penalty under the $\text{Beta}(1, p^2)$ prior. The $\text{Beta}(1, p)$ prior occupies an intermediate position, imposing a constant penalty on the prior odds of adding a predictor. Our results suggest that in finite- p regimes with relatively modest values of p , the practical behavior of the MD prior is often comparable to that of the $\text{Beta}(1, p)$ prior, though the two arise from different conceptual motivations.

Our empirical results under logistic regression and streaming data illustrate that no single model space prior uniformly dominates across scenarios. In sparse settings, stronger sparsity-inducing priors tend to better control false positives, whereas, in non-sparse settings, milder priors often perform better by avoiding false negatives. The MD prior and its approximations generally exhibit behavior between these extremes, consistent with their theoretical construction. The two approximations to the MD prior track its behavior closely in finite- p regimes, with the truncated Poisson approximation being nearly indistinguishable from the MD prior and the Bernoulli(θ/p) approximation showing only minor deviations while offering substantial computational convenience.

The settings considered here involve relatively small model spaces that allow the enumeration of all candidate models with BIC-based marginal likelihood approximations. An important direction for future work is to investigate the behavior of the MD prior and its approximations in settings with larger values of p , where full model space enumeration is computationally infeasible and posterior inference relies on MCMC or other stochastic search methods (De *et al.* (2025)). Understanding how the theoretical advantages of the MD prior translate to such high-dimensional regimes remains an interesting and open problem.

References

- Clyde, M. and George, E. I. (2004). Model uncertainty. *Statistical Science*, **19**, 81–94.
- De, S., Ghosal, P., and Ghosh, J. (2025). An online algorithm for Bayesian variable selection in logistic regression models with streaming data. *Sankhya B*, **Online first article**, To appear.
- De, S. and Ghosh, J. (2026). Robust Bayesian model averaging for linear regression models with heavy-tailed errors. *Journal of Applied Statistics*, **53**, 304–330.
- Ghosh, J. and Clyde, M. A. (2011). Rao-Blackwellization for Bayesian variable selection and model averaging in linear and binary regression: A novel data augmentation approach. *Journal of the American Statistical Association*, **106**, 1041–1052.
- Ghosh, J., Tan, A., and Luo, L. (2025). Online Bayesian variable selection and Bayesian model averaging for streaming data. *Stat*, **14**, e70044.
- Li, Y. (2013). *Bayesian Hierarchical Models for Model Choice*. PhD thesis, Duke University.
- Luo, L. and Song, P. X.-K. (2020). Renewable estimation and incremental inference in generalized linear models with streaming data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **82**, 69–97.
- Ročková, V. and George, E. (2014). EMVS: The EM approach to Bayesian variable selection. *Journal of the American Statistical Association*, **109**, 828–846.
- Ročková, V. and George, E. I. (2018). The spike-and-slab lasso. *Journal of the American Statistical Association*, **113**, 431–444.

- Scott, J. G. and Berger, J. O. (2010). Bayes and empirical-Bayes multiplicity adjustment in the variable-selection problem. *Annals of Statistics*, **38**, 2587–2619.
- Wilson, M. A., Iversen, E. S., Clyde, M. A., Schmidler, S. C., and Schildkraut, J. M. (2010). Bayesian model search and multilevel inference for SNP association studies. *Annals of Applied Statistics*, **4**, 1342–1364.
- Womack, A. J., Taylor-Rodriguez, D., and Fuentes, C. (2025). The Matryoshka doll prior - principled multiplicity correction in Bayesian model comparison. *Bayesian Analysis, Advance Publication*, 1–25.

Appendix A

Proposition 1: Fix $\theta > 0$. Suppose $\gamma_j | \theta \stackrel{iid}{\sim} \text{Bernoulli}(\theta/p)$ for $j = 1, \dots, p$, and $k_\gamma = \sum_{j=1}^p \gamma_j$. Then, for each fixed integer $k \geq 0$,

$$\lim_{p \rightarrow \infty} p(k_\gamma = k | \theta) = e^{-\theta} \frac{\theta^k}{k!}.$$

Proof: For each p , $k_\gamma | \theta \sim \text{Binomial}(p, \theta/p)$, hence for fixed k ,

$$p(k_\gamma = k | \theta) = \binom{p}{k} \left(\frac{\theta}{p}\right)^k \left(1 - \frac{\theta}{p}\right)^{p-k}. \quad (1)$$

We simplify the product of the first two terms in the right hand side (RHS) of (1) below.

$$\binom{p}{k} \left(\frac{\theta}{p}\right)^k = \frac{\theta^k}{k!} \cdot \frac{p(p-1) \cdots (p-k+1)}{p^k} = \frac{\theta^k}{k!} \prod_{j=0}^{k-1} \left(1 - \frac{j}{p}\right).$$

Since k is fixed, each factor in the above product term converges to 1 as $p \rightarrow \infty$, and we have

$$\lim_{p \rightarrow \infty} \binom{p}{k} \left(\frac{\theta}{p}\right)^k = \frac{\theta^k}{k!}. \quad (2)$$

We now simplify the third term in the RHS of (1) below.

$$\left(1 - \frac{\theta}{p}\right)^{p-k} = \left(1 - \frac{\theta}{p}\right)^p \left(1 - \frac{\theta}{p}\right)^{-k}.$$

Because θ is fixed, the standard limit $(1 - \theta/p)^p \rightarrow e^{-\theta}$ holds as $p \rightarrow \infty$. Moreover, since k is fixed,

$$\lim_{p \rightarrow \infty} \left(1 - \frac{\theta}{p}\right)^{-k} = 1.$$

Combining these gives

$$\lim_{p \rightarrow \infty} \left(1 - \frac{\theta}{p}\right)^{p-k} = e^{-\theta}. \quad (3)$$

Combining (2) and (3) proves the claim. $\square$

Proposition 2: Fix $\theta > 0$. Suppose $\gamma_j|\theta \stackrel{iid}{\sim} \text{Bernoulli}(\theta/p)$ for $j = 1, \dots, p$, and let $k_\gamma = \sum_{j=1}^p \gamma_j$. Then, for each fixed integer $k \geq 0$,

$$\lim_{p \rightarrow \infty} \frac{p(k_\gamma = k+1|\theta)}{p(k_\gamma = k|\theta)} = \frac{\theta}{k+1}.$$

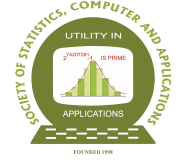
Proof: Since $k_\gamma|\theta \sim \text{Binomial}(p, \theta/p)$,

$$p(k_\gamma = k|\theta) = \binom{p}{k} \left(\frac{\theta}{p}\right)^k \left(1 - \frac{\theta}{p}\right)^{p-k}.$$

Therefore, for fixed $k \geq 0$,

$$\begin{aligned} \frac{p(k_\gamma = k+1|\theta)}{p(k_\gamma = k|\theta)} &= \frac{\binom{p}{k+1} \left(\frac{\theta}{p}\right)^{k+1} \left(1 - \frac{\theta}{p}\right)^{p-k-1}}{\binom{p}{k} \left(\frac{\theta}{p}\right)^k \left(1 - \frac{\theta}{p}\right)^{p-k}} \\ &= \frac{\binom{p}{k+1}}{\binom{p}{k}} \cdot \frac{\theta}{p} \cdot \left(1 - \frac{\theta}{p}\right)^{-1} \\ &= \frac{p-k}{k+1} \cdot \frac{\theta}{p} \cdot \frac{p}{p-\theta} \\ &= \frac{\theta}{k+1} \cdot \frac{p-k}{p-\theta}. \end{aligned}$$

Because θ and k are fixed, $(p-k)/(p-\theta) \rightarrow 1$ as $p \rightarrow \infty$, which proves the claim. $\square$



Function Optimization with Posterior Gaussian Derivative Process

Sucharita Roy¹ and Sourabh Bhattacharya²

¹*Department of Mathematics*

St. Xavier's College (Autonomous), Kolkata

²*Interdisciplinary Statistical Research Unit, Indian Statistical Institute, Kolkata*

Received: 1 September 2025; Revised: 24 May 2026; Accepted: 11 July 2026

Abstract

Function optimization is fundamental to all scientific disciplines, yet no single method dominates. We propose a Bayesian optimization algorithm for functions with known first and second partial derivatives. The objective function is modeled as a Gaussian process, inducing a Gaussian derivative process. Given a small set of initial function evaluations, we obtain the posterior of the derivative process and construct a posterior for stationary points by setting the derivative to zero. A recursive importance-resampling scheme with prior constraints on gradient norm and Hessian definiteness drives the algorithm toward the true optima.

We provide a rigorous theoretical foundation, proving almost sure convergence of the algorithm to the true optima as the number of stages tends to infinity. This relies on novel results establishing almost sure uniform convergence of the Gaussian process and its derivative process posteriors to the true function and its derivatives under fixed-domain infill asymptotics; rates of convergence are also derived. In addition, we give a Bayesian characterization of the number of optima using the information accumulated during the algorithm.

Five optimization problems (maxima, minima, saddle points, inconclusive cases) from one to 100 dimensions illustrate the method. On a real-world Poisson regression (AIDS deaths data), our procedure achieves a substantially smaller gradient norm at the MLE than Fisher scoring, BFGS, simulated annealing, and multi-start quasi-Newton. The posterior simulation nature of the algorithm yields consistently more accurate results. C/MPI code related to the AIDS data is available at https://github.com/Sourabh-Bhattacharya/FUNCTION_OPT_GDP.

Key words: Dirichlet process; Fixed-domain infill asymptotics; Function optimization; Gaussian derivative process; Importance resampling; Transformation based MCMC.

AMS Subject Classifications: 62F15, 62G08, 65K10

1. Introduction

Function optimization is a cornerstone of scientific computing, with applications across virtually every discipline. Despite decades of research, no single optimization method dominates all problem classes; for any sufficiently large family of problems, it is remarkably difficult to identify a methodology that consistently outperforms others in terms of theoretical foundation, accuracy, computational efficiency, or robustness. Consequently, practitioners often resort to heuristic methods tailored to the specific problem at hand.

Among stochastic optimization methods, simulated annealing is a general-purpose technique, but its practical success depends critically on carefully chosen temperature schedules and proposal distributions, both of which become extremely challenging to tune in moderate to high dimensions.

In this article, we propose and develop a novel Bayesian algorithm for optimizing functions whose first and second partial derivatives are available. Our method assumes that the objective function can be evaluated exactly (no observation noise) and that its first and second partial derivatives are available analytically or via automatic differentiation. Thus the setting is that of deterministic optimization, typical in computer experiments and engineering design, rather than stochastic optimization with noisy evaluations. Our approach embeds the objective function, together with its derivatives, into a random function framework driven by Gaussian processes and their induced derivative processes. With data comprising a set of input points and the corresponding function values, we first obtain the posterior derivative process. We then construct the posterior distribution of the solutions obtained by setting the random partial derivative functions to the null vector. This posterior is expected to emulate the stationary points of the objective function.

We further incorporate a uniform prior over the domain, constrained so that the first partial derivatives have small Euclidean norm and the Hessian matrix is positive definite (for minimization) or negative definite (for maximization). These constraints help the posterior solutions approximate the true optima even when the dataset is small. However, for very small datasets the prior may be too restrictive to allow effective posterior simulation. Therefore, we begin with a weaker restriction (for instance, bounding the gradient norm by a relatively large constant) to obtain an initial set of posterior solution simulations. Then, in successive iterative stages, we refine the prior restrictions (making them stricter), and at each stage we augment the dataset with new realizations (and their function values) that satisfy the current, tighter constraints. As the number of stages tends to infinity, the posterior distributions converge to the true optima. These ideas lead to a general, effective, and adaptive optimization algorithm.

The convergence of our algorithm hinges on the convergence of the posterior derivative process to the true derivatives of the objective function, which in turn depends on a suitable design of the input points. Under an appropriate fixed-domain infill asymptotics setup, we prove almost sure uniform convergence of the posterior Gaussian process and its derivative process to the objective function and its derivatives. Notably, we also obtain rates of convergence under a specific infill design. To the best of our knowledge, these results are new and of independent interest. Using them, we prove almost sure convergence of our optimization algorithm to the true optima as the number of stages grows. As an aside, we also provide a Bayesian characterization of the number of optima, exploiting information

accumulated by the algorithm.

We illustrate our method on five optimization problems that include finding maxima, minima, saddle points, and even inconclusive cases. The examples range from simple one-dimensional problems to challenging 50- and 100-dimensional problems. In each case we obtain encouraging and insightful results. Along the way we discuss various computational and accuracy issues. A key strength of our approach is its ability to achieve significantly more accurate solutions than existing optimization algorithms, thanks to the posterior simulation paradigm embedded in the method. Indeed, by exploring neighborhoods around any solution provided by other methods, our algorithm almost surely finds a point at least as close to the true optimum.

We note that existing Bayesian optimization methods using Gaussian processes typically treat the objective function as a black box and assume derivatives are unavailable (see, for example, Frazier (2018) and references therein). Those methods are naturally less accurate when derivatives are accessible. Moreover, we are not aware of any convergence results for such derivative-free Gaussian process methods, which rely on heuristic acquisition functions. Because many acquisition functions exist, each with its own advantages and drawbacks, those methods lack a solid theoretical foundation and may be unreliable in practice. In this article we do not further discuss those existing approaches.

The remainder of the paper is organized as follows. Section 2 provides the derivation of the posterior Gaussian derivative process. Section 3 derives the posterior distribution for the random optima obtained by setting the derivative process to zero, together with additional restrictions based on the objective function’s derivatives. Section 4 presents almost sure uniform convergence results for the Gaussian process and its derivative process posteriors, along with proofs. Section 5 introduces our general-purpose Bayesian optimization algorithm. Section 6 establishes a Bayesian characterization of the number of optima. Section 7 illustrates the algorithm on various optimization problems. Section 8 summarizes our contributions and offers concluding remarks.

2. Posterior Gaussian derivative process

2.1. Details of the objective function

Consider any function $f : \mathbb{R}^d \mapsto \mathbb{R}$, where $\mathbb{R}$ is the real line and $d (\geq 1)$ is the dimension of the input space, assumed to be finite. We further assume that the second order partial derivatives of f , namely, $\partial^2 f(\mathbf{x}) / \partial x_i \partial x_j$ exist and are continuous for all $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ for $i, j = 1, \dots, d$. The objective is to optimize the function $f(\mathbf{x})$ with respect to $\mathbf{x} \in \mathcal{X}$. For theoretical purposes, we assume that $\mathcal{X}$ is compact; this assumption is not required for implementation of our methodology.

2.2. Data from the objective function

Assume that corresponding to arbitrary inputs $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} \in \mathcal{X}$, where $n > 1$, the output vector $\mathbf{f}_n = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))^T$ is available. Here T denotes transpose. Let $\mathbf{D}_n = \{(\mathbf{x}_i, f(\mathbf{x}_i)) : i = 1, \dots, n\}$.

2.3. Gaussian process representation of the objective function

Let $g : \mathbb{R}^d \mapsto \mathbb{R}$ denote a random function such that given $\mathbf{D}_n$, $g(\mathbf{x}_i) = f(\mathbf{x}_i)$ for $i = 1, \dots, n$.

Since Gaussian processes have the above interpolation property, we model $g(\cdot)$ by a Gaussian process with mean function $\mu(\cdot)$ and covariance function $\sigma^2 c(\cdot, \cdot)$, where σ^2 is the process variance. In other words, $E[g(\mathbf{x})] = \mu(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ and $Cov(g(\mathbf{x}), g(\mathbf{y})) = \sigma^2 c(\mathbf{x}, \mathbf{y})$, for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Let $\mu(\cdot)$ be continuous in $\mathcal{X}$ and $c(\cdot, \cdot)$ be Lipschitz continuous on $\mathcal{X} \times \mathcal{X}$. Then $g(\cdot)$ is actually a continuous-path Gaussian process with mean function $\mu(\cdot)$ and covariance function $\sigma^2 c(\cdot, \cdot)$.

We shall use the notation $\mathbf{g}_n$ to denote $(g(\mathbf{x}_1), \dots, g(\mathbf{x}_n))^T$ when distribution of this vector will be considered and $\mathbf{f}_n$ when this vector is conditioned upon.

2.4. Gaussian derivative process

For $\mathbf{x} = (x_1, \dots, x_d)$ and $\mathbf{y} = (y_1, \dots, y_d)$, let us assume that the second order mixed partial derivatives

$$\frac{\partial^2 c(\mathbf{x}^*, \mathbf{y}^*)}{\partial x_i \partial y_i} = \frac{\partial^2 c(\mathbf{x}, \mathbf{y})}{\partial x_i \partial y_i} \Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{y}=\mathbf{y}^*}$$

are Lipschitz continuous on $\mathcal{X} \times \mathcal{X}$ for $i = 1, \dots, d$.

With the above assumption on the covariance function and with the further assumption that $\mu(\cdot)$ is twice continuously differentiable with continuous mixed second order partial derivatives, for $\mathbf{x} = (x_1, \dots, x_d)$,

$$g'_i(\mathbf{x}^*) = \frac{\partial g(\mathbf{x}^*)}{\partial x_i} = \frac{\partial g(\mathbf{x})}{\partial x_i} \Big|_{\mathbf{x}=\mathbf{x}^*},$$

corresponding to the original continuous-path Gaussian process $g(\cdot)$ exists for $i = 1, \dots, d$, for all $\mathbf{x}^* \in \mathcal{X}$. Specifically, for $i = 1, \dots, d$, $g'_i(\cdot) = \partial g(\cdot) / \partial x_i$ is a continuous-path Gaussian process with mean function $\mu'_i(\cdot) = \partial \mu(\cdot) / \partial x_i$ and covariance function $\sigma^2 \partial^2 c(\cdot, \cdot) / \partial x_i \partial y_i$.

For general details on Gaussian and Gaussian derivative processes, see, for example, Adler (1981), Adler and Taylor (2007).

2.5. Joint distribution of Gaussian variables and Gaussian derivative variables

Note that, given $\mathbf{x}^*, \mathbf{y}^* \in \mathcal{X}$,

$$Cov(g'_i(\mathbf{x}^*), g'_j(\mathbf{x}^*)) = \sigma^2 \frac{\partial^2 c(\mathbf{x}, \mathbf{y})}{\partial x_i \partial y_j} \Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{y}=\mathbf{x}^*}; \quad (1)$$

$$Cov(g'_i(\mathbf{x}^*), g(\mathbf{y}^*)) = \sigma^2 \frac{\partial c(\mathbf{x}, \mathbf{y})}{\partial x_i} \Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{y}=\mathbf{y}^*}. \quad (2)$$

With the above covariance forms, for given $\mathbf{x}^* \in \mathcal{X}$, we have

$$(g'_1(\mathbf{x}^*), \dots, g'_d(\mathbf{x}^*), g(\mathbf{x}_1), \dots, g(\mathbf{x}_n))^T \sim N_{d+n}(\boldsymbol{\nu}_{d+n}, \sigma^2 \boldsymbol{\Sigma}^{\overline{d+n \times d+n}}), \quad (3)$$

that is, the vector on the left hand side of (3) has the $(d+n)$ -variate normal distribution with mean vector $\boldsymbol{\nu}_{d+n}$ and covariance matrix $\sigma^2 \boldsymbol{\Sigma}^{\overline{d+n} \times \overline{d+n}}$. Here

$$\boldsymbol{\nu}_{d+n}(\mathbf{x}^*) = \left(\boldsymbol{\mu}'_d(\mathbf{x}^*)^T, \boldsymbol{\mu}_n^T \right)^T, \quad (4)$$

where $\boldsymbol{\mu}'_d(\mathbf{x}^*) = (\partial\mu(\mathbf{x}^*)/\partial x_1, \dots, \partial\mu(\mathbf{x}^*)/\partial x_d)^T$ with $\partial\mu(\mathbf{x}^*)/\partial x_i = \left. \frac{\partial\mu(\mathbf{x})}{\partial x_i} \right|_{\mathbf{x}=\mathbf{x}^*}$ for $i = 1, \dots, d$, and $\boldsymbol{\mu}_n = (\mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n))^T$. Also,

$$\boldsymbol{\Sigma}^{\overline{d+n} \times \overline{d+n}}(\mathbf{x}^*) = \begin{pmatrix} \boldsymbol{\Sigma}_{11}^{d \times d}(\mathbf{x}^*) & \boldsymbol{\Sigma}_{12}^{d \times n}(\mathbf{x}^*) \\ \boldsymbol{\Sigma}_{21}^{n \times d}(\mathbf{x}^*) & \boldsymbol{\Sigma}_{22}^{n \times n} \end{pmatrix}, \quad (5)$$

where $\boldsymbol{\Sigma}_{11}^{d \times d}$ is the d -th order correlation matrix with (i, j) -th element $\sigma^{-2} \text{Cov}(g'_i(\mathbf{x}^*), g'_j(\mathbf{x}^*))$ where the covariance term is given by (1), $\boldsymbol{\Sigma}_{12}^{d \times n}(\mathbf{x}^*)$ is the $d \times n$ matrix with (i, j) -th element $\sigma^{-2} \text{Cov}(g'_i(\mathbf{x}^*), g(\mathbf{x}_j))$ where the covariance term is of the same form as (2) with $\mathbf{y}^*$ replaced by $\mathbf{x}_j$, $\boldsymbol{\Sigma}_{21}^{n \times d}(\mathbf{x}^*)$ is the transpose of $\boldsymbol{\Sigma}_{12}^{d \times n}(\mathbf{x}^*)$, and $\boldsymbol{\Sigma}_{22}^{n \times n}$ is the $n \times n$ matrix with (i, j) -th element $c(\mathbf{x}_i, \mathbf{x}_j)$, the correlation between $g(\mathbf{x}_i)$ and $g(\mathbf{x}_j)$. In our examples, we shall consider the squared exponential correlation function having the form

$$c(\mathbf{x}, \mathbf{y}) = \exp \left(-\frac{1}{2}(\mathbf{x} - \mathbf{y})^T \boldsymbol{\Lambda}^{-1}(\mathbf{x} - \mathbf{y}) \right), \quad (6)$$

for $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, where $\boldsymbol{\Lambda}$ is a $d \times d$ diagonal matrix with positive diagonal elements λ_i ; $i = 1, \dots, d$. It follows that $\boldsymbol{\Sigma}_{11} = \boldsymbol{\Lambda}^{-1}$ and for $j = 1, \dots, n$, the j -th column of $\boldsymbol{\Sigma}_{12}(\mathbf{x}^*)$ is $-\boldsymbol{\Lambda}^{-1}(\mathbf{x}^* - \mathbf{x}_j)c(\mathbf{x}^*, \mathbf{x}_j)$.

2.6. Posterior distribution of the Gaussian derivatives given the data and parameters

From (3) it follows that the joint posterior distribution of $\mathbf{g}'(\mathbf{x}^*) = (g'_1(\mathbf{x}^*), \dots, g'_d(\mathbf{x}^*))^T$ given $\mathbf{x}^*$, $\mathbf{D}_n$, σ^2 and other parameters $\boldsymbol{\theta}$, is the following d -variate normal distribution:

$$\pi(\mathbf{g}'(\mathbf{x}^*) | \sigma^2, \boldsymbol{\theta}, \mathbf{D}_n) \equiv N_d(\tilde{\boldsymbol{\mu}}(\mathbf{x}^*), \sigma^2 \tilde{\boldsymbol{\Sigma}}(\mathbf{x}^*)), \quad (7)$$

where

$$\tilde{\boldsymbol{\mu}}(\mathbf{x}^*) = \boldsymbol{\mu}'_d(\mathbf{x}^*) + \boldsymbol{\Sigma}_{12}(\mathbf{x}^*) \boldsymbol{\Sigma}_{22}^{-1}(\mathbf{f}_n - \boldsymbol{\mu}_n); \quad (8)$$

$$\tilde{\boldsymbol{\Sigma}}(\mathbf{x}^*) = \boldsymbol{\Sigma}_{11}(\mathbf{x}^*) - \boldsymbol{\Sigma}_{12}(\mathbf{x}^*) \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21}(\mathbf{x}^*). \quad (9)$$

2.7. Prior and posterior distributions of the parameters

In our examples we assume that $\mu(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \boldsymbol{\beta}$, where $\mathbf{h}(\mathbf{x})^T = (1, x_1, \dots, x_d)$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_d)^T \in \mathbb{R}^d$. Let $\mathbf{H}^{n \times \overline{d+1}} = (\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_n))^T$.

2.7.1. Priors for the parameters

We assume that *a priori*

$$\pi(\boldsymbol{\beta}|\sigma^2) \equiv N_{d+1}(\boldsymbol{\beta}_0, \sigma^2 \boldsymbol{\Sigma}_0), \quad (10)$$

where $\boldsymbol{\beta}_0$ is the mean vector and $\boldsymbol{\Sigma}_0$ is the positive definite covariance matrix, and

$$\pi(\sigma^{-2}) \equiv \mathcal{G}(a, b), \quad (11)$$

the gamma distribution with mean a/b and variance a/b^2 , where $a, b > 0$.

2.7.2. Posteriors for the parameters

Let us first obtain the posterior distribution of σ^{-2} given $\mathbf{D}_n$. Note that

$$\begin{aligned} \pi(\sigma^{-2}|\mathbf{D}_n) &\propto \pi(\sigma^{-2}) \pi(\mathbf{g}_n|\sigma^{-2}) \\ &= \pi(\sigma^{-2}) \int \pi(\mathbf{g}_n|\sigma^{-2}, \boldsymbol{\beta}) \pi(\boldsymbol{\beta}|\sigma^2) d\boldsymbol{\beta}. \end{aligned} \quad (12)$$

To obtain $\pi(\mathbf{g}_n|\sigma^{-2}) = \int \pi(\mathbf{g}_n|\sigma^{-2}, \boldsymbol{\beta}) \pi(\boldsymbol{\beta}|\sigma^2) d\boldsymbol{\beta}$ note that

$$\pi(\mathbf{g}_n|\sigma^{-2}, \boldsymbol{\beta}) \equiv N_n(\mathbf{H}\boldsymbol{\beta}, \sigma^2 \boldsymbol{\Sigma}_{22}) \quad (13)$$

and since $\pi(\boldsymbol{\beta}|\sigma^2)$ has the normal distribution (10), it follows that

$$\pi(\mathbf{g}_n|\sigma^{-2}) \equiv N_n(\mathbf{H}\boldsymbol{\beta}_0, \sigma^2 (\mathbf{H}\boldsymbol{\Sigma}_0\mathbf{H}^T + \boldsymbol{\Sigma}_{22})). \quad (14)$$

Combining (14) with (12) and (11) it follows that

$$\pi(\sigma^{-2}|\mathbf{D}_n) \equiv \mathcal{G}\left(a + \frac{d}{2}, b + \frac{1}{2}(\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}_0)^T (\mathbf{H}\boldsymbol{\Sigma}_0\mathbf{H}^T + \boldsymbol{\Sigma}_{22})^{-1} (\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}_0)\right). \quad (15)$$

Also, combining (13) and (10) it is easy to see that

$$\pi(\boldsymbol{\beta}|\mathbf{D}_n, \sigma^2) \equiv N_{d+1}\left(\left(\mathbf{H}^T \boldsymbol{\Sigma}_{22}^{-1} \mathbf{H} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1} \left(\mathbf{H}^T \boldsymbol{\Sigma}_{22}^{-1} \mathbf{f}_n + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\beta}_0\right), \sigma^2 \left(\mathbf{H}^T \boldsymbol{\Sigma}_{22}^{-1} \mathbf{H} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1}\right). \quad (16)$$

2.8. Marginal posterior distribution of the derivative process

Now, from (7) it follows that

$$\pi(\mathbf{g}'(\mathbf{x}^*)|\sigma^2, \boldsymbol{\beta}, \mathbf{D}_n) \equiv N_d(\mathbf{A}\boldsymbol{\beta} + \boldsymbol{\Sigma}_{12}(\mathbf{x}^*)\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}), \sigma^2 \tilde{\boldsymbol{\Sigma}}(\mathbf{x}^*)), \quad (17)$$

where $\mathbf{A}^{d \times \overline{d+1}} = (\mathbf{0}^{d \times 1} \quad \mathbb{I}_d)$. Here $\mathbf{0}^{d \times 1}$ is the d -dimensional null vector and $\mathbb{I}_d$ is the identity matrix of order d . Integrating (17) with respect to (16) we obtain

$$\pi(\mathbf{g}'(\mathbf{x}^*)|\sigma^2, \mathbf{D}_n) \equiv N_d(\hat{\boldsymbol{\mu}}'(\mathbf{x}^*), \sigma^2 \hat{\boldsymbol{\Sigma}}(\mathbf{x}^*)), \quad (18)$$

where $\hat{\boldsymbol{\mu}}'(\mathbf{x}^*)$ and $\hat{\boldsymbol{\Sigma}}(\mathbf{x}^*)$ are given by

$$\hat{\boldsymbol{\mu}}'(\mathbf{x}^*) = \mathbf{A}\hat{\boldsymbol{\beta}} + \boldsymbol{\Sigma}_{12}(\mathbf{x}^*)\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{f}_n - \mathbf{H}\hat{\boldsymbol{\beta}}); \quad (19)$$

$$\hat{\boldsymbol{\Sigma}}(\mathbf{x}^*) = \tilde{\boldsymbol{\Sigma}}(\mathbf{x}^*) + \left(\mathbf{A} - \boldsymbol{\Sigma}_{12}(\mathbf{x}^*)\boldsymbol{\Sigma}_{22}^{-1}\mathbf{H}\right) \left(\mathbf{H}^T\boldsymbol{\Sigma}_{22}^{-1}\mathbf{H} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1} \left(\mathbf{A} - \boldsymbol{\Sigma}_{12}(\mathbf{x}^*)\boldsymbol{\Sigma}_{22}^{-1}\mathbf{H}\right)^T, \quad (20)$$

with

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}^T\boldsymbol{\Sigma}_{22}^{-1}\mathbf{H} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1} \left(\mathbf{H}^T\boldsymbol{\Sigma}_{22}^{-1}\mathbf{f}_n + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\beta}_0\right). \quad (21)$$

Integrating (18) with respect to (15) we obtain

$$\pi(\mathbf{g}'(\mathbf{x}^*)|\mathbf{D}_n) \equiv t_d\left(\hat{\boldsymbol{\mu}}'(\mathbf{x}^*), \frac{\left(a + \frac{d}{2}\right)\hat{\boldsymbol{\Sigma}}(\mathbf{x}^*)^{-1}}{b + \frac{1}{2}(\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}_0)^T(\mathbf{H}\boldsymbol{\Sigma}_0\mathbf{H}^T + \boldsymbol{\Sigma}_{22})^{-1}(\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}_0)}, 2\left(a + \frac{d}{2}\right)\right), \quad (22)$$

where for any d -dimensional vector $\boldsymbol{\mu}$, d -th order covariance matrix $\boldsymbol{\Sigma}$, and $\alpha > 0$, $t_d(\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1}, \alpha)$ is a d -variate Student's t distribution with density at $\mathbf{x} \in \mathbb{R}^d$ given by

$$t_d(\mathbf{x} : \boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1}, \alpha) = C \left[1 + \alpha^{-1}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right]^{-\frac{(\alpha+d)}{2}},$$

where

$$C = \frac{\Gamma\left(\frac{\alpha+d}{2}\right)}{\Gamma\left(\frac{\alpha}{2}\right)(\alpha\pi)^{\frac{d}{2}}},$$

with $\Gamma(\cdot)$ denoting the gamma function.

3. Posterior distribution of random optima corresponding to the posterior derivative process

From (22) we obtain the following posterior density at $\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}$:

$$\pi(\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}|\mathbf{D}_n) \propto \left[1 + \frac{\hat{\boldsymbol{\mu}}'(\mathbf{x}^*)^T \hat{\boldsymbol{\Sigma}}(\mathbf{x}^*)^{-1} \hat{\boldsymbol{\mu}}'(\mathbf{x}^*)}{2b + (\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}_0)^T(\mathbf{H}\boldsymbol{\Sigma}_0\mathbf{H}^T + \boldsymbol{\Sigma}_{22})^{-1}(\mathbf{f}_n - \mathbf{H}\boldsymbol{\beta}_0)}\right]^{-(a+d)}. \quad (23)$$

Now, with prior $\pi(\mathbf{x}^*)$ on $\mathbf{x}^*$, the posterior of $\mathbf{x}^*$, given $\mathbf{g}'(\mathbf{x}^*)$ and $\mathbf{D}_n$ can be obtained as follows:

$$\begin{aligned} \pi(\mathbf{x}^*|\mathbf{g}'(\mathbf{x}^*), \mathbf{D}_n) &\propto \pi(\mathbf{x}^*)\pi(\mathbf{g}'(\mathbf{x}^*), \mathbf{g}_n|\mathbf{x}^*) \\ &= \pi(\mathbf{x}^*)\pi(\mathbf{g}'(\mathbf{x}^*)|\mathbf{D}_n, \mathbf{x}^*)\pi(\mathbf{g}_n|\mathbf{x}^*) \\ &= \pi(\mathbf{x}^*)\pi(\mathbf{g}'(\mathbf{x}^*)|\mathbf{D}_n, \mathbf{x}^*)\pi(\mathbf{g}_n) \\ &\propto \pi(\mathbf{x}^*)\pi(\mathbf{g}'(\mathbf{x}^*)|\mathbf{D}_n, \mathbf{x}^*). \end{aligned} \quad (24)$$

In the second step of (24), $\pi(\mathbf{g}_n|\mathbf{x}^*)$ is the marginal distribution of $\mathbf{g}_n$, integrated over the parameters. Since this does not depend upon $\mathbf{x}^*$, we denoted this as $\pi(\mathbf{g}_n)$ in the third step of (24). From (24) it then follows that

$$\pi(\mathbf{x}^*|\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_n) \propto \pi(\mathbf{x}^*)\pi(\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}|\mathbf{D}_n, \mathbf{x}^*). \quad (25)$$

Now, $\pi(\mathbf{g}'(\mathbf{x}^*) = \mathbf{0} | \mathbf{D}_n, \mathbf{x}^*)$ will also depend upon parameters of the covariance function, which will be unknown generally. It is legitimate to estimate them using the maximum likelihood estimation method (see, for example, Santner *et al.* (2003)) and treat them as fixed.

The formula (25) holds for any prior $\pi(\mathbf{x}^*)$ for $\mathbf{x}^*$. However, we shall consider a uniform prior on $\mathcal{X}$ constrained by the first and second derivatives of $f(\cdot)$. The details are presented below.

3.1. Prior for $\mathbf{x}^*$

Without loss of generality, let us assume that our objective is to obtain the minima of the function $f(\cdot)$ on $\mathcal{X}$. For $i, j = 1, \dots, d$, let $f''_{ij}(\mathbf{x}^*) = \left. \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \right|_{\mathbf{x}=\mathbf{x}^*}$ denote the second order partial derivatives of the objective function $f(\cdot)$ at any $\mathbf{x}^* \in \mathcal{X}$. Let $\Sigma''(\mathbf{x}^*)$ stand for the $d \times d$ matrix of such second order partial derivatives at $\mathbf{x}^*$ with (i, j) -th element $f''_{ij}(\mathbf{x}^*)$. Let $\Sigma''(\mathbf{x}^*) > 0$ denote that $\Sigma''(\mathbf{x}^*)$ is positive definite. Then we consider the following prior for $\mathbf{x}^*$:

$$\pi(\mathbf{x}^*) \propto I_{B(\epsilon)}(\mathbf{x}^*), \quad (26)$$

where, for any set A and vector $\mathbf{x}$, $I_A(\mathbf{x}) = 1$ if $\mathbf{x} \in A$ and zero otherwise. Also, for any $\mathbf{x}$ and $\epsilon > 0$,

$$B(\epsilon) = \mathcal{X} \cap \{\mathbf{x} : \|\mathbf{f}'(\mathbf{x})\|_d < \epsilon\} \cap \{\mathbf{x} : \Sigma''(\mathbf{x}) > 0\}, \quad (27)$$

where $\|\cdot\|_d$ denotes the Euclidean norm in the d -dimensional Euclidean space.

3.2. Interpretation of the posterior for $\mathbf{x}^*$ as a pseudo-posterior

Before proceeding to the convergence theory, we comment on the nature of the target in (25). The event $\{\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}\}$ has probability zero under the continuous distribution of the derivative process. Therefore, the expression $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_n)$ is not a conditional probability in the strict Kolmogorov sense. Instead, it should be understood as a density-based target obtained by evaluating the conditional density of the derivative at zero, which serves as a surrogate posterior for the stationary point. This construction is akin to a Gibbs posterior or an energy-based formulation; it is well-defined as a density and leads to a valid sampling scheme. Such pseudo-posteriors have been successfully used in various contexts (see, for example, Bissiri *et al.* (2016)) and we adopt a similar perspective here.

4. Almost sure uniform convergence of posterior Gaussian and Gaussian derivative processes

Consider the joint posterior distribution of

$$\pi(g(\cdot), \mathbf{g}'(\cdot) | \mathbf{D}_n) = \pi(g(\cdot) | \mathbf{D}_n) \pi(\mathbf{g}'(\cdot) | g(\cdot), \mathbf{D}_n). \quad (28)$$

Then the marginal posterior $\pi(\mathbf{g}'(\cdot) | \mathbf{D}_n)$ is of the same form as (22), and the marginal posterior distribution $\pi(g(\cdot) | \mathbf{D}_n)$ in (28) corresponds to the t_1 process, the form of which is not relevant for our purpose.

For $n \geq 1$, let $\mathbf{X}_n$ denote the n input points in $\mathbf{D}_n$. Note that even after marginalizing out the parameters of the Gaussian process with respect to their posteriors (here $\boldsymbol{\beta}$ and σ^2),

the interpolation property of $g(\cdot)$ given $\mathbf{D}_n$ is preserved. That is, the marginal posterior $\pi(g(\mathbf{x}^*)|\mathbf{D}_n)$ gives full posterior mass to $f(\mathbf{x}^*)$ if $\mathbf{x}^* \in \mathbf{X}_n$.

Let $g_n(\cdot)$ denote any random function associated with any non-null set of the marginalized posterior measure of $g(\cdot)$ given $\mathbf{D}_n$ (here, the t_1 posterior measure). Also, let $\mathbf{g}'_n(\cdot)$ denote any d -dimensional random function associated with any non-null set of the marginalized posterior measure of $\mathbf{g}'(\cdot)$ given $\mathbf{D}_n$, the form of which is provided explicitly by (22). Theorems 1, 2, 3 and 4 prove almost sure uniform convergence of $g_n(\cdot)$ and $\mathbf{g}'_n(\cdot)$ to $f(\cdot)$ and $\mathbf{f}'(\cdot)$ respectively, as $n \rightarrow \infty$. In particular, Theorems 3 and 4 also provide rates of such convergences.

Before presenting the theorems, we state a set of standing assumptions that will be used throughout this section.

Assumption 4.1 (Assumptions for convergence results): We assume the following:

- (A1) The domain $\mathcal{X} \subset \mathbb{R}^d$ is compact and convex.
- (A2) The objective function f is twice continuously differentiable on an open set containing $\mathcal{X}$, with continuous mixed second-order partial derivatives.
- (A3) The mean function $\mu(\cdot)$ of the Gaussian process prior is twice continuously differentiable with continuous mixed second-order partial derivatives.
- (A4) The correlation function $c(\cdot, \cdot)$ is such that the mixed partial derivatives $\frac{\partial^2 c(\mathbf{x}, \mathbf{y})}{\partial x_i \partial y_i}$ exist and are Lipschitz continuous on $\mathcal{X} \times \mathcal{X}$ for all $i = 1, \dots, d$. For the higher-order convergence results (Theorems 3 and 4), we additionally require the existence and Lipschitz continuity of $\frac{\partial^4 c(\mathbf{x}, \mathbf{y})}{\partial x_i \partial y_i \partial x_j \partial y_j}$ for all i, j .
- (A5) The input points $\mathbf{X}_n$ satisfy a fixed-domain infill asymptotics condition: for any $\mathbf{x} \in \mathcal{X}$, there exists a sequence $\mathbf{x}_n \in \mathbf{X}_n$ such that

$$\|\mathbf{x}_n - \mathbf{x}\|_d \rightarrow 0 \text{ as } n \rightarrow \infty, \quad (29)$$

and moreover points of the form $\mathbf{x}_n + \mathbf{h}_n$ with $\mathbf{h}_n \rightarrow \mathbf{0}$ also belong to $\mathbf{X}_n$.

- (A6) The correlation matrices Σ_{22} and their derivatives are uniformly non-degenerate: there exists a constant $c_0 > 0$ such that the smallest eigenvalue of Σ_{22} is bounded below by c_0 for all n .

Assumption (A6) is used in two essential places. First, the posterior distributions derived in Sections 2.6 and 2.8 explicitly depend on Σ_{22}^{-1} ; the uniform lower bound on the smallest eigenvalue guarantees that Σ_{22} is invertible for every n and that its inverse remains numerically stable as n increases. Second, in the proofs of Theorems 2 and 3 we require a uniform bound on the second derivatives $g''_{ijn}(\cdot)$ over n . Such a bound follows from the uniform boundedness of the conditional covariance $\tilde{\Sigma}(\mathbf{x}^*)$ in (9) and the stability of the posterior of $\mathbf{g}'(\mathbf{x}^*)$; both rely on Σ_{22}^{-1} not becoming arbitrarily large. Consequently, (A6) ensures that the almost sure uniform convergence results hold without pathological degeneracy of the design points.

These assumptions are standard in the literature on fixed-domain asymptotics for Gaussian processes (see, for example, Stein (1999)) and are sufficient for the proofs that follow.

Lemma 1: Consider a sequence of real-valued continuous functions $\{f_n\}_{n=1}^\infty$ on any compact set $\mathcal{X}$ such that $f_n(\mathbf{x}) \rightarrow f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, where f is some real-valued continuous function on $\mathcal{X}$. Then

$$\limsup_{n \rightarrow \infty} \sup_{\mathbf{x} \in \mathcal{X}} |f_n(\mathbf{x})| < \infty.$$

Proof: Note that f_n , for $n \geq 1$, and f are actually uniformly continuous since $\mathcal{X}$ is compact. Now let us first consider an arbitrary $\mathbf{x}_1 \in \mathcal{X}$. Then due to pointwise convergence of f_n to f , for any $\epsilon > 0$, there exists $n_1 \geq 1$ such that for $n \geq n_1$, $|f_n(\mathbf{x}_1) - f(\mathbf{x}_1)| < \epsilon$. Moreover, due to uniform continuity of f_n and f , there exists an open neighborhood $\mathcal{N}(\mathbf{x}_1)$ of $\mathbf{x}_1$ such that $|f_n(\mathbf{x}) - f(\mathbf{x})| < \epsilon_1$ for all $\mathbf{x} \in \mathcal{N}(\mathbf{x}_1)$, where ϵ_1 is some positive finite constant. Since f is continuous on the compact set $\mathcal{X}$, it is uniformly bounded. Hence, $\sup_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_1)} |f_n(\mathbf{x})| < M_1$

for all $n \geq n_1$, where M_1 is some positive finite constant.

Now consider another point $\mathbf{x}_2 \in \mathcal{X} \setminus \mathcal{N}(\mathbf{x}_1)$. Then similar argument shows that $\sup_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_2)} |f_n(\mathbf{x})| < M_2$ for all $n \geq n_2 \geq n_1$, where $\mathcal{N}(\mathbf{x}_2)$ is some appropriate open neighborhood of $\mathbf{x}_2$ and M_2 is some positive finite constant.

Thus, starting with $\mathcal{N}(\mathbf{x}_1)$ and the associated bound $\sup_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_1)} |f_n(\mathbf{x})| < M_1$, continuing the procedure for $i \geq 2$, we can construct neighborhoods $\mathcal{N}(\mathbf{x}_i)$ with $\mathbf{x}_i \in \mathcal{X} \setminus \bigcup_{j=1}^{i-1} \mathcal{N}(\mathbf{x}_j)$ and bounds M_i such that for all $n \geq n_i \geq n_{i-1} \geq \dots \geq n_2 \geq n_1$, $\sup_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i)} |f_n(\mathbf{x})| < M_i$. Note that $\mathcal{X} \subseteq \bigcup_{i=1}^\infty \mathcal{N}(\mathbf{x}_i)$. That is, the set of neighborhoods $\{\mathcal{N}(\mathbf{x}_i) : i = 1, 2, \dots\}$ constitutes an open cover for $\mathcal{X}$. Since $\mathcal{X}$ is compact, there exists a finite sub-cover for $\mathcal{X}$, say, $\{\mathcal{N}(\mathbf{x}_{i_j}) : j = 1, 2, \dots, K\}$, where K is finite. Now, by our construction, for $n \geq n_{i_j}$, $\sup_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_{i_j})} |f_n(\mathbf{x})| < M_{i_j}$, for $j = 1, \dots, K$. Let $n_0 = \max\{n_{i_j} : j = 1, \dots, K\}$ and $M = \max\{M_{i_j} : j = 1, \dots, K\}$. Then for all $n \geq n_0$, $\sup_{\mathbf{x} \in \mathcal{X}} |f_n(\mathbf{x})| < M < \infty$. $\square$

Theorem 1: Consider a fixed-domain infill asymptotics framework satisfying Assumption (A5). Under Assumptions (A1)–(A4) and (A6), for almost all sequences $\{g_n(\cdot)\}_{n=1}^\infty$,

$$\sup_{\mathbf{x} \in \mathcal{X}} |g_n(\mathbf{x}) - f(\mathbf{x})| \rightarrow 0, \text{ as } n \rightarrow \infty. \quad (30)$$

Proof: Consider any $\mathbf{x} \in \mathcal{X}$. Then there exists $\mathbf{x}_n \in \mathbf{X}_n$ for $n \geq 1$ satisfying (29). Now by Taylor's series expansion up to the first order,

$$g_n(\mathbf{x}_n) = g_n(\mathbf{x}) + (\mathbf{x}_n - \mathbf{x})^T \mathbf{g}'_n(\mathbf{c}_n), \quad (31)$$

where $\mathbf{c}_n$ lies on the line joining $\mathbf{x}$ and $\mathbf{x}_n$.

Now, for $i = 1, \dots, d$, consider the i -th partial derivative $g'_{in}(\cdot)$ of $g_n(\cdot)$. With any sequence $h_{in} \rightarrow 0$ as $n \rightarrow \infty$, we have

$$\frac{g_n(x_{1n}, \dots, x_{i-1,n}, x_{in} + h_{in}, x_{i+1,n}, \dots, x_{dn}) - g_n(\mathbf{x}_n)}{h_{in}} = g'_{in}(\mathbf{x}_n^*), \quad (32)$$

where $\mathbf{x}_n^* = (x_{1n}, \dots, x_{i-1,n}, x_{in}^*, x_{i+1,n}, \dots, x_{dn})$; here x_{in}^* lies between x_{in} and $x_{in} + h_{in}$. Since $(x_{1n}, \dots, x_{i-1,n}, x_{in} + h_{in}, x_{i+1,n}, \dots, x_{dn})^T \in \mathbf{X}_n$ and $\mathbf{x}_n \in \mathbf{X}_n$, $g_n(x_{1n}, \dots, x_{i-1,n}, x_{in} + h_{in}, x_{i+1,n}, \dots, x_{dn}) = f(x_{1n}, \dots, x_{i-1,n}, x_{in} + h_{in}, x_{i+1,n}, \dots, x_{dn})$ and $g_n(\mathbf{x}_n) = f(\mathbf{x}_n)$, almost surely. Hence, from (32) it follows that

$$g'_{in}(\mathbf{x}_n^*) = f'_i(\mathbf{z}_n), \text{ almost surely,} \quad (33)$$

with $\mathbf{z}_n = (x_{1n}, \dots, x_{i-1,n}, z_{in}, x_{i+1,n}, \dots, x_{dn})$, where z_{in} lies between x_{in} and $x_{in} + h_{in}$. Clearly, $\mathbf{z}_n \rightarrow \mathbf{x}$, as $n \rightarrow \infty$. Hence, taking limits of both sides of (33) as $n \rightarrow \infty$, and using continuity of $f'_i(\cdot)$, yields

$$\lim_{n \rightarrow \infty} g'_{in}(\mathbf{x}_n^*) = f'_i(\mathbf{x}), \text{ almost surely.} \quad (34)$$

Now, by the hypothesis of Lipschitz continuity of the second order mixed partial derivatives of the correlation function ensures existence and sample path continuity of the partial derivatives $g'_{in}(\cdot)$, for $i = 1, \dots, d$, for any $n \geq 1$. Since $\mathcal{X}$ is compact, $g'_{in}(\cdot)$ are uniformly continuous on $\mathcal{X}$, for $i = 1, \dots, d$, for any $n \geq 1$. Uniform continuity of $g'_{in}(\cdot)$ for all $n \geq 1$ implies that for any $\epsilon > 0$, $|g'_{in}(\mathbf{x}_n^*) - g'_{in}(\mathbf{x})| < \epsilon$, whenever $\|\mathbf{x}_n^* - \mathbf{x}\|_d < \delta$, where $\delta (> 0)$ depends upon ϵ only. Now, since $\mathbf{x}_n^* \rightarrow \mathbf{x}$ as $n \rightarrow \infty$, there exists $n_0 (\geq 1)$ depending upon δ such that $\|\mathbf{x}_n^* - \mathbf{x}\|_d < \delta$ for $n \geq n_0$. Further, using (34) we obtain for any $\mathbf{x} \in \mathcal{X}$, the following:

$$\lim_{n \rightarrow \infty} g'_{in}(\mathbf{x}) = \lim_{n \rightarrow \infty} g'_{in}(\mathbf{x}_n^*) + \lim_{n \rightarrow \infty} (g'_{in}(\mathbf{x}) - g'_{in}(\mathbf{x}_n^*)) = f'_i(\mathbf{x}), \text{ almost surely.} \quad (35)$$

That is, $g'_{in}(\cdot)$ converges pointwise to $f'_i(\cdot)$ almost surely, as $n \rightarrow \infty$. Moreover, $g'_{in}(\cdot)$ is almost surely continuous on $\mathcal{X}$ for all $n \geq 1$ and $f'_i(\cdot)$ is continuous on $\mathcal{X}$. Since $\mathcal{X}$ is compact, we invoke Lemma 1 to conclude that there exists a positive, finite constant M depending upon $f'_i(\cdot)$; $i = 1, \dots, d$ such that

$$\lim_{n \rightarrow \infty} \sup_{\mathbf{x} \in \mathcal{X}} |g'_{in}(\mathbf{x})| < M, \text{ almost surely, for } i = 1, \dots, d. \quad (36)$$

Hence, using the Cauchy-Schwartz inequality in (31), boundedness of the partial derivatives $g'_{in}(\cdot)$ for $i = 1, \dots, d$ for large enough n and (29), we obtain

$$|g_n(\mathbf{x}_n) - g_n(\mathbf{x})| = |(\mathbf{x}_n - \mathbf{x})^T \mathbf{g}'_n(\mathbf{c}_n)| \leq \|\mathbf{x}_n - \mathbf{x}\|_d \times \|\mathbf{g}'_n(\mathbf{c}_n)\|_d \rightarrow 0, \text{ as } n \rightarrow \infty. \quad (37)$$

Hence, using (37) and continuity of $f(\cdot)$ we obtain, for any $\mathbf{x} \in \mathcal{X}$,

$$\lim_{n \rightarrow \infty} g_n(\mathbf{x}) = \lim_{n \rightarrow \infty} g_n(\mathbf{x}_n) = \lim_{n \rightarrow \infty} f(\mathbf{x}_n) = f(\lim_{n \rightarrow \infty} \mathbf{x}_n) = f(\mathbf{x}), \quad (38)$$

proving pointwise convergence of $g_n(\cdot)$ to $f(\cdot)$.

Thus, we have shown that $g_n(\cdot)$ converges pointwise to $f(\cdot)$ on $\mathcal{X}$ almost surely, as $n \rightarrow \infty$ (equation (38)), and also that the partial derivatives of $g_n(\cdot)$ are uniformly bounded in the limit almost surely (equation (36)). The latter also implies that $g_n(\cdot)$ is almost surely Lipschitz continuous on $\mathcal{X}$. Since $\mathcal{X}$ is compact, by the stochastic Arzelà–Ascoli lemma (see, for example, Theorem 1.6.6 of van der Vaart and Wellner (1996)), which states that if a sequence of random continuous functions is pointwise convergent and has uniformly bounded derivatives (hence equicontinuous and tight), then uniform convergence follows, we conclude that (30) holds. $\square$

Theorem 2: Consider the same setup as in Theorem 1 with the additional smoothness condition on the correlation function (fourth-order mixed derivatives Lipschitz). Then, for almost all sequences $\{\mathbf{g}'_n(\cdot)\}_{n=1}^\infty$,

$$\sup_{\mathbf{x} \in \mathcal{X}} \|\mathbf{g}'_n(\mathbf{x}) - \mathbf{f}'(\mathbf{x})\|_d \rightarrow 0, \text{ as } n \rightarrow \infty. \quad (39)$$

Proof: Note that for $i = 1, \dots, d$, pointwise convergence of $g'_{in}(\cdot)$ to $f'_i(\cdot)$ as $n \rightarrow \infty$, is already shown by (35), in the proof of Theorem 1. Hence, if we can show that for $i, j = 1, \dots, d$, the absolute values of the second order partial derivatives $|g''_{ijn}(\cdot)|$ are uniformly bounded on $\mathcal{X}$ as $n \rightarrow \infty$, then this would imply that $g'_{in}(\cdot)$ are almost surely Lipschitz continuous on $\mathcal{X}$ for large enough n . Since $\mathcal{X}$ is compact, this would then imply by the stochastic Arzelà–Ascoli lemma (Theorem 1.6.6 of van der Vaart and Wellner (1996)) that $\sup_{\mathbf{x} \in \mathcal{X}} |g'_{in}(\mathbf{x}) - f'_i(\mathbf{x})| \rightarrow 0$, almost surely, as $n \rightarrow \infty$, for $i = 1, \dots, d$, which is equivalent to (39). Hence, in the rest of the proof, we show that $\limsup_{n \rightarrow \infty} \sup_{\mathbf{x} \in \mathcal{X}} |g''_{ijn}(\mathbf{x})| < \infty$. The argument is given for $i = j$; mixed partials follow similarly.

Fix $\mathbf{x} \in \mathcal{X}$. By the infill design condition (A5) there exists a sequence $\mathbf{x}_n \in X_n$ (the set of design points) such that $\mathbf{x}_n \rightarrow \mathbf{x}$. Choose a scalar $h_n > 0$ with $h_n \rightarrow 0$ and such that $\mathbf{x}_n \pm h_n \mathbf{e}_i \in X_n$, where $\mathbf{e}_i$ is the i -th standard basis vector. This is possible because (A5) requires points of the form $\mathbf{x}_n + \mathbf{h}_n$ with $\mathbf{h}_n \rightarrow \mathbf{0}$ to belong to X_n .

Since f is twice continuously differentiable (Assumption A2), Taylor's theorem with Lagrange remainder gives

$$f(\mathbf{x}_n + h_n \mathbf{e}_i) = f(\mathbf{x}_n) + f'_i(\mathbf{x}_n)h_n + \frac{1}{2}f''_{ii}(\mathbf{x}_n + \theta_{1n}h_n \mathbf{e}_i)h_n^2, \quad (40)$$

$$f(\mathbf{x}_n - h_n \mathbf{e}_i) = f(\mathbf{x}_n) - f'_i(\mathbf{x}_n)h_n + \frac{1}{2}f''_{ii}(\mathbf{x}_n - \theta_{2n}h_n \mathbf{e}_i)h_n^2, \quad (41)$$

with $\theta_{1n}, \theta_{2n} \in (0, 1)$. Adding the two equalities and subtracting $2f(\mathbf{x}_n)$ yields

$$\frac{f(\mathbf{x}_n + h_n \mathbf{e}_i) - 2f(\mathbf{x}_n) + f(\mathbf{x}_n - h_n \mathbf{e}_i)}{h_n^2} = \frac{1}{2} [f''_{ii}(\mathbf{x}_n + \theta_{1n}h_n \mathbf{e}_i) + f''_{ii}(\mathbf{x}_n - \theta_{2n}h_n \mathbf{e}_i)].$$

Because f''_{ii} is Lipschitz continuous on the compact set $\mathcal{X}$ (Assumption A4), there exists a constant $L_f < \infty$ such that $|f''_{ii}(\mathbf{y}) - f''_{ii}(\mathbf{z})| \leq L_f \|\mathbf{y} - \mathbf{z}\|_d$ for all $\mathbf{y}, \mathbf{z} \in \mathcal{X}$. Consequently,

$$\left| \frac{1}{2} [f''_{ii}(\mathbf{x}_n + \theta_{1n}h_n \mathbf{e}_i) + f''_{ii}(\mathbf{x}_n - \theta_{2n}h_n \mathbf{e}_i)] - f''_{ii}(\mathbf{x}) \right| \leq \frac{L_f}{2} (\|\mathbf{x}_n + \theta_{1n}h_n \mathbf{e}_i - \mathbf{x}\|_d + \|\mathbf{x}_n - \theta_{2n}h_n \mathbf{e}_i - \mathbf{x}\|_d).$$

Since $\mathbf{x}_n \rightarrow \mathbf{x}$ and $h_n \rightarrow 0$, the right-hand side is bounded by $C_f h_n$ for some constant C_f (depending only on L_f and the rate of convergence of $\mathbf{x}_n$). Thus we obtain the remainder bound

$$\left| \frac{f(\mathbf{x}_n + h_n \mathbf{e}_i) - 2f(\mathbf{x}_n) + f(\mathbf{x}_n - h_n \mathbf{e}_i)}{h_n^2} - f''_{ii}(\mathbf{x}) \right| \leq C_f h_n. \quad (42)$$

The random function g_n is almost surely twice continuously differentiable because the covariance kernel satisfies the Lipschitz condition (A4) and the mean function is twice

differentiable (A3). Moreover, the interpolation property of the posterior gives $g_n(\mathbf{x}_n \pm h_n \mathbf{e}_i) = f(\mathbf{x}_n \pm h_n \mathbf{e}_i)$ and $g_n(\mathbf{x}_n) = f(\mathbf{x}_n)$ almost surely, since these points belong to X_n . Repeating the Taylor expansion for g_n yields

$$\frac{g_n(\mathbf{x}_n + h_n \mathbf{e}_i) - 2g_n(\mathbf{x}_n) + g_n(\mathbf{x}_n - h_n \mathbf{e}_i)}{h_n^2} = \frac{1}{2} [g''_{iin}(\mathbf{x}_n + \tilde{\theta}_{1n} h_n \mathbf{e}_i) + g''_{iin}(\mathbf{x}_n - \tilde{\theta}_{2n} h_n \mathbf{e}_i)],$$

with $\tilde{\theta}_{1n}, \tilde{\theta}_{2n} \in (0, 1)$. From the proof of uniform boundedness of second derivatives (see the discussion preceding this theorem) we know that $\sup_n \sup_{\mathbf{y} \in \mathcal{X}} |g''_{iin}(\mathbf{y})| < \infty$ almost surely, and the sample paths are equicontinuous. Hence there exists a constant C_g (independent of n) such that

$$\left| \frac{1}{2} [g''_{iin}(\mathbf{x}_n + \tilde{\theta}_{1n} h_n \mathbf{e}_i) + g''_{iin}(\mathbf{x}_n - \tilde{\theta}_{2n} h_n \mathbf{e}_i)] - g''_{iin}(\mathbf{x}) \right| \leq C_g h_n. \quad (43)$$

The left-hand sides of (42) and (43) are exactly equal because they involve the same function values f at the three design points. Therefore we have

$$g''_{iin}(\mathbf{x}) + \tilde{R}_n = f''_{ii}(\mathbf{x}) + R_n,$$

where $|R_n| \leq C_f h_n$ and $|\tilde{R}_n| \leq C_g h_n$. Rearranging gives

$$|g''_{iin}(\mathbf{x}) - f''_{ii}(\mathbf{x})| = |\tilde{R}_n - R_n| \leq (C_f + C_g) h_n.$$

Since $h_n \rightarrow 0$, we conclude

$$\lim_{n \rightarrow \infty} g''_{iin}(\mathbf{x}) = f''_{ii}(\mathbf{x}) \quad \text{almost surely.} \quad (44)$$

For $i \neq j$, a similar argument using second-order mixed differences (for example, with $\mathbf{h}_{ijn} = h_n \mathbf{e}_i + h_n \mathbf{e}_j$) or the polarization identity yields

$$\lim_{n \rightarrow \infty} g''_{ijn}(\mathbf{x}) = f''_{ij}(\mathbf{x}) \quad \text{almost surely.} \quad (45)$$

The remainder bounds again rely on the Lipschitz continuity of f'_{ij} and the uniform boundedness of g''_{ijn} .

Since $g''_{ijn}(\cdot)$ is almost surely continuous for all $n \geq 1$ and $f''_{ij}(\cdot)$ is continuous, with $\mathcal{X}$ being compact, the pointwise convergence results (44) and (45) lets us conclude, using Lemma 1, that $\limsup_{n \rightarrow \infty} \sup_{\mathbf{x} \in \mathcal{X}} |g''_{ijn}(\mathbf{x})| < \infty$.

□

Remark 1: The proof above never divides by $f'_{ii}(\mathbf{x})$, so it remains valid when $f'_{ii}(\mathbf{x}) = 0$. In that case the bound $|g''_{iin}(\mathbf{x})| \leq (C_f + C_g) h_n$ directly forces convergence to zero. Hence no special treatment is required.

Theorems 1 and 2 prove almost sure uniform convergence of $g_n(\cdot)$ and $\mathbf{g}'_n(\cdot)$ to $f(\cdot)$ and $\mathbf{f}'(\cdot)$, respectively. However, the rates of convergence are not provided by these theorems. Further fine-tuning the structure of the set of input points $\mathbf{X}_n$ helps achieve desired rates of convergence, as we show next in Theorems 3 and 4.

Theorem 3: Let $\mathcal{X} = \prod_{i=1}^d \mathcal{X}_i$, where, for $i = 1, \dots, d$, $\mathcal{X}_i$ are compact subsets of $\mathbb{R}$. For each $i \in \{1, \dots, d\}$, let $x_{1i} < x_{2i} < \dots < x_{\tilde{n}_i i}$ be an ordered set of points partitioning $\mathcal{X}_i$, with $h_i = \max_{1 \leq j \leq \tilde{n}_i - 1} (x_{j+1i} - x_{ji})$. For $i = 1, \dots, d$, and for $j = 1, \dots, \tilde{n}_i$, let input points of the form $(x_1^*, \dots, x_{i-1}^*, x_{ji}, x_{i+1}^*, \dots, x_d^*)$ belong to $\mathbf{X}_n$, where $(x_1^*, \dots, x_{i-1}^*, x_{i+1}^*, \dots, x_d^*) \in \prod_{j \neq i} \mathcal{X}_j$ may be arbitrary.

For $\mathbf{x} = (x_1, \dots, x_d)$ and $\mathbf{y} = (y_1, \dots, y_d)$, let the correlation function be such that

$$\frac{\partial^4 c(\mathbf{x}^*, \mathbf{y}^*)}{\partial x_i \partial y_i \partial x_j \partial y_j} = \frac{\partial^4 c(\mathbf{x}, \mathbf{y})}{\partial x_i \partial y_i \partial x_j \partial y_j} \Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{y}=\mathbf{y}^*}$$

exists for all $\mathbf{x}^*, \mathbf{y}^* \in \mathcal{X}$ and is Lipschitz continuous on $\mathcal{X} \times \mathcal{X}$ for $i, j = 1, \dots, d$.

Then, letting $h = \max_{1 \leq i \leq d} h_i$, the following holds for almost all sequences $\{\mathbf{g}'_n(\cdot)\}_{n=1}^\infty$ with the above forms of the input points:

$$\sup_{\mathbf{x} \in \mathcal{X}} \|\mathbf{g}'_n(\mathbf{x}) - \mathbf{f}'(\mathbf{x})\|_d = O(\sqrt{h}), \text{ as } n \rightarrow \infty \text{ (equivalently, as } h \rightarrow 0). \quad (46)$$

The constant associated with $O(\sqrt{h})$ depends only upon d and $f(\cdot)$.

Proof: For any $i \in \{1, \dots, d\}$, and for arbitrary $\mathbf{X}_{-i}^* = (x_1^*, \dots, x_{i-1}^*, x_{i+1}^*, \dots, x_d^*)^T \in \prod_{j \neq i} \mathcal{X}_j$, for any $x \in \mathcal{X}_i$, let

$$g_{in}(x|\mathbf{X}_{-i}^*) = g_n(x_1^*, \dots, x_{i-1}^*, x, x_{i+1}^*, \dots, x_d^*); \quad (47)$$

$$f_i(x|\mathbf{X}_{-i}^*) = f(x_1^*, \dots, x_{i-1}^*, x, x_{i+1}^*, \dots, x_d^*). \quad (48)$$

Since $x \in \mathcal{X}_i$, it must belong to some interval of the form $[x_{ji}, x_{j+1i}]$, for some $j \in \{1, 2, \dots, \tilde{n}_i - 1\}$. Let us fix that j . For $y = x_{ji}$ and $y = x_{j+1i}$, $g_{in}(y|\mathbf{X}_{-i}^*) = f_i(y|\mathbf{X}_{-i}^*)$ by interpolation property of the posterior Gaussian process, assuming that $(x_1^*, \dots, x_{i-1}^*, y, x_{i+1}^*, \dots, x_d^*) \in \mathbf{X}_n$ for $y = x_{ji}$ and $y = x_{j+1i}$. That is, $g_{in}(y|\mathbf{X}_{-i}^*) - f_i(y|\mathbf{X}_{-i}^*) = 0$ for $y = x_{ji}$ and $y = x_{j+1i}$. Hence, by Rolle's theorem, $g'_{in}(u|\mathbf{X}_{-i}^*) - f'_i(u|\mathbf{X}_{-i}^*) = 0$, for some $u \in (x_{ji}, x_{j+1i})$. This permits the following representation:

$$g'_{in}(x|\mathbf{X}_{-i}^*) - f'_i(x|\mathbf{X}_{-i}^*) = \int_u^x (g''_{in}(v|\mathbf{X}_{-i}^*) - f''_i(v|\mathbf{X}_{-i}^*)) dv, \quad (49)$$

The hypothesis of Lipschitz continuity of the 4-th order mixed partial derivatives of the correlation function ensures existence and sample path continuity of $g''_{in}(v|\mathbf{X}_{-i}^*)$. Hence, by the Cauchy-Schwartz inequality we obtain from (49), the following:

$$\begin{aligned} & |g'_{in}(x|\mathbf{X}_{-i}^*) - f'_i(x|\mathbf{X}_{-i}^*)| \\ & \leq \left[\int_u^x (g''_{in}(v|\mathbf{X}_{-i}^*) - f''_i(v|\mathbf{X}_{-i}^*))^2 dv \right]^{1/2} \times |x - u|^{1/2} \\ & \leq \left[\int_{\mathcal{X}_i} (g''_{in}(v|\mathbf{X}_{-i}^*) - f''_i(v|\mathbf{X}_{-i}^*))^2 dv \right]^{1/2} \times h_i^{1/2} \\ & \leq \sup_{\mathbf{X}_{-i}^* \in \prod_{j \neq i} \mathcal{X}_j} \left[\int_{\mathcal{X}_i} (g''_{in}(v|\mathbf{X}_{-i}^*) - f''_i(v|\mathbf{X}_{-i}^*))^2 dv \right]^{1/2} \times h^{1/2}. \end{aligned} \quad (50)$$

Now, since the hypotheses of this theorem constitute a special case of Theorem 2, the result of almost sure uniform boundedness of $|g''_{in}(\cdot)|$ as $n \rightarrow \infty$, is valid here. Specifically, from the proof of Theorem 2 in this case it holds that $\lim_{n \rightarrow \infty} \sup_{(u, \mathbf{X}_{-i}^*) \in \mathcal{X}} |g''_{in}(v|\mathbf{X}_{-i}^*)| < \infty$. This, along with continuity of $f''_i(v|\mathbf{X}_{-i}^*)$ and compactness of $\mathcal{X}$, shows that the integral term is bounded by a constant C_i (depending only on f and d) almost surely for large n . Hence the right-hand side of (50) is $O(\sqrt{h})$.

Hence, switching to our usual notation, it follows that

$$\sup_{\mathbf{x} \in \mathcal{X}} (g'_{in}(\mathbf{x}) - f'_i(\mathbf{x}))^2 = O(h), \text{ almost surely, as } n \rightarrow \infty. \quad (51)$$

Since $\sup_{\mathbf{x} \in \mathcal{X}} \sum_{i=1}^d (g'_{in}(\mathbf{x}) - f'_i(\mathbf{x}))^2 \leq \sum_{i=1}^d \sup_{\mathbf{x} \in \mathcal{X}} (g'_{in}(\mathbf{x}) - f'_i(\mathbf{x}))^2$, it follows from (51) that

$$\sup_{\mathbf{x} \in \mathcal{X}} \|\mathbf{g}'_n(\mathbf{x}) - \mathbf{f}'(\mathbf{x})\|_d = O(\sqrt{h}), \text{ almost surely, as } n \rightarrow \infty \text{ (equivalently, as } h \rightarrow 0),$$

proving (46). Note that the constant associated with $O(\sqrt{h})$ above depends only upon d and f . $\square$

The following result holds as a consequence of Theorem 3.

Theorem 4: Under the conditions of Theorem 3, the following holds with the forms of the input points as specified in Theorem 3: for almost all sequences $\{g_n(\cdot)\}_{n=1}^\infty$,

$$\sup_{\mathbf{x} \in \mathcal{X}} |g_n(\mathbf{x}) - f(\mathbf{x})| = O(h^{3/2}), \text{ as } n \rightarrow \infty. \quad (52)$$

The constant associated with $O(h^{3/2})$ depends only upon d and f .

Proof: As in the proof of Theorem 3, for any $x \in \mathcal{X}_i$, it must belong to some interval of the form $[x_{ji}, x_{j+1i}]$, for some $j \in \{1, 2, \dots, \tilde{n}_i - 1\}$. Let us fix that j . Now, $g_{in}(x_{ji}|\mathbf{X}_{-i}^*) = f_i(x_{ji}|\mathbf{X}_{-i}^*)$ almost surely, by interpolation property of the posterior process $g_n(\cdot)$, assuming that $(x_1^*, \dots, x_{i-1}^*, x_{ji}, x_{i+1}^*, \dots, x_d^*) \in \mathbf{X}_n$. Hence,

$$g_{in}(x|\mathbf{X}_{-i}^*) - f_i(x|\mathbf{X}_{-i}^*) = \int_{x_{ji}}^x (g'_{in}(v|\mathbf{X}_{-i}^*) - f'_i(v|\mathbf{X}_{-i}^*)) dv. \quad (53)$$

The Cauchy-Schwartz inequality applied to (53) gives

$$|g_{in}(x|\mathbf{X}_{-i}^*) - f_i(x|\mathbf{X}_{-i}^*)| \leq \left[\int_{x_{ji}}^x (g'_{in}(v|\mathbf{X}_{-i}^*) - f'_i(v|\mathbf{X}_{-i}^*))^2 dv \right]^{1/2} \times |x - x_{ji}|^{1/2}. \quad (54)$$

From (50) it follows that the integral on the right hand side of (54) is $O(h) \times |x - x_{ji}|$, almost surely, as $n \rightarrow \infty$. Recall that the constant associated with $O(h) \times |x - x_{ji}|$ depends only upon d and f . Since $|x - x_{ji}|$ is bounded above by h , it follows that the right hand side of (54) is $O(h^{3/2})$, almost surely, as $n \rightarrow \infty$. Switching to the usual notation, it is seen that (52) holds. $\square$

Remark 2: As $h \rightarrow 0$, $\mathbf{g}'(\cdot)$ uniformly converges to $\mathbf{f}'(\cdot)$ at the rate $h^{1/2}$ and $g(\cdot)$ uniformly converges to $f(\cdot)$ at the rate $h^{3/2}$, almost surely with respect to their posteriors.

Remark 3: In Theorems 1, 2, 3 and 4 we have referred to the posterior (22), which corresponds to a linear mean structure of the Gaussian process prior and conjugate priors for β and σ^2 . However, as can be seen from the proofs, both the theorems continue to hold for any mean function that has continuous second order mixed partial derivatives and any prior on the parameters, including the parameters of the correlation function such that the posteriors of the parameters are proper.

Remark 4: The hypotheses of Theorems 3 and 4 require input points of the form $(x_1^*, \dots, x_{i-1}^*, x_{ji}, x_{i+1}^*, \dots, x_d^*)$ to belong to $\mathbf{D}_n$ for $i = 1, \dots, d$, and for $j = 1, \dots, \tilde{n}_i$, where $(x_1^*, \dots, x_{i-1}^*, x_{i+1}^*, \dots, x_d^*) \in \prod_{j \neq i} \mathcal{X}_j$ may be chosen arbitrarily. Now observe that if $\mathcal{X}_i = [a, b]$, for $i = 1, \dots, d$, for some $a < b$, then we can set $\tilde{n}_i = n$ and $h_i = h$, for $i = 1, \dots, d$. In such cases, inclusion of the set of n^d points $\{(x_{j_1}, \dots, x_{j_d}) : j_1, \dots, j_d \in \{1, \dots, n\}\}$ in $\mathbf{D}_n$ is sufficient for Theorems 3 and 4 to hold. However, when d and n are even moderately large, n^d is an extremely large number, which would prohibit computation of Σ_{22}^{-1} , and hence computation of the posterior of $\mathbf{g}'(\cdot)$. Hence, for practical purposes it makes sense to refer to the general setup of Theorems 1 and 2.

5. Algorithm for optimization with the Gaussian process derivative method

We now propose a general methodology for function optimization, which judiciously exploits the posterior form (25). Without loss of generality, we consider the minimization problem for notational convenience. In a nutshell, the initial stage (say, the 0-th stage) of the methodology involves simulations from $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_n)$ satisfying $\|\mathbf{f}'(\cdot)\|_d < \epsilon$ for some $\epsilon > 0$ and $\Sigma''(\cdot) > 0$. In the subsequent stages $k = 1, 2, \dots$, previous stage realizations satisfying $\|\mathbf{f}'(\cdot)\|_d < \eta_k$, where $\eta_k \rightarrow 0$ as $k \rightarrow \infty$, are successively augmented with $\mathbf{D}_n$ and realizations from the posterior associated with the augmented data are generated at each stage k by a judicious importance resampling strategy. As $k \rightarrow \infty$, the posteriors given the successively augmented data converge to the true optima.

The importance of the 0-th stage simulation algorithm for generating realizations from $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_n)$ satisfying the restrictions $\|\mathbf{f}'(\cdot)\|_d < \epsilon$ for some $\epsilon > 0$ and $\Sigma''(\cdot) > 0$, is enormous, particularly because the entire d -dimensional random variable $\mathbf{x}^*$ must be updated in a single block to meet the restrictions. Traditional MCMC algorithms are not known to be efficient in such problems as even for moderately large dimensions good proposal distributions are difficult to devise, and the acceptance rates can be poor, along with poor mixing properties. In this regard, the transformation based Markov Chain Monte Carlo (TMCMC) proposed by Dutta and Bhattacharya (2014) is an effective methodology. Indeed, TMCMC is designed to update all (or most of) the components of the high-dimensional random variable in a single block using appropriate deterministic transformations of some single (or low-dimensional) random variable. As such, this strategy drastically reduces effective dimensionality, which is responsible for maintaining good acceptance rates in spite of high dimensions. Good mixing properties can also be ensured by judiciously choosing the relevant “move-types”, and judicious mixtures of additive and multiplicative transformations usually lead to desired mixing properties. For details on TMCMC and its properties, see Dutta and Bhattacharya (2014), Dey and Bhattacharya (2016), Dey and Bhattacharya (2017), Dey and

Bhattacharya (2019). As such, we recommend TMCMC for our optimization methodology. We provide the detailed Bayesian optimization methodology below as Algorithm 1.

A detailed analysis of the computational complexity of Algorithm 1 is presented in Appendix A. In summary, each TMCMC iteration (step (1)) requires $O(d^3 + d^2n_0 + dn_0^2)$ operations, where d is the input dimension and n_0 the initial dataset size. The total complexity of step (1) is $O(N_{\text{TMCMC}}d^3)$ when d is moderate or large, with N_{TMCMC} the number of TMCMC iterations. The refinement stage (step (2)) adds $O(KN(d^3 + d^2N_{\text{max}} + dN_{\text{max}}^2) + K_{\text{aug}}C_f)$ operations, where K is the number of stages, N the number of stored samples, N_{max} the maximum dataset size, K_{aug} the number of stages where augmentation occurs, and C_f the cost of evaluating the objective function f once. For the experimental settings in this paper, $C_f = O(d^2)$ and the TMCMC term $O(N_{\text{TMCMC}}d^3)$ dominates the runtime.

5.1. Practical guidance for Algorithm 1

To facilitate reproducibility and effective application of Algorithm 1, we provide the following practical recommendations.

5.1.1. Initial design

In all our experiments, we used a simple initial design consisting of $n = 10$ points uniformly spaced along the main diagonal of $\mathcal{X} = [-10, 10]^d$, that is, $\mathbf{x}_i = (x_i, \dots, x_i)$ with $x_i = -10 + 2(i - 1)$. This very sparse design suffices because the algorithm subsequently augments the dataset adaptively. For general applications, we recommend starting with a small number of points (for example, $n = 10$ to 20) chosen either along a diagonal or as a simple random sample; the algorithm is robust to the initial design as long as the domain is covered to some extent. More sophisticated space-filling designs (for example, Latin hypercube or Sobol sequences) may be used if n is kept small (for example, $n = 10d$) (for example, McKay *et al.* (1979), Sobol' (1967)), but we do not advocate dense grids because they would make the initial GP computations infeasible.

5.1.2. Choice of tolerance decay η_k

The sequence η_k should converge to zero slowly enough to maintain stable importance weights. In our experiments we used the following choices (with $k = 1, 2, \dots$):

- (a) For $d = 1, 2$: $\eta_k = 1/(10 + k - 1)^2$ (quadratic decay).
- (b) For $d = 5$: $\eta_k = 1.5/\log(10 + k)$.
- (c) For $d = 10$: $\eta_k = 7/\log(10 + k - 1)$.
- (d) For $d = 50$: $\eta_k = 200/\log(10 + k - 1)$.
- (e) For $d = 100$: $\eta_k = 850/\log(10 + k - 1)$.

In all cases $\eta_k \rightarrow 0$ and the decay is sufficiently slow to keep importance weights stable. For other problems, we recommend a logarithmic decay $\eta_k = C/\log(10 + k)$ for moderate

Algorithm 1 Optimization with Gaussian process derivatives

(1) First simulate N realizations $\{\mathbf{x}_1^*, \dots, \mathbf{x}_N^*\}$ from $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_n)$ given by (25) using TCMCMC, where the prior for $\mathbf{x}^*$ is given by (26) for some pre-fixed $\epsilon > 0$. This includes simulations from posteriors associated with most plausible functions $g(\cdot)$ satisfying $\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}$ for $\mathbf{x}^* \in \mathcal{X}$, given $\mathbf{D}_n$. That is, $\{\mathbf{x}_i^*; i = 1, \dots, N\}$, represents the set of solutions for $\mathbf{g}'(\mathbf{x}^*) = \mathbf{0}$ for functions $g(\cdot)$ that satisfy $g(\mathbf{x}_i) = f(\mathbf{x}_i); i = 1, \dots, n$. Thanks to the prior (26), these solutions further satisfy $\|\mathbf{f}'(\mathbf{x}_i^*)\|_d < \epsilon$ and $\Sigma''(\mathbf{x}_i^*) > 0$, for $i = 1, \dots, N$. Note that $\Sigma''(\mathbf{x}^*) > 0$ can be checked by computing the eigenvalues and checking if all the eigenvalues are positive. But a more efficient alternative is to check if Cholesky decomposition of $\Sigma''(\mathbf{x}^*)$ is possible, the information of which is provided by the subroutines of the BLAS and LAPACK libraries. We exploit the latter for our implementation.

(2) For stages $k = 1, 2, 3, \dots$,

(i) For $i = 1, \dots, N$, compute importance weights proportional to

$$w_k(\mathbf{x}_i^*) = \begin{cases} 1 & \text{if } k = 1; \\ w_{k-1}(\mathbf{x}_i^*) \times \frac{\pi(\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0} | \mathbf{D}_{n+\sum_{j=0}^{k-1} n_j}, \mathbf{x}_i^*)}{\pi(\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0} | \mathbf{D}_{n+\sum_{j=0}^{k-2} n_j}, \mathbf{x}_i^*)} & \text{if } k \geq 2, \end{cases} \quad (55)$$

where $n_0 = 0$.

(ii) Without replacement, select a subsample $\{\mathbf{x}_{i_1}^*, \dots, \mathbf{x}_{i_M}^*\}$ from $\{\mathbf{x}_1^*, \dots, \mathbf{x}_N^*\}$ with probabilities proportional to $w_k(\mathbf{x}_i^*)$; $i = 1, \dots, N$. Note that, as $M \rightarrow \infty$ and $N \rightarrow \infty$ such that $M/N \rightarrow 0$, $\mathbf{x}_{i_j}^*$; $j = 1, \dots, M$, follow the distribution $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_{n+\sum_{j=1}^{k-1} n_j})$. The recursively computed importance weights $w_k(\mathbf{x}_i^*)$; $i = 1, \dots, N$, are expected to be stable, since for each stage k , for $i = 1, \dots, N$, the factors $\frac{\pi(\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0} | \mathbf{D}_{n+\sum_{j=1}^{k-1} n_j}, \mathbf{x}_i^*)}{\pi(\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0} | \mathbf{D}_{n+\sum_{j=1}^{k-2} n_j}, \mathbf{x}_i^*)}$ in (55) are not expected to be very different from 1 if n_{k-1} is not significantly greater than zero. Note that the importance weights $w_k(\mathbf{x}_i^*)$, for $i = 1, \dots, N$, can be computed simultaneously on parallel processors. This, along with stability of the recursive formulation (55), is expected to make for an efficient computational strategy.

(iii) For $j = 1, \dots, M$, check if $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$, where $\eta_k \rightarrow 0$ as $k \rightarrow \infty$. If $\mathbf{x}_{i_j}^*$ satisfies this condition, then $\mathbf{x}_{i_j}^*$ is a realization from $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_{n+\sum_{j=1}^{k-1} n_j})$, where the prior for $\mathbf{x}^*$ is uniform on $B(\eta_k)$, the form of which is given by (27).

(iv) Let n_k (≥ 0) realizations among the M realizations satisfy the condition $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$. Without loss of generality, assume that $\mathbf{x}_{i_j}^*$; $j = 1, \dots, n_k$ are such realizations. Compute $f(\mathbf{x}_{i_j}^*)$; $j = 1, \dots, n_k$, and augment $(\mathbf{x}_{i_j}^*, f(\mathbf{x}_{i_j}^*))$; $j = 1, \dots, n_k$, with $\mathbf{D}_{n+\sum_{j=0}^{k-1} n_j}$ to form $\mathbf{D}_{n+\sum_{j=0}^k n_j}$. In practice, to avoid excessive growth of the dataset and numerical instability, we recommend augmenting at most 5 points per stage (see also Remark 5).

(v) Store the realizations $\{\mathbf{x}_{i_j}^* : j = 1, \dots, n_k\}$.

to high dimensions ($d \geq 5$), with C chosen such that the initial η_1 is about 10% of the typical gradient norm at random points (for example, C roughly proportional to d). For low dimensions ($d \leq 2$), a quadratic decay works well.

5.1.3. Stopping criteria

In our experiments, we simply fixed the number of refinement stages to $S = 40$ for dimensions $d = 2, 5, 10$ and $S = 100$ for $d = 50, 100$. In all cases, n_k became zero after a small number of stages (typically $k \leq 10$ for low dimensions, $k \leq 4$ for high dimensions). For general use, we recommend the following adaptive stopping criteria:

- (a) Stop if $n_k = 0$ for three consecutive stages (no new points satisfy the gradient bound).
- (b) Stop if the gradient norm $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d$ for the best candidate has not decreased by more than 1% over the last 10 stages.
- (c) Stop after a maximum number of stages $K_{\max} = 100$ (safety bound).

With these criteria, the algorithm typically terminates after 10–40 stages in practice.

5.2. Pseudo-code for key components

To aid implementation, we provide pseudo-code for the TMCMC and the importance resampling components of Algorithm 1.

TMCMC (single iteration):

Input: current state $\mathbf{x}$, target log-density $L(\mathbf{x})$, scaling parameters a_1, a_2 ,
move probabilities p, q (e.g., 0.5)
Output: next state $\mathbf{x}'$

```

1. Draw  $U \sim \text{Uniform}(0,1)$ 
2. if  $U < p$ :
    // Additive move
     $\epsilon \sim N(0,1)$ 
    for  $j=1..d$ :  $b_j \sim \text{Uniform}(\{-1,1\})$ 
     $y_j = x_j + b_j * a_1 * |\epsilon|$ 
    accept with prob  $\min(1, \exp(L(y)-L(x)))$ 
3. else:
    // Multiplicative move
     $\epsilon \sim \text{Uniform}(-1,1)$ 
    for  $j=1..d$ :  $b_j \sim \text{Uniform}(\{-1,0,1\})$ 
     $y_j = x_j * \epsilon$  if  $b_j=1$ 
     $y_j = x_j / \epsilon$  if  $b_j=-1$ 
     $y_j = x_j$  otherwise
     $J = |\epsilon|^{\sum(b_j)}$ 

```

```

    accept with prob min(1, exp(L(y)-L(x)) * J)
4. Set  $x_{\text{tilde}} = \text{accepted } y \text{ or } x$ 
5. Draw  $U \sim \text{Uniform}(0,1)$ 
6. if  $U < q$ :
    // Additive refinement
    epsilon  $\sim N(0,1)$ 
    for  $j=1..d$ :  $y_j = x_{\text{tilde}_j} + a2 * |\text{epsilon}|$  with prob 0.5,
        else  $- a2 * |\text{epsilon}|$ 
    accept with prob min(1, exp(L(y)-L( $x_{\text{tilde}}$ )))
7. else:
    // Multiplicative refinement
    epsilon  $\sim \text{Uniform}(-1,1)$ 
    if  $\text{Uniform}(0,1) < 0.5$ :  $y_j = x_{\text{tilde}_j} * \text{epsilon}$ ,  $J = |\text{epsilon}|^d$ 
    else:  $y_j = x_{\text{tilde}_j} / \text{epsilon}$ ,  $J = |\text{epsilon}|^{-d}$ 
    accept with prob min(1, exp(L(y)-L( $x_{\text{tilde}}$ )) * J)
8. return accepted value

```

Importance resampling at stage k :

Input: prior samples $\{x_i\}$ ($i=1..N$), current dataset $D_{\{k-1\}}$, target dataset D_k

Output: resampled indices $\{i_1, \dots, i_M\}$

1. For each i , compute $w_i = w_{\{k-1\}}(x_i) * L_k(x_i) / L_{\{k-1\}}(x_i)$
where $L_k(x) = \pi(g'(x)=0 \mid D_k, x)$
2. Normalize w_i to sum to 1
3. Sample M indices without replacement with probabilities w_i
4. Return the sampled indices

5.3. Further discussion of Algorithm 1

Algorithm 1 begins by simulating from the posterior of $\mathbf{x}^*$ satisfying $\|\mathbf{f}'(\mathbf{x}^*)\|_d < \epsilon$ and $\Sigma''(\mathbf{x}^*) > 0$. In the subsequent steps $k \geq 1$, the set of realizations $\{\mathbf{x}_{i_j}^* : j = 1, \dots, n_k\}$ generated by importance resampling further satisfy $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$ along with $\Sigma''(\mathbf{x}_{i_j}^*) > 0$, for $j = 1, \dots, n_k$. The implication is that, ϵ may be chosen somewhat larger to achieve reasonably good TMCMC mixing and acceptance rates. Indeed, if n is not large enough, then $\mathbf{g}'$ is not expected to be sufficiently close to $\mathbf{f}'$, and hence for too small ϵ , $\{\mathbf{x}^* : \|\mathbf{f}'(\mathbf{x}^*)\|_d < \epsilon\}$ would be too small a region to contain the solutions $\{\mathbf{x}^* : \|\mathbf{g}'(\mathbf{x}^*)\|_d = 0\}$, given $\mathbf{D}_n$. This would result in poor TMCMC mixing.

Once adequate mixing with reasonable acceptance rates are achieved with relatively small n and relatively large ϵ , the subsequent steps increase the data size by augmenting the data with those values that satisfy $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$. Thus, in the subsequent steps, these data points help better approximate the region around the stationary points of f by the posterior, and enables more simulations from the region $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$, finally leading to convergence of the solutions $\{\mathbf{x}^* : \mathbf{g}'_k(\mathbf{x}^*) = \mathbf{0}, \Sigma''(\mathbf{x}^*) > 0\}$ to $\{\mathbf{x}^* : \mathbf{f}'(\mathbf{x}^*) = \mathbf{0}, \Sigma''(\mathbf{x}^*) > 0\}$, almost

surely, as $k \rightarrow \infty$, where $\mathbf{g}'_k(\cdot)$ denotes any realization from the posterior of $\mathbf{g}'(\cdot)$ given $\mathbf{D}_{n+\sum_{j=0}^{k-1} n_j}$. This intuition is formalized below as Theorem 5.

Theorem 5: Consider the setup of Theorem 2 (or more specifically, that of Theorem 3). Then, as $k \rightarrow \infty$, the set $\{\mathbf{x}_{i_j}^* : j = 1, \dots, n_k\}$ of Algorithm 1 almost surely contains all the local minima of the objective function $f(\cdot)$, as $M \rightarrow \infty$ and $N \rightarrow \infty$ such that $M/N \rightarrow 0$.

Proof: Note that at stage k , as $M \rightarrow \infty$ and $N \rightarrow \infty$ such that $M/N \rightarrow 0$, $\mathbf{x}_{i_j}^* ; j = 1, \dots, n_k$, arise from $\pi(\mathbf{x}^* | \mathbf{g}'(\mathbf{x}^*) = \mathbf{0}, \mathbf{D}_{n+\sum_{j=0}^{k-1} n_j})$, subject to $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$ and $\Sigma''(\mathbf{x}_{i_j}^*) > 0$ for $j = 1, \dots, n_k$. These realizations are solutions of $\mathbf{g}'_k(\mathbf{x}^*) = \mathbf{0}$ and $\Sigma''(\mathbf{x}^*) > 0$ when the data observed is $\mathbf{D}_{n+\sum_{j=0}^{k-1} n_j}$. By Theorem 2 (or more specifically by Theorem 3), as $k \rightarrow \infty$ (equivalently, as $h \rightarrow 0$ in Theorem 3), $\mathbf{g}'_k(\cdot)$ uniformly converges to $\mathbf{f}'(\cdot)$ almost surely. Hence, as $k \rightarrow \infty$,

$$\{\mathbf{x}^* : \mathbf{g}'_k(\mathbf{x}^*) = \mathbf{0}, \|\mathbf{f}'(\mathbf{x}^*)\|_d < \eta_k, \Sigma''(\mathbf{x}^*) > 0\} \rightarrow \{\mathbf{x}^* : \mathbf{f}'(\mathbf{x}^*) = \mathbf{0}, \Sigma''(\mathbf{x}^*) > 0\}, \quad (56)$$

almost surely.

Due to (56), as $k \rightarrow \infty$, the set $\{\mathbf{x}_{i_j}^* : j = 1, \dots, n_k\}$ contains all the local minima of the objective function $f(\cdot)$, as $M \rightarrow \infty$ and $N \rightarrow \infty$ such that $M/N \rightarrow 0$. $\square$

Remark 5: In step (2) (iv) of Algorithm 1 we have suggested augmentation of all realizations $(\mathbf{x}_{i_j}^*, f(\mathbf{x}_{i_j}^*))$ satisfying $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$ to the existing data $\mathbf{D}_{n+\sum_{j=0}^{k-1} n_j}$. In practice, augmentation of all such realizations may enlarge the dataset to such an extent that invertibility of the resultant Σ_{22} may be infeasible or numerically unstable, so that computation of the corresponding posterior densities of $\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0}$, and hence the importance weights (55), may not yield reliable results. Hence in practice, as a rule of thumb, we recommend augmenting at most 5 realizations satisfying $\|\mathbf{f}'(\mathbf{x}_{i_j}^*)\|_d < \eta_k$, which we consider in all our applications.

Remark 6: As the stage number k in step (2) of Algorithm 1 increases, n_k decreases. Hence, in practice, n_k will be zero after some large enough k . When d is large, due to the curse of dimensionality, only the first few stages are expect to yield positive n_k .

Remark 7: In step (1) of Algorithm 1, that is, in the TMCMC step, as well as in any stage k of step (2) of the algorithm, provided that n_k is sufficiently large, desired credible regions of the respective posterior distributions of $\mathbf{x}^*$ can be obtained. These quantify the stage-wise uncertainty in *a posteriori* learning about the optima, given $\|\mathbf{f}'(\cdot)\|_d < \epsilon$ or $\|\mathbf{f}'(\cdot)\|_d < \eta_k$. As $k \rightarrow \infty$, the uncertainty decreases, and the credibility regions shrink to the points representing the true optima. However, as mentioned in Remark 2, in practice, particularly for large d , n_k would be zero for most stages k , which would preclude computation of credible regions for most stages.

Remark 8: Note that if it is known beforehand that there is a single global minimum of $f(\cdot)$ on $\mathcal{X}$, then step (2) of Algorithm 1 is not required. It is then sufficient to report $\mathbf{x}_{i^*}^*$ as the (approximate) minimizer of $f(\cdot)$, where $i^* = \arg \min\{f(\mathbf{x}_i^*) : i = 1, \dots, N\}$.

6. Bayesian characterization of the number of local minima of the objective function with recursive posteriors

Steps (2) (iii) and (2) (iv) can be combined to obtain a Bayesian characterization of the number of local minima of the objective function. In this regard, for stage j , let us define $Y_j = \sum_{r=1}^M I_{B(\eta_j)}(\mathbf{x}_{ir}^*)$, where $B(\eta_j)$ is given by (27). Thus, $\{Y_j = m\}$ with probability p_{mj} , for $m = 0, 1, 2, \dots, M$. Since $M \rightarrow \infty$, we allow Y_j to take values on the entire set of non-negative integers. That is, we set

$$P(Y_j = m) = p_{mj}; \quad m = 0, 1, 2, \dots, \quad (57)$$

the infinite-dimensional multinomial distribution, where $0 \leq p_{mj} \leq 1$ for $m = 0, 1, 2, \dots$ and $j \geq 1$. Further, $\sum_{m=0}^{\infty} p_{mj} = 1$ for all $j \geq 1$. We assume that the true probabilities $p_{m0} \in [0, 1]$; $m = 0, 1, 2, \dots$, such that $\sum_{m=0}^{\infty} p_{m0} = 1$, are unknown. Indeed, if $f(\cdot)$ has finite number of local minima, then there must exist $\tilde{m} \geq 0$ such that $p_{\tilde{m}0} = 1$ and $p_{m0} = 0$ for $m \neq \tilde{m}$. For infinite number of local minima, we must have $p_{m0} = 0$ for any finite integer $m \geq 0$.

We adopt the approach of Roy and Bhattacharya (2020) based on Dirichlet process (Ferguson (1973)) to obtain the posterior distribution of the infinite set of parameters $\{p_{mk}; m = 0, 1, 2, \dots\}$. Roy and Bhattacharya (2020) came up with the following strategy for characterizing possibly infinite number of limit points of infinite series, which we usefully employ for our current purpose, albeit with a definition of Y_j that is different from that of Roy and Bhattacharya (2020).

Let $\mathcal{Y} = \{1, 2, \dots\}$ and let $\mathcal{B}(\mathcal{Y})$ denote the Borel σ -field on $\mathcal{Y}$ (assuming every singleton of $\mathcal{Y}$ is an open set). Let $\mathcal{P}$ denote the set of probability measures on $\mathcal{Y}$. Then, at the first stage,

$$\pi(Y_1|P_1) \equiv P_1, \quad (58)$$

where $P_1 \in \mathcal{P}$. We assume that P_1 is the following Dirichlet process:

$$\pi(P_1) \equiv DP(G), \quad (59)$$

where, the probability measure G is such that, for every $j \geq 1$,

$$G(Y_j = m) = \frac{1}{2^m}. \quad (60)$$

Then

$$\pi(P_1|y_1) \equiv DP(G + \delta_{y_1}),$$

where, for any x , δ_x denotes point mass at x .

At the second stage, we set the prior for P_2 to be the posterior given y_1 corresponding to $DP\left(\left(1 + \frac{1}{2^2}\right)G\right)$ prior for P_1 . That is, $\pi(P_2) \equiv DP\left(\left(1 + \frac{1}{2^2}\right)G + \delta_{y_1}\right)$. Then, with respect to this prior for P_2 , the posterior of P_2 is given by

$$\pi(P_2|y_2) \equiv DP\left(\left(1 + \frac{1}{2^2}\right)G + \delta_{y_1} + \delta_{y_2}\right).$$

Continuing this recursive process we obtain that, at the k -th stage, the posterior of P_k to be a Dirichlet process, given by

$$\pi(P_k|y_k) \sim DP\left(\sum_{j=1}^k \frac{1}{j^2} G + \sum_{j=1}^k \delta_{y_j}\right). \quad (61)$$

It follows from (61) that

$$E(p_{mk}|y_k) = \frac{\frac{1}{2^m} \sum_{j=1}^k \frac{1}{j^2} + \sum_{j=1}^k I(y_j = m)}{\sum_{j=1}^k \frac{1}{j^2} + k}; \quad (62)$$

$$Var(p_{mk}|y_k) = \frac{\left(\sum_{j=1}^k \frac{1}{j^2} + \sum_{j=1}^k I(y_j = m)\right) \left((1 - \frac{1}{2^m}) \sum_{j=1}^k \frac{1}{j^2} + k - \sum_{j=1}^k I(y_j = m)\right)}{\left(\sum_{j=1}^k \frac{1}{j^2} + k\right)^2 \left(\sum_{j=1}^k \frac{1}{j^2} + k + 1\right)}. \quad (63)$$

The theorem below characterizes the number of local minima of the objective function $f(\cdot)$ in terms of the limit of the marginal posterior probabilities of p_{mk} , denoted by $\pi_m(\cdot|y_k)$, as $k \rightarrow \infty$.

Theorem 6: Assume the conditions of Theorem 3, and in Algorithm 1, assume that $M \rightarrow \infty$ and $N \rightarrow \infty$ such that $M/N \rightarrow 0$. Then $f(\cdot)$ has $\tilde{m}$ (≥ 0) local minima if and only if

$$\pi_{\tilde{m}}(\mathcal{N}_1|y_k) \rightarrow 1, \text{ almost surely with respect to the posterior (25),} \quad (64)$$

as $k \rightarrow \infty$. In the above, $\mathcal{N}_1$ is any neighborhood of 1 (one).

Proof: First, let us assume that $f(\cdot)$ has $\tilde{m}$ (≥ 0) local minima. Then, by Theorem 5, Algorithm 1 converges to the $\tilde{m}$ local minima almost surely, as $k \rightarrow \infty$, provided that $M \rightarrow \infty$ and $N \rightarrow \infty$ such that $M/N \rightarrow 0$. Hence, there exists $j_0 \geq 1$, such that for $j \geq j_0$, $y_j = \tilde{m}_0$. Thus, almost surely, $I(y_j = \tilde{m}) = 1$, for $j \geq j_0$. Consequently, it easily follows from (62) and (63) that almost surely, as $k \rightarrow \infty$,

$$E(p_{\tilde{m}k}|y_k) \rightarrow 1, \text{ and} \quad (65)$$

$$Var(p_{\tilde{m}k}|y_k) = O\left(\frac{1}{k}\right) \rightarrow 0. \quad (66)$$

Now let $\mathcal{N}_1$ denote any neighborhood of 1, and let ϵ (> 0) be sufficiently small such that $\mathcal{N}_1 \supseteq \{1 - p_{\tilde{m}k} < \epsilon\}$. Then using Markov's inequality we obtain

$$\begin{aligned} \pi_{\tilde{m}}(\mathcal{N}_1|y_k) &\geq \pi_{\tilde{m}}(1 - p_{\tilde{m}k} < \epsilon|y_k) \\ &= 1 - \pi_{\tilde{m}}(1 - p_{\tilde{m}k} \geq \epsilon|y_k) \\ &\geq 1 - \frac{E(1 - p_{\tilde{m}k}|y_k)^2}{\epsilon^2} \\ &= 1 - \frac{1 - 2E(p_{\tilde{m}k}|y_k) + E(p_{\tilde{m}k}^2|y_k)}{\epsilon^2}. \end{aligned} \quad (67)$$

Now, as $k \rightarrow \infty$, $E(p_{\tilde{m}k}|y_k) \rightarrow 1$ by (65), and $E(p_{\tilde{m}k}^2|y_k) = \text{Var}(p_{\tilde{m}k}|y_k) + [E(p_{\tilde{m}k}|y_k)]^2 \rightarrow 1$ by (65) and (66). Hence, the right hand side of (67) converges to 1 almost surely, as $k \rightarrow \infty$. This proves (64).

Now assume that (64) holds for any neighborhood $\mathcal{N}_1$ of 1. Let us fix $\eta \in (0, 1)$. Then given any $\epsilon \in (0, 1 - \eta)$,

$$\pi_{\tilde{m}}(1 - p_{\tilde{m}k} < \epsilon|y_k) \rightarrow 1, \quad (68)$$

almost surely as $k \rightarrow \infty$. The left hand side of (68) admits the following Markov's inequality:

$$\pi_{\tilde{m}}(1 - p_{\tilde{m}k} < \epsilon|y_k) = \pi_{\tilde{m}}(p_{\tilde{m}k} > 1 - \epsilon|y_k) < \frac{E(p_{\tilde{m}k}|y_k)}{1 - \epsilon}. \quad (69)$$

Due to (68), validity of (69) for all $\epsilon \in (0, 1 - \eta)$, and almost sure upper boundedness of $p_{\tilde{m}k}$ by 1, it follows that

$$E(p_{\tilde{m}k}|y_k) \rightarrow 1, \text{ almost surely, as } k \rightarrow \infty. \quad (70)$$

Now, if $f(\cdot)$ is assumed to have m^* local minima where $m^* \neq \tilde{m}$, then due to (65) we must have

$$E(p_{m^*k}|y_k) \rightarrow 1, \text{ almost surely, as } k \rightarrow \infty. \quad (71)$$

Also note that since $0 \leq p_{mk} \leq 1$ for all m and k , the dominated convergence theorem ensures the following, almost surely:

$$1 = \lim_{k \rightarrow \infty} \sum_{m=1}^{\infty} E(p_{mk}|y_k) = \sum_{m=1}^{\infty} \lim_{k \rightarrow \infty} E(p_{mk}|y_k),$$

which implies, due to (71), that $E(p_{\tilde{m}k}|y_k) \rightarrow 0$, almost surely, as $k \rightarrow \infty$. But this would contradict (70). Hence, $f(\cdot)$ must have $\tilde{m}$ local minima. $\square$

7. Experiments

We consider application of Algorithm 1 to 5 different optimization problems, ranging from simple to challenging, several of which are problems of finding both maxima and minima, and one is concerned with saddle points and inconclusiveness in addition to maximum. Encouragingly, all our experiments bring forth the versatility of Algorithm 1 in capturing all the optima, saddlepoints, as well as inconclusiveness of the problems.

As regards TMCMC, in our examples, we expectedly find that in the less challenging, low-dimensional problems, additive TMCMC is sufficient, while in the more challenging cases, we consider appropriate mixtures of additive and multiplicative moves, followed by a further move of a specialized mixture of additive and multiplicative transformations to improve mixing.

In most of our experiments, we discard the first 10^5 TMCMC iterations as burn-in and store every 10-th TMCMC realization in the next 5×10^5 iterations, to obtain 50000 realizations before proceeding to step (2) of Algorithm 1. However, in the 4-th example, due to inadequate mixing, we discard the first 10^6 iterations and store every 10-th realization in the next 5×10^6 iterations to obtain 5×10^5 TMCMC realizations. For the resample size in step (2), we set $M = 1000$.

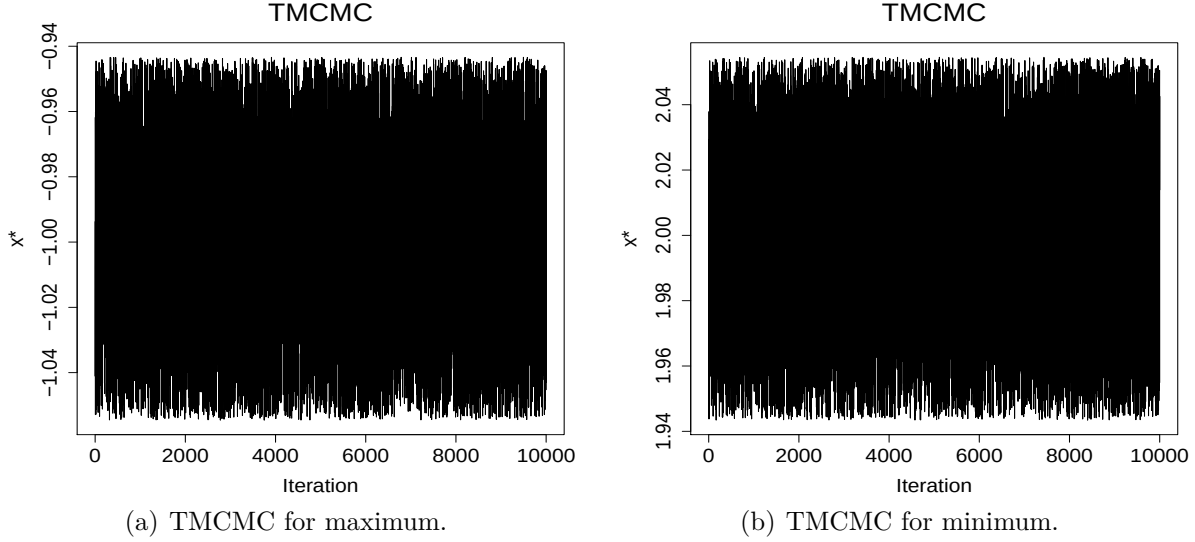


Figure 7.1: TCMC trace plots for Example 1.

For the posterior of $\mathbf{g}'(\cdot) = \mathbf{0}$ given by (23), we set $a = b = 0.1$, $\beta_0 = \mathbf{0}$ and $\Sigma_0 = \mathbb{I}_d$, for all the examples, where d is the dimension relevant to the problem. In all the examples, we also set $\mathcal{X} = [-10, 10]^d$ and the initial input size $n = 10$. With $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$ for $i = 1, \dots, n$, we choose the inputs points as $x_{ik} = -10 + 2(i - 1)$, for $i = 1, \dots, n$ and for $k = 1, \dots, d$. We then evaluate $f(\cdot)$ at each $\mathbf{x}_i$; $i = 1, \dots, n$, to form $\mathbf{D}_n$ with $n = 10$. As we shall demonstrate, this strategy, in conjunction with the rest of the methodology, leads to adequate estimation of the optima in our examples.

7.1. Example 1

We begin with a simple example, where the goal is to obtain the maxima and minima of the function

$$f(x) = 2x^3 - 3x^2 - 12x + 6. \quad (72)$$

Here $f'(x) = 6(x - 2)(x + 1)$ and $f''(x) = 6(2x - 1)$. Hence, this function has a maximum at $x = -1$ and minimum at $x = 2$.

We apply step (1) of Algorithm 1 with $\epsilon = 1$, implementing additive TCMC with equal move-type probabilities for forward and backward transformations. Specifically, at iteration $t = 1, 2, \dots$, letting $x^{(t-1)}$ denote the TCMC realization at iteration $t - 1$, we draw $\varepsilon \sim N(0, 1)$ and consider the transformation $y = x^{(t-1)} + b|\varepsilon|$, where b takes the values 1 and -1 with equal probabilities. We set $x^{(t)} = y$ with probability

$$\alpha = \min \left\{ 1, \frac{\pi(y)\pi(g'(y) = 0 | \mathbf{D}_n, y)}{\pi(x^{(t-1)})\pi(g'(x^{(t-1)}) = 0 | \mathbf{D}_n, x^{(t-1)})} \right\},$$

and set $x^{(t)} = x^{(t-1)}$ with the remaining probability. This is of course same as the ordinary random walk Metropolis algorithm since the dimension here is $d = 1$; see Dutta and Bhattacharya (2014) for ramifications and detailed discussions. Figure 7.1 shows the trace plots of x^* for maximum and minimum, where to reduce the figure file size, we thinned the original TCMC sample of size $N = 50000$ to 10000 by displaying every 5-th realization. The trace

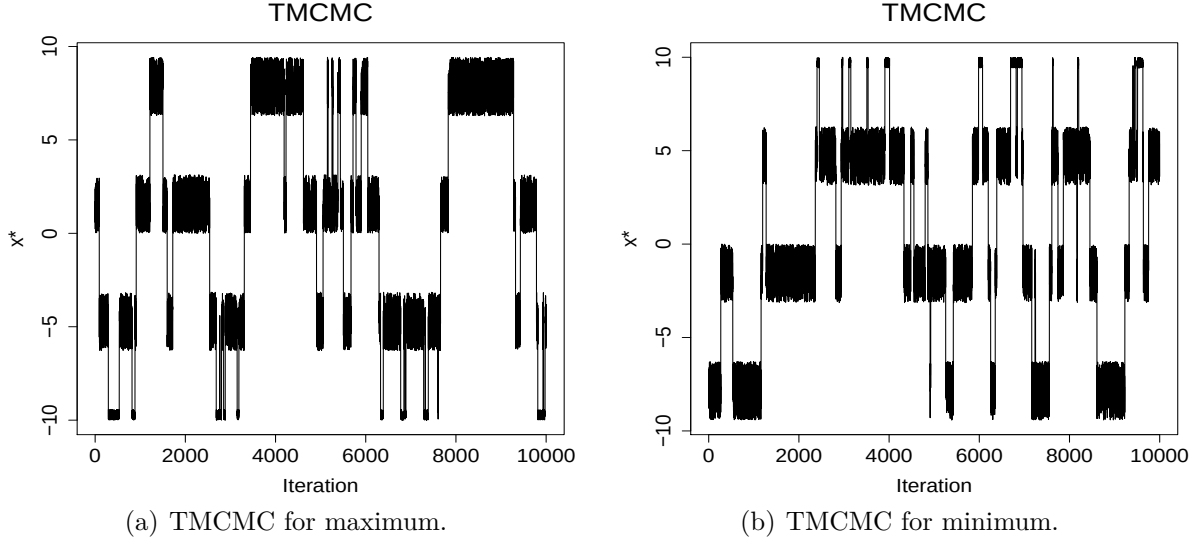


Figure 7.2: TCMC trace plots for Example 2.

plots indicate adequate mixing properties of TCMC. We run step (2) of Algorithm 1 for $k = 1, \dots, S$ stages with $S = 40$, setting $\eta_k = 1/(10 + k - 1)^2$, computing the importance weights for the N TCMC realizations at each stage on 100 64-bit cores in a VMWare parallel computing environment. The cores have 2.80 GHz speed, and have access to 1 TB memory. All our codes are written in C, using the Message Passing Interface (MPI) protocol for parallel processing. As such, our entire exercise is completed in about 2 minutes. We obtain $\hat{x}_{\max} = -0.999995$ and $\hat{x}_{\min} = 2.000023$, as our estimates of the maximum and the minimum, respectively, which are quite accurate.

7.2. Example 2

We now consider maximization and minimization of the function $f(x) = \sin(x)$ for $x \in [-10, 10]$. Here, the true maxima are $x_{\max} = \{\frac{\pi}{2} - 2\pi = -4.712389, \frac{\pi}{2} = 1.570796, \frac{\pi}{2} + 2\pi = 7.853982\}$, and the minima are $x_{\min} = \{-7.853982, -1.570796, 4.712389\}$.

We implement Algorithm 1 in the same way as in Example 1. Figure 7.2 displays the TCMC trace plots, thinned to 10000 realizations to reduce figure file sizes. Clear tri-modality can be visualized from both the trace plots. After implementing step (2) of Algorithm 1 for $S = 40$ stages on our VMWare parallel computing architecture, we obtain $\hat{x}_{\max} = \{-4.712581, 1.570655, 7.860879\}$ and $\hat{x}_{\min} = \{-7.854051, -1.570423, 4.713222\}$, which turn out to be adequate approximations to the truths. Again, the entire exercise takes about 2 minutes.

7.3. Example 3

Let us now consider a two-dimensional example, given by $f(x_1, x_2) = x_1 x_2 (x_1 + x_2) (1 + x_2)$.

The first derivatives are given by $f'_1(x_1, x_2) = x_2(2x_1 + x_2)(x_2 + 1)$ and $f'_2(x_1, x_2) =$

$$x_1(3x_2^2 + 2x_2(x_1 + 1) + x_1).$$

The second derivatives are $f''_{11}(x_1, x_2) = 2x_2(x_2 + 1)$, $f''_{12}(x_1, x_2) = f''_{21}(x_1, x_2) = 4x_1x_2 + 3x_2^2 + 2(x_1 + x_2)$ and $f''_{22}(x_1, x_2) = 2x_1(3x_2 + x_1 + 1)$.

Consider the determinant $D(x_1, x_2) = f''_{11}(x_1, x_2)f''_{22}(x_1, x_2) - [f''_{12}(x_1, x_2)]^2$. Now, if (a, b) is any critical point of $f(\cdot)$ satisfying $f'_1(a, b) = 0$ and $f'_2(a, b) = 0$, then (a, b) is a local maximum if $D(a, b) > 0$ and $f''_{11}(a, b) < 0$; (a, b) is a local minimum if $D(a, b) > 0$ and $f''_{11}(a, b) > 0$; (a, b) is a saddle point if $D(a, b) < 0$. Furthermore, if $D(a, b) = 0$, then (a, b) may be either maximum, minimum or even a saddle point, that is, the derivative test remains inconclusive in such cases.

In this example, it is easy to verify that there are four critical points $(0, 0)$, $(0, -1)$, $(1, -1)$ and $(\frac{3}{8}, -\frac{3}{4})$. The last point is a local maximum; $(0, -1)$ and $(1, -1)$ are saddle points, and the derivative test remains inconclusive about $(0, 0)$.

We implement Algorithm 1 using the above conditions on $D(\mathbf{x}^*)$ and $f''_{11}(\mathbf{x}^*) > 0$, along with the condition $\|\mathbf{f}'(\mathbf{x}^*)\|_2 < \epsilon$, with $\epsilon = 1$, in the prior for $\mathbf{x}^* = (x_1^*, x_2^*)$, for detection of maxima, minima, saddle points and inconclusiveness.

We implement additive TMCMC with equal move-type probabilities for forward and backward transformations. In these cases, at iteration $t = 1, 2, \dots$, letting $\mathbf{x}^{(t-1)}$ denote the TMCMC realization at iteration $t - 1$, we draw $\varepsilon \sim N(0, 1)$ and consider the transformation $\mathbf{y} = \mathbf{x}^{(t-1)} + \mathbf{b}|\varepsilon|$, where, $\mathbf{b} = (b_1, b_2)^T$ and each of b_1 and b_2 independently takes the values 1 and -1 with equal probabilities. We set $\mathbf{x}^{(t)} = \mathbf{y}$ with probability

$$\alpha = \min \left\{ 1, \frac{\pi(\mathbf{y})\pi(\mathbf{g}'(\mathbf{y}) = \mathbf{0} | \mathbf{D}_n, \mathbf{y})}{\pi(\mathbf{x}^{(t-1)})\pi(\mathbf{g}'(\mathbf{x}^{(t-1)}) = \mathbf{0} | \mathbf{D}_n, \mathbf{x}^{(t-1)})} \right\},$$

and set $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$ with the remaining probability. Note that unlike the one-dimensional setup, this additive TMCMC is no longer equivalent to random walk Metropolis (see Dutta and Bhattacharya (2014) for details).

Implementation of Algorithm 1 for obtaining the maximum and the saddle points took about 7 minutes to complete on our VMWare, implemented on 100 cores.

7.3.1. Maximum

Figure 7.3 displaying the TMCMC trace plots for maxima finding, indicates quite adequate mixing. Running step (2) of Algorithm 1 for $S = 40$ steps yields $\hat{\mathbf{x}}_{max} = (0.376858, -0.752406)$, which is reasonably close to the true maximum.

7.3.2. Saddle points

Our initial TMCMC investigations revealed two modal regions roughly around $(0.1, -1.1)$ and $(1.1, -1.1)$. For better exploration of the two modal regions, we implemented two separate TMCMC runs beginning at the above two points, and continued Algorithm 1 to ultimately obtain two separate results after $S = 40$ stages at step (2), associated with the two different starting points. Figures 7.4 and 7.5 display the TMCMC trace plots associated with the two different starting points.

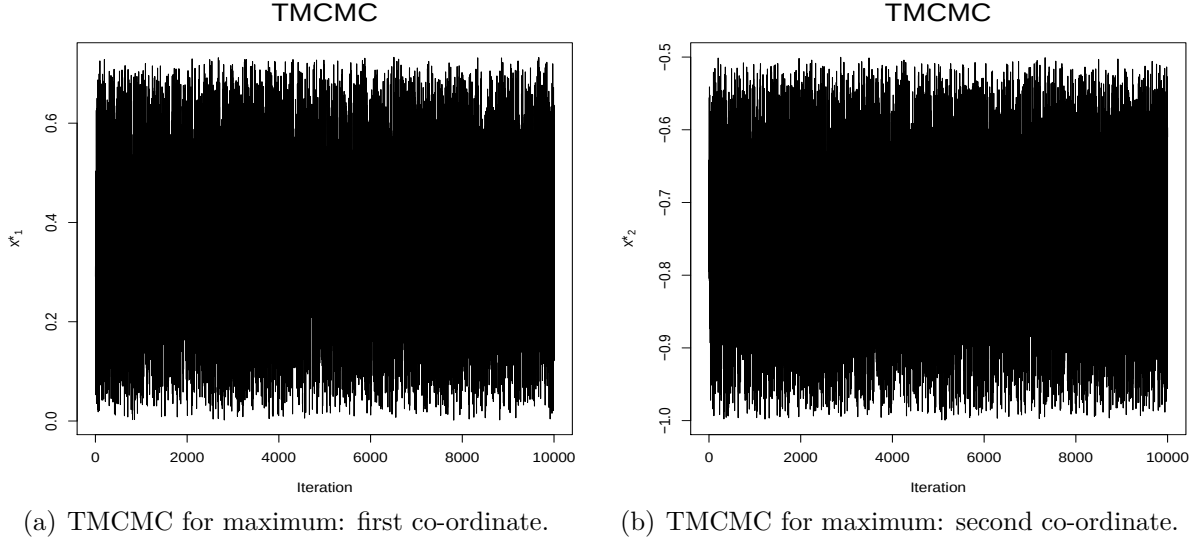


Figure 7.3: TCMC trace plots for Example 3 for finding maxima.

Running step (2) of Algorithm 1 for $S = 40$ steps yields $\hat{\mathbf{x}}_{saddle}^{(1)} = (-0.000309, -0.999580)$ and $\hat{\mathbf{x}}_{saddle}^{(2)} = (0.999433, -0.999915)$ as estimates of two saddle points, which are both reasonably close to the true saddle points.

7.3.3. Inconclusiveness

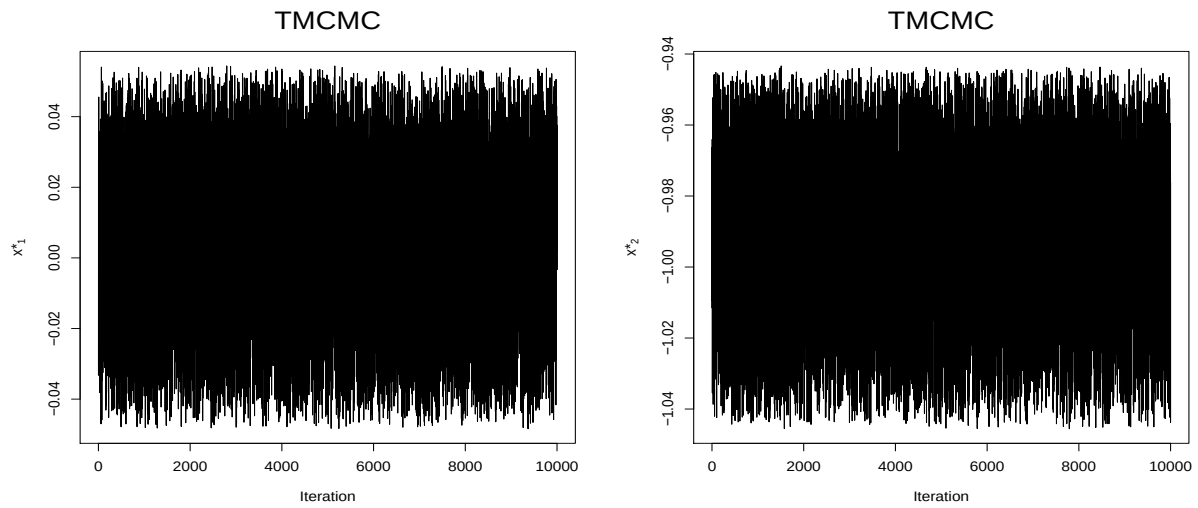
Investigation of situations where $D(a, b) = 0$ for any critical point (a, b) yielded the TCMC trace plots displayed as Figure 7.6. On completion of step (2) of Algorithm 1, we obtain $\hat{\mathbf{x}}_{incon} = (-0.000087, -0.000265)$ as the estimate of the stationary point regarding which conclusion can not be drawn using the second derivatives. Observe that $\hat{\mathbf{x}}_{incon}$ quite adequately estimates the actual point $(0, 0)$ where conclusion fails.

7.3.4. Minimum

Our attempt to implement TCMC with the restrictions $D(\mathbf{x}^*) > 0$ and $f''_{11}(\mathbf{x}^*) > 0$ with $\|\mathbf{f}'(\mathbf{x}^*)\|_2 < \epsilon$ did not yield any acceptance, even for arbitrary initial values. In other words, we could not obtain any solution that satisfies all the above restrictions, and hence conclude that there is no critical point on $\mathcal{X}$ that satisfy the above restrictions.

7.4. Example 4

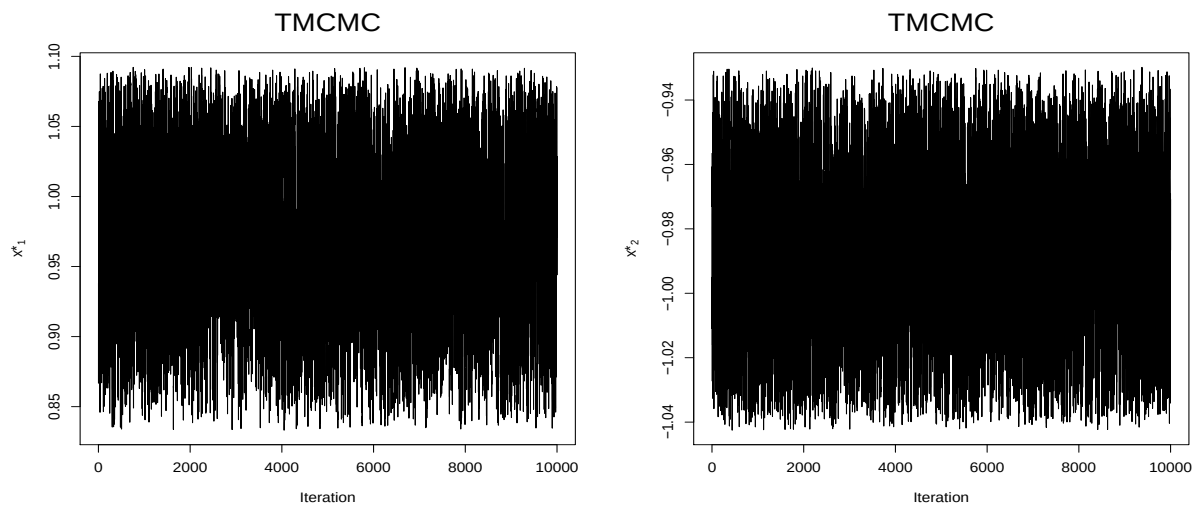
Table 14.2 of Lange (2010) reports quarterly data on AIDS deaths in Australia during 1983 – 1986, which is considered for illustration of fitting Poisson regression model. Specifically, for $i = 1, \dots, 14$, Lange (2010) considers the model $Y_i \sim \text{Poisson}(\lambda_i)$, with $\lambda_i = \exp(\beta_0 + i\beta_1)$. Lange (2010) computed the maximum likelihood estimate (MLE) of $\beta = (\beta_0, \beta_1)$ using Fisher's scoring method, which is equivalent to Newton's method in this



(a) TCMCMC for first saddle point: first coordinate.

(b) TCMCMC for first saddle point: second coordinate.

Figure 7.4: TCMCMC trace plots for Example 3 for finding the first saddle point.



(a) TCMCMC for second saddle point: first coordinate.

(b) TCMCMC for second saddle point: second coordinate.

Figure 7.5: TCMCMC trace plots for Example 3 for finding the second saddle point.

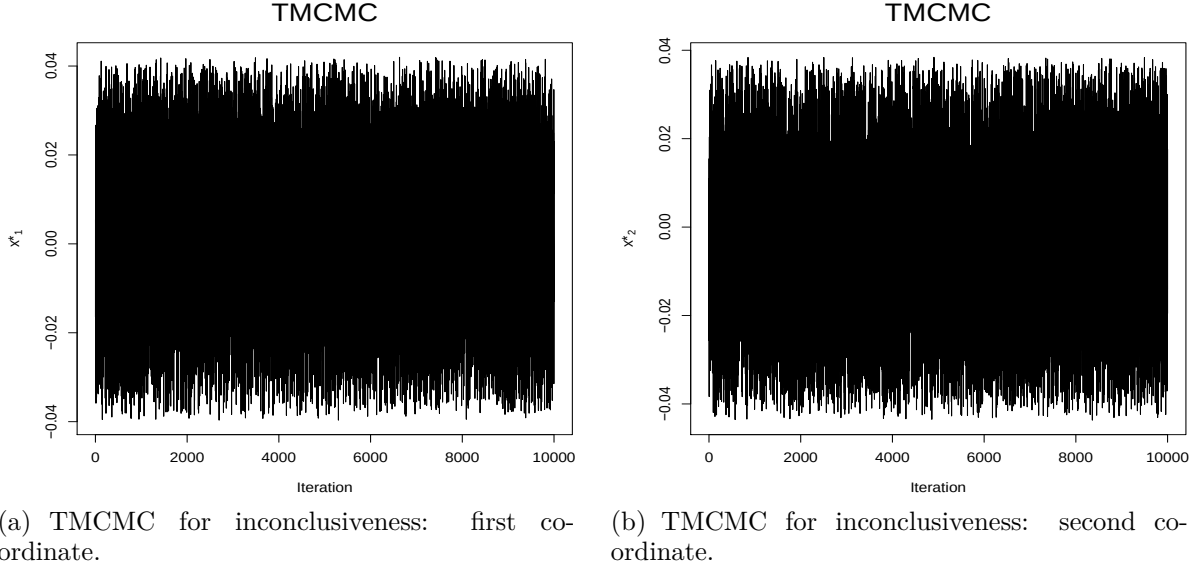


Figure 7.6: TCMC trace plots for Example 3 for investigating inconclusiveness.

case. The final estimate obtained by Lange (2010) is $\hat{\beta}_{MLE} = (0.3396, 0.2565)$.

In our notation, the function to maximize is $f(x_1, x_2) = -\sum_{i=1}^{14} \exp(x_1 + ix_2) + \sum_{i=1}^{14} y_i(x_1 + ix_2)$, with respect to $\mathbf{x} = (x_1, x_2)$. Note that this is a concave maximization problem and hence the second derivative is irrelevant. We thus consider the only constraint $\|\mathbf{f}'(\mathbf{x}^*)\|_2 < \epsilon$ for our implementation, with $\epsilon = 1$. However, additive TCMC did not exhibit adequate mixing properties in this example, and hence we consider a mixture of additive and multiplicative TCMC that is expected to improve mixing by using a mixture of localised moves of additive TCMC and non-local (“random dive”) moves of multiplicative TCMC (see Dutta (2012), Dey and Bhattacharya (2016) for details). We strengthen the mixture TCMC strategy with a further step of a mixture of specialized additive and multiplicative moves, which has parallels with Liu and Sabatti (2000). The detailed TCMC algorithm, for general dimension d , is provided below as Algorithm 2.

In our case, $d = 2$, and we choose $a_j^{(1)} = a_j^{(2)} = 0.05$, for $j = 1, 2$; we also set $p = q = 1/2$. However, in spite of such a sophisticated TCMC algorithm, we failed to achieve excellent mixing in this example, even with long TCMC runs, discarding the first 10^6 iterations and storing every 10-th realization in the next 5×10^6 iterations. The trace plots of all stored 5×10^5 realizations shown in Figure 7.7 indeed demonstrate that the TCMC chain does not have excellent mixing properties. In fact, the trace plots correspond to a reasonable initial value, chosen as the optimum obtained by the Fisher scoring method, which we implemented in R independently of Lange (2010). However, our goal here is to demonstrate that even when TCMC mixing is less than ideal, the estimates obtained by our method can still significantly outperform existing techniques.

Since this is a concave maximization problem, step (2) of Algorithm 1 is unnecessary. Following Remark 8, we set $i^* = \arg \min\{f(\mathbf{x}_i^*) : i = 1, \dots, N\}$ and report $\mathbf{x}_{i^*}^*$ as the approximate maximizer of $f(\cdot)$. With the $N = 5 \times 10^5$ TCMC realizations shown in

Algorithm 2 Mixture TMCMC

(1) Fix $p, q \in (0, 1)$. Set an initial value $\mathbf{x}^{(0)}$.

(2) For $t = 1, \dots, N$, do the following:

1. Generate $U \sim U(0, 1)$.

(a) If $U < p$, then do the following:

- (i) Generate $\varepsilon \sim N(0, 1)$, $b_j \stackrel{iid}{\sim} U(\{-1, 1\})$ for $j = 1, \dots, d$, and set $y_j = x_j^{(t-1)} + b_j a_j^{(1)} |\varepsilon|$, for $j = 1, \dots, d$. Here $a_j^{(1)}$ are positive scaling constants.
- (ii) Evaluate

$$\alpha_1 = \min \left\{ 1, \frac{\pi(\mathbf{y})\pi(\mathbf{g}'(\mathbf{y}) = \mathbf{0} | \mathbf{D}_n, \mathbf{y})}{\pi(\mathbf{x}^{(t-1)})\pi(\mathbf{g}'(\mathbf{x}^{(t-1)}) = \mathbf{0} | \mathbf{D}_n, \mathbf{x}^{(t-1)})} \right\}.$$

- (iii) Set $\tilde{\mathbf{x}}^{(t)} = \mathbf{y}$ with probability α_1 , else set $\tilde{\mathbf{x}}^{(t)} = \mathbf{x}^{(t-1)}$.

(b) If $U \geq p$, then

- (i) Generate $\varepsilon \sim U(-1, 1)$, $b_j \stackrel{iid}{\sim} U(\{-1, 0, 1\})$ for $j = 1, \dots, d$, and set $y_j = x_j^{(t-1)} \varepsilon$ if $b_j = 1$, $y_j = x_j^{(t-1)} / \varepsilon$ if $b_j = -1$ and $y_j = x_j^{(t-1)}$ if $b_j = 0$, for $j = 1, \dots, d$. Calculate $|J| = |\varepsilon|^{\sum_{j=1}^d b_j}$.
- (ii) Evaluate

$$\alpha_2 = \min \left\{ 1, \frac{\pi(\mathbf{y})\pi(\mathbf{g}'(\mathbf{y}) = \mathbf{0} | \mathbf{D}_n, \mathbf{y})}{\pi(\mathbf{x}^{(t-1)})\pi(\mathbf{g}'(\mathbf{x}^{(t-1)}) = \mathbf{0} | \mathbf{D}_n, \mathbf{x}^{(t-1)})} \times |J| \right\}.$$

- (iii) Set $\tilde{\mathbf{x}}^{(t)} = \mathbf{y}$ with probability α_2 , else set $\tilde{\mathbf{x}}^{(t)} = \mathbf{x}^{(t-1)}$.

2. Generate $U \sim U(0, 1)$.

(a) If $U < q$, then do the following

- (i) Generate $\tilde{U} \sim U(0, 1)$ and $\varepsilon \sim N(0, 1)$. If $\tilde{U} < 1/2$, set $y_j = \tilde{x}_j^{(t)} + a_j^{(2)} |\varepsilon|$, for $j = 1, \dots, d$; else, set $y_j = \tilde{x}_j^{(t)} - a_j^{(2)} |\varepsilon|$, for $j = 1, \dots, d$. Here $a_j^{(2)}$ are positive scaling constants.
- (ii) Evaluate

$$\alpha_3 = \min \left\{ 1, \frac{\pi(\mathbf{y})\pi(\mathbf{g}'(\mathbf{y}) = \mathbf{0} | \mathbf{D}_n, \mathbf{y})}{\pi(\tilde{\mathbf{x}}^{(t)})\pi(\mathbf{g}'(\tilde{\mathbf{x}}^{(t)}) = \mathbf{0} | \mathbf{D}_n, \tilde{\mathbf{x}}^{(t)})} \right\}.$$

- (iii) Set $\mathbf{x}^{(t)} = \mathbf{y}$ with probability α_3 , else set $\mathbf{x}^{(t)} = \tilde{\mathbf{x}}^{(t)}$.

(b) If $U \geq q$, then

- (i) Generate $\varepsilon \sim U(-1, 1)$ and $\tilde{U} \sim U(0, 1)$. If $\tilde{U} < 1/2$, set $y_j = \tilde{x}_j^{(t)} \varepsilon$ for $j = 1, \dots, d$ and $|J| = |\varepsilon|^d$, else set $y_j = \tilde{x}_j^{(t)} / \varepsilon$ for $j = 1, \dots, d$ and $|J| = |\varepsilon|^{-d}$.
- (ii) Evaluate

$$\alpha_4 = \min \left\{ 1, \frac{\pi(\mathbf{y})\pi(\mathbf{g}'(\mathbf{y}) = \mathbf{0} | \mathbf{D}_n, \mathbf{y})}{\pi(\tilde{\mathbf{x}}^{(t)})\pi(\mathbf{g}'(\tilde{\mathbf{x}}^{(t)}) = \mathbf{0} | \mathbf{D}_n, \tilde{\mathbf{x}}^{(t)})} \times |J| \right\}.$$

- (iii) Set $\mathbf{x}^{(t)} = \mathbf{y}$ with probability α_4 , else set $\mathbf{x}^{(t)} = \tilde{\mathbf{x}}^{(t)}$.

(3) Store the realizations $\{\mathbf{x}^{(t)}; t = 0, 1, \dots, N\}$ for Bayesian inference.

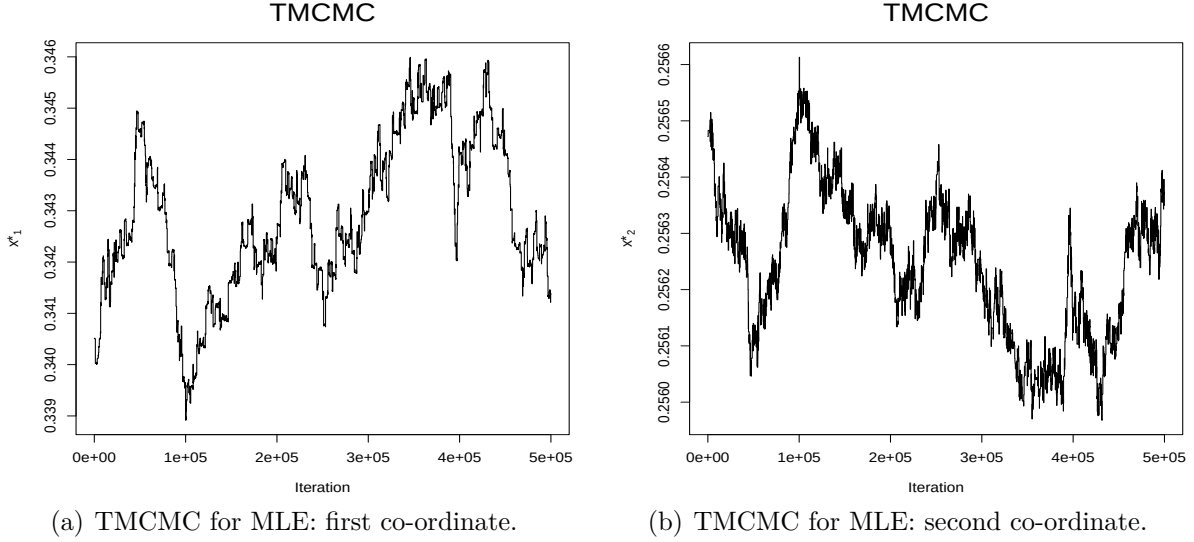


Figure 7.7: TCMC trace plots for Example 4 for finding MLE.

Figure 7.7, we obtain the estimate $\hat{\mathbf{x}}_{\text{MLE}} = (0.339627, 0.256524)$, for which $\|\mathbf{f}'(\hat{\mathbf{x}}_{\text{MLE}})\|_2 = 0.005108$. In contrast, the MLE obtained by Fisher's scoring (reported to four digits by Lange (2010)) is $\hat{\beta}_{\text{MLE}} = (0.339634, 0.256524)$, with a gradient norm of 0.011791 — substantially farther from zero.

We also compared our approach with other popular optimization methods implemented in R: the BFGS (Broyden–Fletcher–Goldfarb–Shanno) method, simulated annealing followed by BFGS refinement, and a multi-start quasi-Newton method with 10 random starts uniformly sampled from $[-2, 2]^2$. The BFGS and simulated annealing results were obtained using R's `optim` function with default settings. For the multi-start quasi-Newton method, we report the mean of the estimates from the 10 runs; we deliberately did not select the best solution because the other competing methods rely on single starts only. Thus, the multi-start results are comparable to an average, not to a minimum-norm solution over multiple starts.

Up to the first six decimal places, Fisher's scoring, BFGS, and simulated annealing produced identical optima. Table 7.1 shows that in every case our Bayesian procedure achieves a substantially smaller gradient norm. All methods completed in a few seconds, so we do not report run times; gradient norms were computed analytically.

Thus, our Bayesian approach proves more reliable than existing methods. This may be a general phenomenon: by starting from good initial values (for instance, optima obtained by popular methods), TCMC can explore regions of $\mathcal{X}$ that almost surely contain points with smaller gradient norms than those found by the initial method. It is also encouraging that for this example, TCMC took well under a minute to generate 6×10^6 iterations on a single processor of our VMWare system.

7.5. Example 5

Hunter and Lange (2000) refer to a nonlinear optimization problem of the form $Y_i \sim$

Table 7.1: Comparison of optimization methods for Example 4 (Poisson regression MLE). The true gradient norm at the optimum is zero.

Method	$\hat{\beta}_0$	$\hat{\beta}_1$	$\ \nabla f\ _2$	$f(\hat{\beta}_0, \hat{\beta}_1)$
Fisher's scoring	0.339634	0.256524	0.011791	472.062548
BFGS	0.339634	0.256524	0.011791	472.062548
Simulated annealing	0.339634	0.256524	0.011791	472.062548
Multi-start quasi-Newton (10 starts)	0.339850	0.256501	0.123481	472.062547
Our method (TMCMC + posterior)	0.339627	0.256524	0.005108	472.062548

Our method achieves the smallest gradient norm, indicating closer proximity to the true MLE.

$N(\mu_i, \sigma^2)$ for $i = 1, \dots, m$, where

$$\mu_i = \sum_{j=1}^d \left[\exp(-z_{ij}\theta_j^2) + z_{ij}\theta_{d-j+1} \right];$$

z_{ij} being the i -th observation of the j -th covariate, for $i = 1, \dots, m$ and $j = 1, \dots, d$. The goal is to compute the MLE of $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)$, assuming that σ is known. In our notation, the objective is to minimize

$$f(\mathbf{x}) = \sum_{i=1}^m \left(y_i - \sum_{j=1}^d \left[\exp(-z_{ij}x_j^2) + z_{ij}x_{d-j+1} \right] \right)^2,$$

with respect to $\mathbf{x}$.

We consider 5 simulation experiments in this regard for $d = 2, 5, 10, 50, 100$. In each case we generate $\theta_{0j} \sim U(-1, 1)$ independently for $j = 1, \dots, d$, $z_{ij} \sim N(0, 1)$ independently, for $i = 1, \dots, m$ and $j = 1, \dots, d$, and set, for $i = 1, \dots, m$, $\mu_{0i} = \sum_{j=1}^d \left[\exp(-z_{ij}\theta_{0j}^2) + z_{ij}\theta_{0,d-j+1} \right]$. We finally generate the response data by simulating $Y_i \sim N(\mu_{0i}, \sigma_0^2)$ independently for $i = 1, \dots, m$, where we set $\sigma_0^2 = 0.1$. For $d = 2, 5, 10, 50, 100$, we generate datasets of sizes $m = 10, 10, 20, 75, 200$.

Note that this is not a convex minimization problem and the matrix of second derivatives $\boldsymbol{\Sigma}''(\cdot)$ plays an important role, along with $\|\mathbf{f}'(\cdot)\|_d$. Thus it is important to check positive definiteness of $\boldsymbol{\Sigma}''(\cdot)$ for any dimension d . We use the LAPACK library function “*dpotrf*” for Cholesky decomposition of $\boldsymbol{\Sigma}''(\cdot)$, which contains a parameter “*info*”. Given any $\mathbf{x}$, *info* = 0 indicates positive definiteness of $\boldsymbol{\Sigma}''(\mathbf{x})$, while other values of *info* rules out positive definiteness. Note that since this is not a convex minimization problem, step (2) of Algorithm 1 is necessary, unlike in Example 4.

We implement the mixture TMCMC algorithm 2 for all values of d , with $p = q = 1/2$ and set $a_j^{(1)} = a_j^{(2)} = 0.05$ for $j = 1, \dots, d$.

7.5.1. Case 1: $d = 2$

In the TMCMC step, we set $\epsilon = 1$ in the restriction $\|\mathbf{f}'(\cdot)\|_2 < \epsilon$. The trace plots, shown in Figure 7.8, exhibit adequate mixing. Running step (2) of Algorithm 1 till $S = 40$

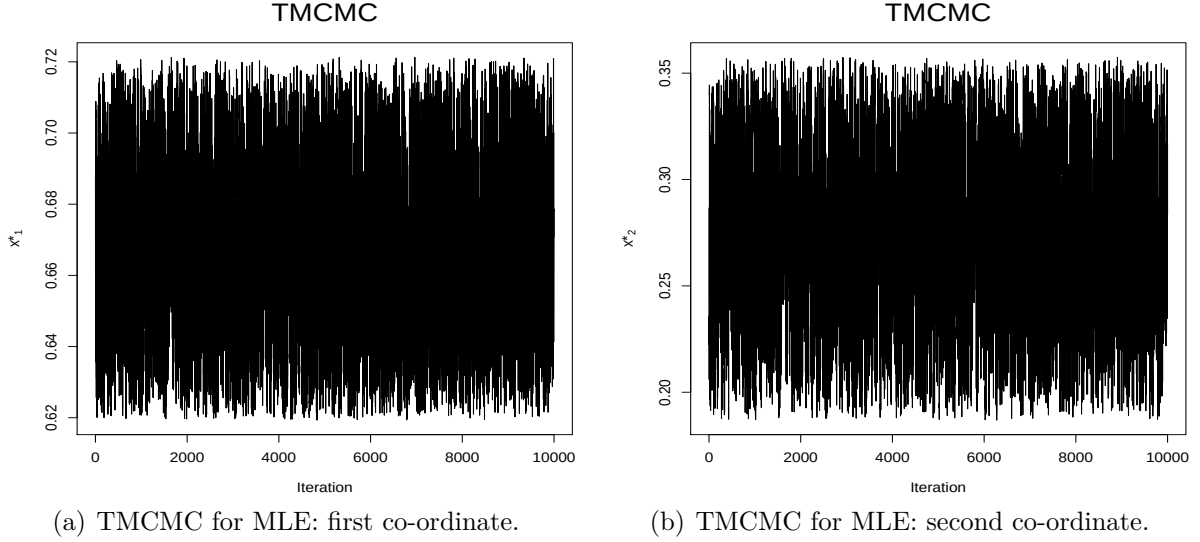


Figure 7.8: TCMC trace plots for Example 5 for finding MLE for dimension $d = 2$.

stages with $\eta_k = 1/(10 + k - 1)$ for $k = 1, \dots, S$, yielded the estimate of the MLE to be $\hat{\mathbf{x}}_{MLE} = (0.678854, 0.293575)$, for which $\|\mathbf{f}'(\hat{\mathbf{x}}_{MLE})\|_2 = 0.012514$. The exercise takes 3 minutes on our VMWare, implemented in parallel on 100 cores.

However, examination of the samples obtained by importance resampling at the different stages of step (2) of Algorithm 1 did not reveal any evidence of multimodality, and hence it is pertinent to consider that estimate which corresponds to the minimum of $\{\|\mathbf{f}'(\mathbf{x}_i^*)\|_2 : i = 1, \dots, N\}$, where $\mathbf{x}_i^*$ are the original TCMC samples, with $N = 50000$. As such, we modify the previous estimate to $\hat{\mathbf{x}}_{MLE} = (0.678912, 0.293809)$, which yields $\|\mathbf{f}'(\hat{\mathbf{x}}_{MLE})\|_2 = 0.007221$, which is somewhat closer to zero compared to that for the previous estimate. Note however, that the two estimates of MLE are quite close to each other.

7.5.2. Case 2: $d = 5$

Here, for the 5-dimensional TCMC, we set $\epsilon = 3$ in the restriction $\|\mathbf{f}'(\cdot)\|_5 < \epsilon$, as smaller values of ϵ led to poor convergence. Note that the Euclidean norm increases with dimension (see, for example, Giraud (2015)), and so it is a natural requirement to increase ϵ as dimension increases. Similarly, we had to increase η_k to $\eta_k = 1.5/\log(10 + k)$ for implementing step (2) of Algorithm 1. The rest of the parameters of Algorithm 2 remain the same as for $d = 2$.

The trace plots exhibited reasonable mixing; those for the first and the last (5-th) co-ordinate of $\mathbf{x}^*$ are depicted in Figure 7.9. Running step (2) of Algorithm 1 till $S = 40$ stages we obtain $\hat{\mathbf{x}}_{MLE} = (0.663327, 0.431669, 0.045598, 0.239091, 0.301665)$, which corresponds to $\min\{\|\mathbf{f}'(\mathbf{x}_i^*)\|_5 : i = 1, \dots, N\} = 0.254217$, with $N = 50000$. Note the increase in $\|\mathbf{f}'(\cdot)\|_5$ compared to those of the smaller dimensions. Again, this is clearly to be expected because of the curse of dimensionality. The exercise takes 4 minutes to complete on our VMWare.

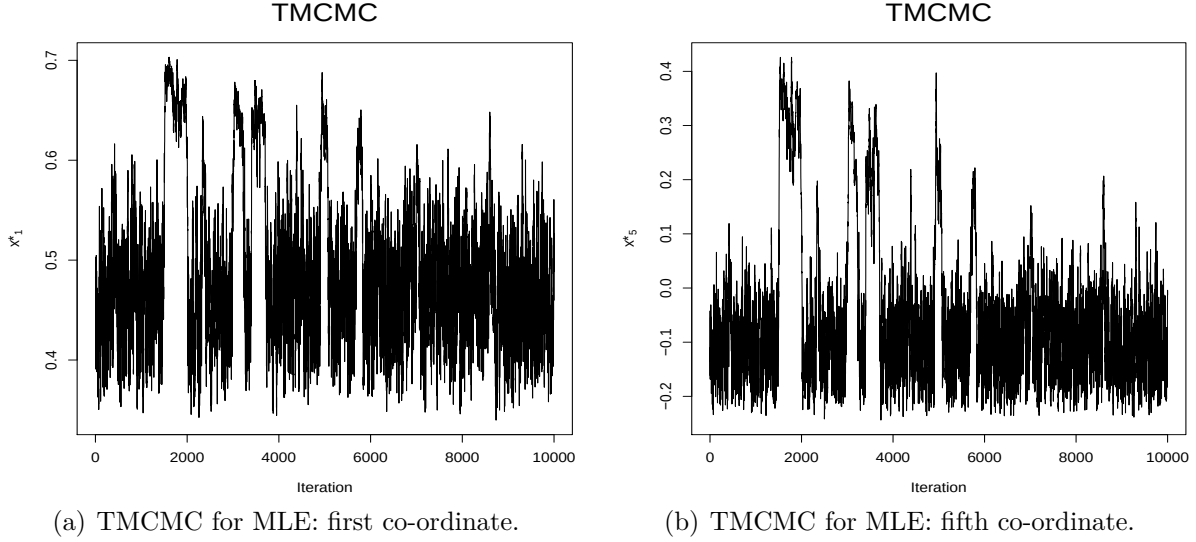


Figure 7.9: TCMCM trace plots for Example 5 for finding MLE for dimension $d = 5$.

7.5.3. Case 3: $d = 10$

Here, for $d = 10$, we had to set $\epsilon = 6$ for the restriction $\|\mathbf{f}'(\cdot)\|_{10} < \epsilon$ in TCMCM for adequate convergence. We also set $\eta_k = 7/\log(10 + k - 1)$ for implementing step (2) of Algorithm 1.

Our investigation shows that the mixing of TCMCM is not inadequate. The trace plots of the first and the last co-ordinate of $\mathbf{x}^*$ shown in Figure 7.10 also bear evidence to this. After implementing step (2) of Algorithm 1 till $S = 40$ stages, we obtain $\hat{\mathbf{x}}_{MLE} = (0.472309, 0.10124, 0.079194, 0.108072, 0.096287, -0.511566, 0.517567, -0.887624, 0.637796, -0.317721)$ with $\|\mathbf{f}'(\hat{\mathbf{x}}_{MLE})\|_{10} = 1.917746$. On the other hand, $\min\{\|\mathbf{f}'(\mathbf{x}_i^*)\|_{10} : i = 1, \dots, 50000\} = 1.902788$, which corresponds to $\hat{\mathbf{x}}_{MLE} = (0.471200, 0.100131, 0.078085, 0.101934, 0.095178, -0.504813, 0.516459, -0.888733, 0.631659, -0.323859)$. Thus, both the estimates as well as the corresponding gradients are quite close to each other. Again note the increase in $\|\mathbf{f}'(\cdot)\|_{10}$ compared to those of the smaller dimensions. The entire exercise takes 17 minutes on our VMWare to complete.

7.5.4. Case 4: $d = 50$

For this somewhat large dimension, we had to set $\epsilon = 100$ and $\eta_k = 200/\log(10+k-1)$. Figure 7.11, which displays all stored 50000 TCMCM realizations for x_1^* and x_{50}^* do not indicate excellent mixing, in spite of the sophistication of Algorithm 2. However, due to the high-dimension and complexity of the posterior this is not unexpected. As demonstrated by Example 4, we can still expect to get closer to the MLE compared to other optimization methods. In this case, we obtain $\min\{\|\mathbf{f}'(\mathbf{x}_i^*)\|_{50} : i = 1, \dots, 50000\} = 73.05261$ while step (2) of Algorithm 1 implemented for $S = 100$ stages, of which only the first four stages consisted of positive n_k , yielded $\|\mathbf{f}'(\hat{\mathbf{x}}_{MLE})\|_{50} = 76.28244$. Thus, the norms of the gradients have increased considerably in this high dimension, compared to the previous $d = 2, 5, 10$. Given the high dimension in this problem, the above gradient values 73.05261 and 76.28244

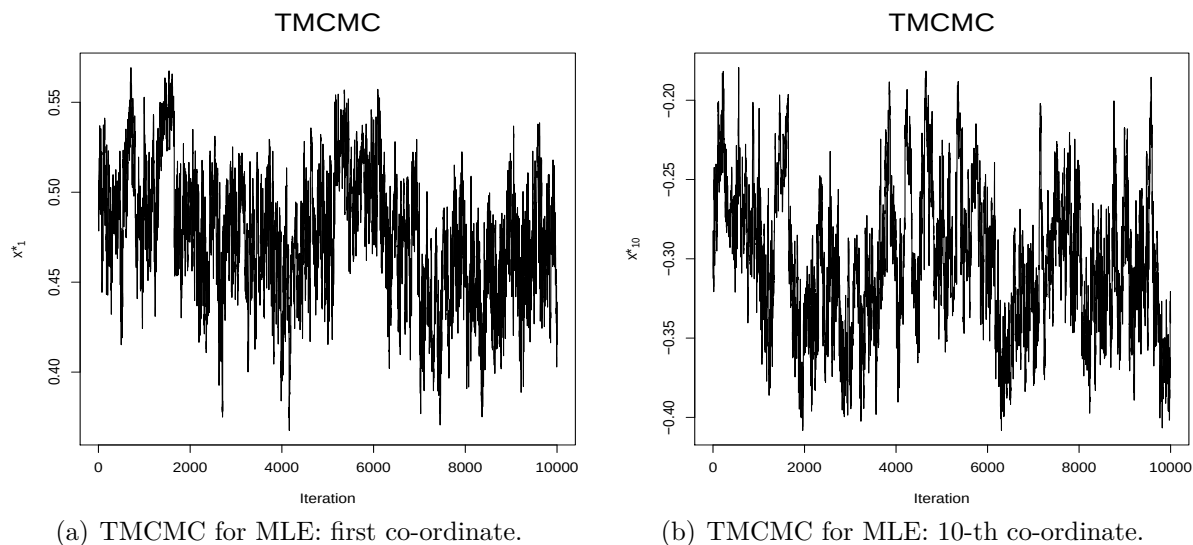


Figure 7.10: TCMC trace plots for Example 5 for finding MLE for dimension $d = 10$.

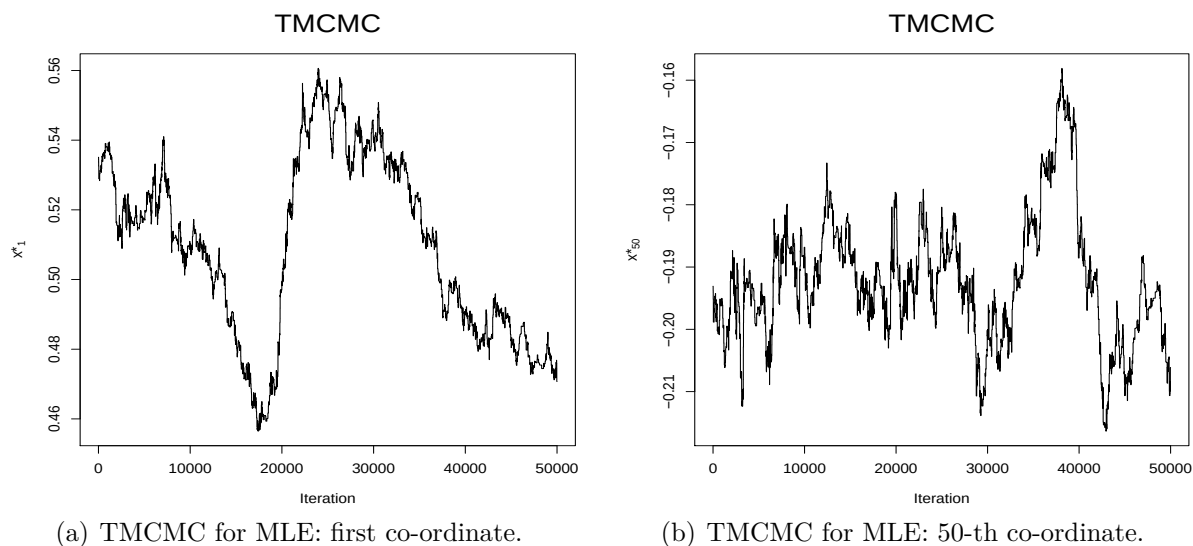


Figure 7.11: TCMC trace plots for Example 5 for finding MLE for dimension $d = 50$.

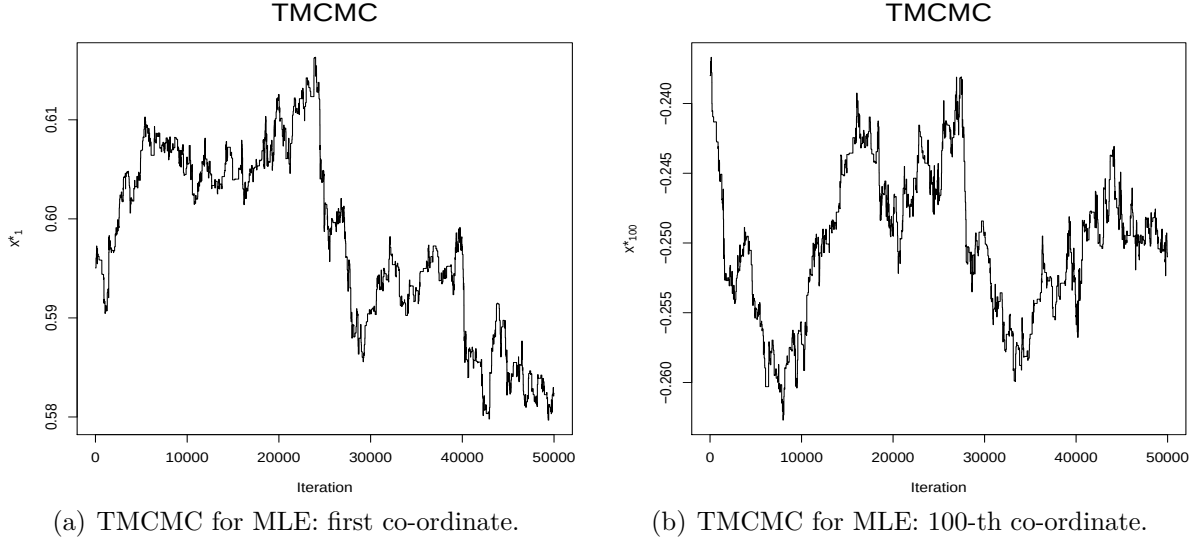


Figure 7.12: TCMC trace plots for Example 5 for finding MLE for dimension $d = 100$.

are quite close.

The TCMC implementation took about 12 hours on a single core in our VMWare and step (2) of Algorithm 1 took additional 2 hours on 100 cores.

7.5.5. Case 5: $d = 100$

The curse of dimensionality now forced us to set $\epsilon = 400$ for TCMC convergence and $\eta_k = 850/\log(10 + k - 1)$. It took around 15 hours to complete the TCMC run; the trace plots shown in Figure 7.12 do not bear evidence of non-convergence, even though mixing is expectedly inadequate in such high dimension. Here we obtain $\min\{\|\mathbf{f}'(\mathbf{x}_i^*)\|_{50} : i = 1, \dots, 50000\} = 330.4483$, while step (2) of Algorithm 1 implemented for till $S = 100$ stages, of which only the first three stages yielded positive n_k , produced $\hat{\mathbf{x}}_{MLE}$ such that $\|\mathbf{f}'(\hat{\mathbf{x}}_{MLE})\|_{100} = 341.2009$. Given the high dimension, this value is quite close to the above minimum gradient. Implementation of step (2) of Algorithm 1 took 2 hours 36 minutes on 100 cores on our VMWare.

8. Summary and conclusion

8.1. The key ideas in a nutshell

In this article, we have proposed and developed a novel Bayesian algorithm for general function optimization that judiciously exploits the derivatives of the objective function in conjunction with the posterior Gaussian derivative process, given data consisting of input points from the function domain and their corresponding function evaluations. The posterior simulation approach inherent to our method ensures improved accuracy of the obtained solutions compared to existing optimization algorithms. Another important feature is that, for any desired degree of accuracy, Bayesian credible regions of the optima are readily available at any stage of the algorithm.

Under an appropriate fixed-domain infill asymptotics setup, we prove that the algorithm converges almost surely to the true optima. Along the way, we establish almost sure uniform convergence of the posteriors corresponding to the Gaussian process and its derivative process to the objective function and its derivatives, and we provide rates of convergence for a specific infill design. Furthermore, we present a Bayesian characterization of the number of optima of the objective function by exploiting the information contained in our algorithm.

8.2. High-dimensional and mixing issues – i.i.d. sampling as an alternative to TMCMC

The application of our Bayesian optimization algorithm to a variety of examples, ranging from simple to challenging, has produced encouraging and insightful results. The choice of initial values for the TMCMC sampler can affect mixing; however, as argued in the paper, the optima obtained by good existing optimization algorithms can serve as effective initial values for TMCMC. Moreover, we have shown that even when mixing is less than ideal, our Bayesian algorithm can still significantly outperform popular optimization methods. Consequently, the TMCMC mixing issue may not be critical, at least for low-dimensional problems.

Dimensionality poses a much more serious challenge. Our experimental results demonstrate that as the dimension increases, the accuracy of the algorithm deteriorates. High dimensionality also severely impairs TMCMC mixing by excessively restricting the input space through the prior constraints. The reason is that the prior requires the gradient norm to be small and the Hessian positive definite; in high dimensions, the region of the parameter space that satisfies these conditions is extremely small, making it difficult for a Markov chain to explore effectively.

Fortunately, the TMCMC mixing problem can be completely circumvented by replacing TMCMC with the general i.i.d. sampling procedure developed by Bhattacharya (2025) (see also Bhattacharya (2021b), Bhattacharya (2021a), Bhattacharya (2022), Bhattacharya *et al.* (2025) for extensions). Using i.i.d. sampling, sampling accuracy can be maintained almost perfectly even as the dimension increases, which is expected to dramatically improve the quality of the optima obtained by our method. This approach eliminates the need for careful tuning of proposal distributions and the risk of poor mixing, making it particularly attractive for high-dimensional optimization problems.

8.3. Towards black-box optimization: a natural extension

Although the current work assumes that the first and second partial derivatives of the objective function are available, a natural and promising extension is to adapt the algorithm to settings where f is a black box – only function evaluations are possible, and derivatives are unavailable. In this case, one can replace the true derivatives $\mathbf{f}'$ and Σ'' in the prior and the refinement step with their Gaussian process surrogates $\mathbf{g}'_n$ and the GP-based Hessian. Because Theorems 2 and 3 establish that $\mathbf{g}'_n$ converges uniformly to $\mathbf{f}'$ as the dataset grows, the modified algorithm would still converge to the true stationary points of f . The refinement condition $\|\mathbf{f}'(\mathbf{x})\|_d < \eta_k$ would be replaced by $\|\mathbf{g}'_n(\mathbf{x})\|_d < \eta_k$, and the Hessian check would use the GP posterior Hessian. This adaptation turns our derivative-based method into a

general-purpose, provably convergent black-box optimization algorithm – a rare combination in the Bayesian optimization literature. We intend to pursue this direction in future work, providing rigorous convergence guarantees and demonstrating its performance on real-world black-box problems.

8.4. Discussion of scalability and potential extensions

The cubic complexity $O(n^3)$ of the Gaussian process posterior, which arises from the inversion of the $n \times n$ correlation matrix Σ_{22} , limits the method to moderate dataset sizes n (typically $n \leq 500$ in our experiments). For high-dimensional problems, the number of refinement stages k is small and the dataset size n remains modest. To scale the method to very large input dimensions or very large dataset sizes, several extensions could be explored.

One promising direction is the use of sparse Gaussian process approximations, such as the fully independent training conditional (FITC) method of Quiñero-Candela and Rasmussen (2005). By introducing a small set of m inducing points with $m \ll n$, the complexity of the GP posterior reduces from $O(n^3)$ to $O(nm^2)$, making it feasible to handle larger datasets.

Another approach is to employ additive kernels, where the covariance function is assumed to decompose as a sum of one-dimensional kernels: $c(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^d c_k(x_k, y_k)$. This additive structure allows separate Gaussian process modeling per dimension, effectively reducing the dimensionality of the problem and enabling more efficient computation; see Duvenaud *et al.* (2011).

Random Fourier features offer an alternative that approximates the covariance kernel using a finite set of random basis functions; see Rahimi and Recht (2007). This approximation yields linear complexity in the number of data points n and is particularly suitable for high-dimensional problems, as the number of features needed to achieve a given accuracy does not depend exponentially on d .

Finally, if the covariance kernel is separable, the posterior of the derivative process factorizes across dimensions. This factorization can be exploited for parallel or distributed computation, where each dimension is processed independently on a separate processor, leading to significant speedups in high dimensions.

These extensions, together with the black-box adaptation and i.i.d. sampling, constitute a rich agenda for future research. They would enable the application of our Bayesian optimization method to very high-dimensional optimization problems and to problems with large numbers of data points, while retaining the theoretical guarantees that are the hallmark of our approach.

Acknowledgment

We sincerely thank the reviewer whose comments have led to significant enhancement of the quality of our manuscript.

Conflict of interest

The authors declare no conflicts of interest.

Data and code availability statement

Our MPI-based parallel C code for implementing the AIDS data problem is available at https://github.com/Sourabh-Bhattacharya/FUNCTION_OPT_GDP. The data is contained in the code itself, and is also available in Lange (2010).

A. Computational complexity of Algorithm 1

A.1. Preliminary notation

- d – dimension of the input space $\mathcal{X}$.
- n_0 – size of the initial dataset $\mathcal{D}_{n_0}$.
- $N_{\text{TMC MC}}$ – number of TMC MC iterations performed in step (1) (including burn-in and thinning).
- N – number of stored samples after thinning (for example, $N = 50000$).
- M – number of points resampled at each stage of step (2) (for example, $M = 1000$).
- K – total number of refinement stages.
- L – maximum number of points augmented per stage (typically $L = 5$).
- n_k – number of points actually augmented at stage k ($0 \leq n_k \leq L$).
- N_k – dataset size before stage k ; $N_1 = n_0$, $N_{k+1} = N_k + n_k$.
- $N_{\max} = \max_k N_k \leq n_0 + LK$.
- $K_{\text{aug}} = \#\{k : n_k > 0\}$ – number of stages where augmentation occurs.

A.2. Assumptions

- (i) *Dense linear algebra.* All matrix operations (multiplication, inversion, Cholesky decomposition) use standard algorithms. Multiplication of a $p \times q$ matrix by a $q \times r$ matrix costs $O(pqr)$; inversion of an $m \times m$ matrix costs $O(m^3)$; Cholesky decomposition costs $O(m^3)$.
- (ii) *Analytical derivatives.* The first and second partial derivatives of the objective function f are available in closed form. Evaluating the gradient $\mathbf{f}'(\mathbf{x})$ costs $O(d)$, assembling the Hessian $\Sigma''(\mathbf{x})$ costs $O(d^2)$, and testing positive definiteness via Cholesky decomposition costs $O(d^3)$.
- (iii) *Dataset augmentation.* When new points are added to the dataset, the correlation matrix Σ_{22} and its inverse are recomputed from scratch (no low-rank updates).

- (iv) *TMCMC implementation.* In each TMCMC iteration, the density of the current point is stored. Hence only one density evaluation (for the proposal) is performed per iteration. We assume additive TMCMC for simplicity.

A.3. Cost of one posterior density evaluation

A.3.1. Precomputed quantities

Before the TMCMC loop, the following matrices are computed once:

$$\mathbf{P} = \left(\mathbf{H}^T \Sigma_{22}^{-1} \mathbf{H} + \Sigma_0^{-1} \right)^{-1},$$

and the vector $\mathbf{v} = \Sigma_{22}^{-1}(\mathbf{f}_n - \mathbf{H}\hat{\boldsymbol{\beta}})$, where $\hat{\boldsymbol{\beta}}$ is the posterior mean of $\boldsymbol{\beta}$. These quantities do not depend on the candidate point $\mathbf{x}$ and are reused throughout.

Lemma 2: For a fixed dataset $\mathcal{D}_n$ of size n and a point $\mathbf{x} \in \mathcal{X}$, the computation of

$$\pi(\mathbf{g}''(\mathbf{x}) = \mathbf{0} \mid \mathcal{D}_n, \mathbf{x})$$

requires $O(dn^2 + d^2n + d^3)$ floating-point operations, assuming that all quantities that do not depend on $\mathbf{x}$ (for example, Σ_{22}^{-1} , $\mathbf{P}$, $\mathbf{A}\hat{\boldsymbol{\beta}}$) have been precomputed.

Proof: The evaluation follows equations 22, 19, 20, 8 and 9 of the main paper. We list the steps and their costs.

- (1) Compute $\Sigma_{12}(\mathbf{x})$ ($d \times n$). Each column requires $O(d)$ work $\implies O(dn)$.
- (2) Compute $\hat{\boldsymbol{\mu}}'(\mathbf{x}) = \mathbf{A}\hat{\boldsymbol{\beta}} + \Sigma_{12}(\mathbf{x})\mathbf{v}$, where $\mathbf{v}$ is a precomputed n -vector. $\Sigma_{12}\mathbf{v}$ costs $O(dn)$; addition of the constant $O(d)$.
- (3) Compute $\mathbf{M} = \Sigma_{12}(\mathbf{x})\Sigma_{22}^{-1}$: $d \times n$ times $n \times n \implies O(dn^2)$.
- (4) Compute $\tilde{\boldsymbol{\Sigma}} = \Sigma_{11} - \mathbf{M}\Sigma_{12}(\mathbf{x})^T$: $\mathbf{M}\Sigma_{12}^T$ is $d \times n$ times $n \times d \implies O(d^2n)$; subtraction $O(d^2)$.
- (5) Compute $\mathbf{Q} = \mathbf{A} - \mathbf{M}\mathbf{H}$: $\mathbf{M}\mathbf{H}$ is $d \times n$ times $n \times (d+1) \implies O(d^2n)$; subtraction $O(d^2)$.
- (6) Compute $\mathbf{Q}\mathbf{P}\mathbf{Q}^T$: $\mathbf{Q}\mathbf{P}$: $d \times (d+1)$ times $(d+1) \times (d+1) \implies O(d^3)$. Then $(\mathbf{Q}\mathbf{P})\mathbf{Q}^T$: $d \times (d+1)$ times $(d+1) \times d \implies O(d^3)$. Add to $\tilde{\boldsymbol{\Sigma}}$: $O(d^2)$.
- (7) Invert $\hat{\boldsymbol{\Sigma}}$ (size $d \times d$): $O(d^3)$ (Cholesky or LU).
- (8) Compute the quadratic form $(\hat{\boldsymbol{\mu}}')^T \hat{\boldsymbol{\Sigma}}^{-1} \hat{\boldsymbol{\mu}}'$: matrix-vector product $O(d^2)$, dot product $O(d)$.
- (9) Combine with the precomputed constant denominator: $O(1)$.

Summing the dominating terms gives $O(dn^2 + d^2n + d^3)$. □

A.4. Cost of one TMCMC iteration (step (1))

Lemma 3: Each iteration of the TMCMC loop executed in step (1) of Algorithm 1 has complexity

$$O(d^3 + d^2 n_0 + d n_0^2).$$

Proof: An iteration proposes a new point $\mathbf{y}$ (cost $O(d)$), evaluates its density using Lemma 2 with $n = n_0$ (cost as above), and performs an acceptance test (constant time). The density of the current point is stored from the previous iteration. Hence the total cost is dominated by the density evaluation. $\square$

A.5. Cost of one stage of the refinement loop (step (2))

Lemma 4: Consider stage k of step (2), where before the stage the dataset size is N_k (with $N_1 = n_0$) and after augmentation it becomes $N_{k+1} = N_k + n_k$ ($0 \leq n_k \leq L$). The following are available from the previous stage: all precomputed matrices for dataset size N_k , and the stored denominator densities $\pi(\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0} \mid \mathcal{D}_{N_k}, \mathbf{x}_i^*)$ for all stored samples $\mathbf{x}_i^*$. Then the computational cost of stage k is

$$\begin{aligned} C_k = & O\left(N(d^3 + d^2 N_k + d N_k^2) \quad (\text{density evaluations for weights})\right. \\ & + N \quad (\text{weight multiplication and storage}) \\ & + (N + M) \quad (\text{resampling}) \\ & + M d^3 \quad (\text{gradient/Hessian checks}) \\ & \left. + \mathbf{1}_{\{n_k > 0\}}(N_{k+1}^3 + d^3 + d^2 N_{k+1} + n_k C_f) \quad (\text{augmentation})\right), \end{aligned}$$

where N is the number of stored samples, M the resample size, and C_f the cost of evaluating the objective function f at a single point.

Proof: The work is broken down as follows.

- (1) *Weight update (density evaluation).* For each of the N stored samples we need the new density $\pi(\mathbf{g}'(\mathbf{x}_i^*) = \mathbf{0} \mid \mathcal{D}_{N_k}, \mathbf{x}_i^*)$ (numerator). By Lemma 2 this costs $N \cdot C_{\text{eval}}(N_k) = O(N(d^3 + d^2 N_k + d N_k^2))$.
- (2) *Weight update (multiplication).* Each new density is multiplied by the stored previous weight $w_{k-1}(\mathbf{x}_i^*)$ to obtain $w_k(\mathbf{x}_i^*)$, which is then stored. This costs $O(N)$.
- (3) *Resampling.* To draw M distinct indices without replacement with probabilities proportional to the normalized weights $w_k(\mathbf{x}_i^*)$, we can use the alias method (Walker (1974, 1977); Vose (1991)) or systematic resampling (Kitagawa (1996); Carpenter *et al.* (1999)). Both methods require $O(N)$ preprocessing: building the alias table for the alias method or computing cumulative sums for systematic resampling. After preprocessing, each sample can be drawn in $O(1)$ time using the alias method, while systematic resampling draws all M samples in $O(M)$ time. Consequently, the total cost of resampling is $O(N + M)$.

- (4) *Gradient and Hessian checks.* For each of the M resampled points, we compute $\|\mathbf{f}'(\mathbf{x})\|_d$ ($O(d)$), assemble the Hessian $\Sigma''(\mathbf{x})$ ($O(d^2)$), and perform Cholesky decomposition to test positive definiteness ($O(d^3)$). Total $O(Md^3)$.
- (5) *Augmentation (if $n_k > 0$).* We evaluate $f(\mathbf{x})$ for the n_k points that satisfy the gradient condition. The cost per evaluation is C_f , so total $O(n_k C_f)$. Then we recompute all precomputed quantities for the new dataset size N_{k+1} : forming Σ_{22} ($O(N_{k+1}^2)$) and its inverse ($O(N_{k+1}^3)$); recomputing $\mathbf{P} = (\mathbf{H}^T \Sigma_{22}^{-1} \mathbf{H} + \Sigma_0^{-1})^{-1}$ ($O(d^3 + d^2 N_{k+1})$); and other auxiliary matrices ($O(d^2 N_{k+1})$). The dominant cost is $O(n_k C_f + N_{k+1}^3 + d^3 + d^2 N_{k+1})$.

Summing all contributions yields the stated bound. $\square$

Total complexity of Algorithm 1

Theorem 7: Let Algorithm 1 be run with initial dataset size n_0 , N_{TMCMC} TMCMC iterations, N stored samples, resample size M , at most K refinement stages, and at most L points augmented per stage. Let C_f denote the cost of evaluating f at a single point. Then the total time complexity is

$$\begin{aligned}
T_{\text{total}} = & O\left(n_0^3 + d^3 + d^2 n_0 \quad (\text{precomputation})\right. \\
& + N_{\text{TMCMC}}(d^3 + d^2 n_0 + d n_0^2) \quad (\text{TMCMC}) \\
& + \sum_{k=1}^K N(d^3 + d^2 N_k + d N_k^2) \quad (\text{density evaluations for weights}) \\
& + K N \quad (\text{weight multiplication}) \\
& + K(N + M) \quad (\text{resampling}) \\
& + K M d^3 \quad (\text{gradient/Hessian checks}) \\
& \left. + \sum_{k: n_k > 0} (N_{k+1}^3 + d^3 + d^2 N_{k+1} + n_k C_f) \quad (\text{augmentation})\right),
\end{aligned}$$

where $N_1 = n_0$, $N_{k+1} = N_k + n_k$, and $0 \leq n_k \leq L$.

Since the maximum dataset size $N_{\text{max}} = \max_k N_k \leq n_0 + LK$, the total complexity simplifies to

$$\begin{aligned}
T_{\text{total}} = & O\left(N_{\text{TMCMC}}(d^3 + d^2 n_0 + d n_0^2) + K N(d^3 + d^2 N_{\text{max}} + d N_{\text{max}}^2)\right. \\
& \left. + K(N + M d^3) + K_{\text{aug}}(N_{\text{max}}^3 + L C_f)\right),
\end{aligned}$$

where $K_{\text{aug}} = \#\{k : n_k > 0\} \leq K$.

Proof: The total cost is the sum of:

- (1) *Precomputation (once).* Form Σ_{22} ($O(n_0^2)$), its inverse ($O(n_0^3)$), $\mathbf{P}$ ($O(d^3 + d^2 n_0)$), and other constant-time operations. Hence $O(n_0^3 + d^3 + d^2 n_0)$.

- (2) *Step 1 (TMCMC)*: N_{TMCMC} iterations, each costing $O(d^3 + d^2 n_0 + d n_0^2)$ by Lemma 3.
- (3) *Step 2*: sum of the costs of K stages given by Lemma 4. Summing over $k = 1, \dots, K$ yields the terms shown. The term $\sum_k N(d^3 + d^2 N_k + d N_k^2)$ appears because each stage requires N density evaluations with dataset size N_k . The additional KN accounts for the multiplications. The $K(N + M)$ and KMd^3 come from resampling and gradient/Hessian checks. The last sum accounts for the recomputation after augmentation.

Using $N_k \leq N_{\max}$, we obtain the simplified bound. The cost C_f appears only in stages where $n_k > 0$, and each such stage contributes at most LC_f , so the total contribution from function evaluations is $O(K_{\text{aug}}LC_f)$. $\square$

Remark 9: The cost C_f is problem-dependent. In all examples of this paper, f is a sum of elementary functions (polynomials, exponentials, etc.) evaluated at $O(md)$ terms, where m is the number of data points. Because m is at most 200 when $d = 100$, we have $C_f = O(d^2)$ (polynomial in d). The dominant term in the total complexity is then $N_{\text{TMCMC}}d^3$, since N_{TMCMC} is large (for example, 5×10^6). This matches the empirical runtimes reported in Section 7.

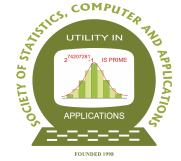
Remark 10: If the objective function f were such that C_f grows faster than any polynomial (for example, exponentially) in d , the total complexity could be dominated by $K_{\text{aug}}LC_f$. However, for the functions considered in this paper (and for most practical optimization problems where derivatives are available), C_f is at most polynomial in d , and the TMCMC term remains the primary bottleneck for moderate to large d .

References

- Adler, R. J. (1981). *The Geometry of Random Fields*. John Wiley & Sons Ltd., New York.
- Adler, R. J. and Taylor, J. E. (2007). *Random Fields and Geometry*. Springer, New York.
- Bhattacharya, D., Maitra, T., Roy, S., and Bhattacharya, S. (2025). Bayesian nonparametrics with random normalizing flows. *ResearchGate preprint*, Available at https://www.researchgate.net/publication/395337211_Bayesian_Nonparametrics_With_Random_Normalizing_Flows.
- Bhattacharya, S. (2021a). IID sampling from doubly intractable distributions. *arXiv preprint*, arXiv:2112.07939.
- Bhattacharya, S. (2021b). IID sampling from intractable multimodal and variable-dimensional distributions. *arXiv preprint*, arXiv:2109.12633.
- Bhattacharya, S. (2022). IID sampling from posterior dirichlet process mixtures. *arXiv preprint*, arXiv:2206.09233.
- Bhattacharya, S. (2025). IID sampling from intractable distributions. *Sankhyā A*, To appear in Professor C. R. Rao special issue.
- Bissiri, P. G., Holmes, C. C., and Walker, S. G. (2016). A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B*, **78**, 1103–1130.
- Carpenter, J., Clifford, P., and Fearnhead, P. (1999). An improved particle filter for non-linear problems. *IEE Proceedings - Radar, Sonar and Navigation*, **146**, 2–7.

- Dey, K. K. and Bhattacharya, S. (2016). On geometric ergodicity of additive and multiplicative transformation based Markov Chain Monte Carlo in high dimensions. *Brazilian Journal of Probability and Statistics*, **30**, 570–613.
- Dey, K. K. and Bhattacharya, S. (2017). A brief tutorial on transformation based Markov Chain Monte Carlo and optimal scaling of the additive transformation. *Brazilian Journal of Probability and Statistics*, **31**, 569–617.
- Dey, K. K. and Bhattacharya, S. (2019). A brief review of optimal scaling of the main MCMC approaches and optimal scaling of additive TMCMC under non-regular cases. *Brazilian Journal of Probability and Statistics*, **33**, 222–266.
- Dutta, S. (2012). Multiplicative random walk Metropolis-Hastings on the real line. *Sankhya. Series B*, **74**, 315–342.
- Dutta, S. and Bhattacharya, S. (2014). Markov Chain Monte Carlo based on deterministic transformations. *Statistical Methodology*, **16**, 100–116.
- Duvenaud, D., Nickisch, H., and Rasmussen, C. E. (2011). Additive Gaussian processes. *Advances in Neural Information Processing Systems*, **24**, 226–234.
- Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, **1**, 209–230.
- Frazier, P. I. (2018). A tutorial on Bayesian optimization. arXiv preprint.
- Giraud, C. (2015). *Introduction to High-Dimensional Statistics*. CRC Press, New York.
- Hunter, D. R. and Lange, K. (2000). Quantile regression via an MM algorithm. *Journal of Computational and Graphical Statistics*, **9**, 60–77.
- Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, **5**, 1–25.
- Lange, K. (2010). *Numerical Analysis for Statisticians*. Springer-Verlag, New York.
- Liu, J. S. and Sabatti, S. (2000). Generalized Gibbs sampler and multigrid Monte Carlo for Bayesian computation. *Biometrika*, **87**, 353–369.
- McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three Methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**, 239–245.
- Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, **6**, 1939–1959.
- Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, **20**, 1177–1184.
- Roy, S. and Bhattacharya, S. (2020). Bayes meets Riemann – Bayesian characterization of infinite series with application to Riemann hypothesis. *International Journal of Applied Mathematics and Statistics*, **59**, 81–128.
- Santner, T. J., Williams, B. J., and I. Notz, W. (2003). *The Design and Analysis of Computer Experiments*. Springer-Verlag, New York.
- Sobol’, I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, **7**, 86–112.
- Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer-Verlag, New York.
- van der Vaart, A. W. and Wellner, J. A. (1996). *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, New York.

- Vose, M. D. (1991). A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on Software Engineering*, **17**, 972–975.
- Walker, A. J. (1974). New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, **10**, 127–128.
- Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, **3**, 253–256.



Neural Architectures for Amortized Bayesian Inference: Statistical Foundations and Empirical Assessments

Roy Shivam Ram Shreshth¹, Arnab Hazra¹ and Gourab Mukherjee²

¹*Department of Statistics and Data Science,
Indian Institute of Technology Kanpur, Kanpur, India 208016*

²*Marshall School of Business,
University of Southern California, Los Angeles, United States 90089-0809*

Received: 13 January 2026; Revised: 8 May 2026; Accepted: 17 July 2026

Abstract

Since the turn of the century, approximate Bayesian inference has steadily evolved as new computational techniques have been incorporated to handle increasingly complex, large-scale predictive problems. The recent success of deep neural networks and foundation models has now given rise to a new paradigm in statistical modeling, in which Bayesian inference can be amortized through large-scale learned predictors. In amortized inference, substantial computation is required at the beginning to train a neural network, but it can subsequently produce approximate posteriors or predictions at much lower computational cost across a wide range of tasks. While the typical Bayesian inference procedures are computationally expensive due to repeated likelihood calculations and Monte Carlo steps for each new dataset, amortized inference provides a much lower computational cost at deployment.

Despite the growing popularity of amortized inference, its statistical interpretation and position within Bayesian inference remain poorly explored. In this paper, we present a statistical perspective on several major neural architectures, including feedforward networks, Deep Sets, and Transformers, and examine how they naturally support amortized Bayesian inference. We explore how these models perform structured approximation and also probabilistic reasoning in ways that yield controlled generalization error throughout a wide range of deployment scenarios, and how these properties can be harnessed for Bayesian computation. Via simulation studies, we evaluate the accuracy, robustness, and uncertainty quantification of amortized inference across varying sample sizes, varying noise distributional families, varying sparsity levels, and multimodality, highlighting its strengths and limitations.

Key words: Amortized inference; Deep sets; Feedforward neural network; Model misspecification; Multimodal posterior approximation; Transformer.

1. Introduction

Nearly twenty-five years ago, Breiman (2001) drew attention to a divide within statistical methodology between a data modeling culture, rooted in explicit probabilistic formulations, and an algorithmic modeling culture, which prioritizes predictive performance through flexible procedures that often lack a generative interpretation. While the former traditionally guided Bayesian analysis, the past two decades have seen a rapid expansion of high-dimensional and highly nonlinear models arising from machine learning, which increasingly influence Bayesian workflows and expand the set of problems for which fully probabilistic treatment is computationally feasible (Shmueli, 2010).

A major force behind this shift is the development of scalable Bayesian computation. Classical tools such as Markov chain Monte Carlo (MCMC) provide well-understood inferential guarantees (Gelfand and Smith, 1990). Still, they can become computationally prohibitive when applied repeatedly across large collections of related datasets or in models with expensive likelihoods. In many modern scientific pipelines, including climate modeling, astrophysics, genomics, and simulation-based inference, analysts must solve thousands or millions of Bayesian inverse problems within tight computational budgets. In such settings, the traditional per-instance paradigm of Bayesian computation becomes infeasible.

The concept of amortized Bayesian inference addresses this challenge by shifting the computational burden from inference at deployment time to a substantial, upfront training phase (Gershman and Goodman, 2014). Instead of running MCMC or optimizing a variational objective separately for each dataset, a flexible neural network is trained on a large collection of simulated or historical Bayesian inference tasks. Once trained, this network provides approximate posterior summaries, including point estimates, densities, or samples, for new datasets at negligible marginal cost. In effect, amortization constructs a global Bayesian surrogate, enabling rapid inference whenever new data arise from the same or a closely related generative mechanism. The benefits of amortization are well-illustrated by the training-deployment dynamics of modern large language models. For example, the Big-Science Large Open-Science Open-access Multilingual Language Model (BLOOM) required more than a million GPU-hours on several hundred high-end processors during its four-month training phase, consuming an estimated 433.2 MWh of energy (Luccioni *et al.*, 2023). As pointed out by Zammit-Mangion *et al.* (2025), this amount of energy is roughly equivalent to the annual usage of 70 Australian households. Once this substantial upfront cost is incurred, the trained model can generate high-quality text on a single GPU at a negligible computational and energy cost. This stark disparity between the cost of training and the efficiency of subsequent inference epitomizes the principle of amortization: a large, one-time computational investment yields an inference mechanism that can be applied repeatedly and efficiently across new inputs.

Amortized Bayesian inference has recently emerged as a powerful alternative to likelihood-based inference for complex data structures, including complex spatial and extreme-value models, where exact likelihoods are intractable or prohibitively expensive (Sainsbury-Dale *et al.*, 2024; Zammit-Mangion *et al.*, 2025). In spatial statistics, Sainsbury-Dale *et al.* (2025) have shown that neural Bayes estimators and graph neural-network-based amortized inferences can deliver fast, accurate inference for Gaussian and non-Gaussian spatial fields,

irregularly spaced data, and high-dimensional latent structures, dramatically reducing computation relative to classical Monte Carlo or Integrated Nested Laplace Approximation (Rue *et al.*, 2009) approaches. In extreme-value analysis, where censored peaks-over-threshold models, heavy-tailed dependence, and spatiotemporal extremes yield challenging likelihood surfaces, amortized Bayesian inference has been employed to construct likelihood-free estimators, accelerate GEV-type Bayesian workflows, and facilitate inference under censoring and weak identifiability (Richards *et al.*, 2024). Collectively, these developments demonstrate that amortized Bayesian inference provides a scalable and flexible framework for tackling the computational bottlenecks inherent in modern spatial and extreme-value modeling, offering near-instant, reusable inference once a simulator-trained network has been fit.

Neural amortization strategies can be broadly classified according to the type of Bayesian quantity they approximate. The first regime, direct neural point estimation, learns a deterministic mapping from data to parameter estimates. Although limited to first-order summaries, this approach is useful in likelihood-free or simulation-based settings where exact Bayesian computation is challenging. Architectures such as Deep Sets (Zaheer *et al.*, 2017) and Set Transformers (Lee *et al.*, 2019) provide principled methods for encoding symmetries inherent in the likelihood or prior structure. Neural posterior estimation (NPE) extends this idea by learning conditional density estimators for the posterior (Greenberg *et al.*, 2019). The second regime, amortized variational inference (AVI), provides approximate posterior distributions rather than point estimates. These methods train neural networks to output parameters of an approximate posterior, effectively learning a variational family indexed by the data. Modern formulations often employ normalizing flows (Rezende and Mohamed, 2015; Papamakarios *et al.*, 2021) to represent posteriors with complex curvature, skewness, and multimodality that cannot be captured by mean-field approximations (Radev *et al.*, 2024). In Bayesian computation, AVI can be viewed as a simultaneous variational approximation over an entire distribution of tasks, rather than a single dataset. The third regime, neural samplers and transport methods, aims to accelerate Bayesian sampling by learning transformations that map complex posterior distributions to simpler latent distributions, where standard Monte Carlo procedures can mix rapidly. Neural transport maps (Hoffman *et al.*, 2019) and flow-matching methods (Lipman *et al.*, 2023) exemplify this approach. These techniques serve as learned reparameterizations or preconditioners for MCMC, yielding substantial computational savings for models with strongly correlated parameters or irregular posterior geometries.

The purpose of this paper is two-fold. We begin by reviewing the statistical background of a few machine learning tools and concepts specific to amortized Bayesian inference. Subsequently, we assess these amortized approaches from a Bayesian perspective, with particular attention to their reliability, structural limitations, and empirical performance relative to classical computational methods. While amortization can offer significant gains in speed when the training distribution accurately represents the deployment environment, its accuracy and stability degrade once the generative mechanism deviates from the conditions encountered during training. To study this behavior systematically, we conduct extensive simulation experiments along four axes. First, we investigate training heterogeneity, examining how variations in task size and diversity influence posterior accuracy and calibration. Second, we study data-scarce deployment and quantify the dependence of amortized posteriors on effective sample size. Third, we evaluate robustness under noise distributional

mismatch, characterizing how predictive and posterior accuracy deteriorate as covariates or structural aspects of the data-generating process drift from the training regime. Fourth, we examine whether amortized neural inference can accurately capture posterior distributions with disconnected support and complex topology, where conventional unimodal approximations are inadequate.

The paper proceeds as follows. Section 2 reviews the neural architectures employed: Feedforward Networks, Deep Sets, and Transformers, and clarifies their connection to Bayesian computation through kernel, variational, and transport-theoretic interpretations. Different aspects of amortized inference are summarized in Section 3. Section 4 presents simulation studies examining the four regimes described above under varying sample sizes, varying noise distributional families, varying sparsity levels, and multimodality.¹ Section 5 summarizes the findings and highlights some future research directions.

2. Popular neural network architectures

In this section, we discuss modern neural network architectures commonly used in amortized inference pipelines. By drawing analogies to classical statistical methods, we explain why each architecture is suited to particular inference tasks rather than treating them as black-box function approximators.

Notation and conventions. We write vectors as columns by default; the only exception is Section 2.3, where the input matrix $X \in \mathbb{R}^{N \times d_{\text{model}}}$ stacks token embeddings as rows. Throughout the paper, θ denotes the trainable weights of the amortized estimator and $\vartheta \in \Theta$ the inferential target (a generic Bayesian parameter). The symbol σ is used for both the component-wise nonlinear activation in an FNN (Section 2.1.1) and standard deviations elsewhere—the meaning is clear from context. By a *token* we mean an indexed unit of an input set; for tabular regression, a token is one observation (x_n, y_n) . An *epoch* is one pass over the meta-training set of tasks, and the per-task sample size N_t is always distinguished from the meta-training-set size T_{train} .

2.1. Feedforward neural networks

The Feedforward Neural Network (FNN), also known as a multilayer perceptron (MLP) (Rumelhart *et al.*, 1986), is the foundational architecture for deep learning. The Universal Approximation Theorem (recalled in Section 2.1.1) guarantees that FNNs can represent arbitrarily complex nonlinear mappings between inputs and outputs.

The mathematical setup of FNN is as follows. Let $L \in \mathbb{N}$ denote the network depth. We define the width of each layer as n_l , where $l \in \{0, \dots, L\}$. Consistent with our notation, the input dimension is $n_0 = d_{\text{in}}$ and the output dimension is $n_L = d_{\text{out}}$. A (dense) feedforward neural network is a parametrized mapping $f_\theta : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{out}}}$, where $\theta = \{(W^{[l]}, b^{[l]})\}_{l=1}^L$, constructed on a strictly layered, acyclic computational graph with full connectivity between consecutive layers. Given an input vector $x \in \mathbb{R}^{d_{\text{in}}}$, the network computes its output through

¹All code used to generate the results is provided at <https://github.com/Royshivam18/Neural-Amortized-Inference> to ensure full reproducibility.

a sequence of recursive transformations. The computation begins by setting the initial activation equal to the input, $a^{[0]} = x$. For each hidden layer $l = 1, \dots, L - 1$, the network forms an affine transformation of the activations of the previous layer as

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}, \quad (1)$$

and then applies a nonlinear activation function $\sigma^{[l]}$ component-wise to obtain

$$a^{[l]} = \sigma^{[l]}(z^{[l]}). \quad (2)$$

Here, $W^{[l]} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b^{[l]} \in \mathbb{R}^{n_l}$ in (1) denote the weight matrix and bias vector for layer l , respectively. An affine transformation of the last hidden-layer activation produces the final output of the network:

$$f_\theta(x) = W^{[L]}a^{[L-1]} + b^{[L]}$$

typically without an additional nonlinearity.

The nonlinear activation functions $\sigma^{[l]}$ in (2) are essential. For instance, the Rectified Linear Unit (ReLU), defined as $\sigma(z) = \max(0, z)$, restricts the network to piecewise-linear functions. Without these nonlinearities, the composition collapses to a single affine map $f_\theta(x) = W_{\text{total}}x + b_{\text{total}}$, and expressivity is lost.

The complete parameter set θ resides in the space $\Theta = \prod_{l=1}^L (\mathbb{R}^{n_l \times n_{l-1}} \times \mathbb{R}^{n_l})$, and the total number of scalar parameters is given by $N_\theta = \sum_{l=1}^L n_l(n_{l-1} + 1)$. This hierarchical composition of functions allows the FNN to construct highly complex decision boundaries from simple linear operations. Figure 1 summarizes this structure schematically.

2.1.1. Universal approximation and adaptive basis functions

The utility of FNNs in statistical inference is rigorously grounded in the Universal Approximation Theorem. Seminal works by Cybenko (1989) and Hornik (1991) established that standard multilayer feedforward networks are universal approximators. Formally, let $C(K)$ denote the space of continuous functions on a compact set $K \subset \mathbb{R}^{d_{\text{in}}}$. The theorem states that the set of functions representable by a neural network with at least one hidden layer and a non-polynomial activation function σ is dense in $C(K)$ with respect to the supremum norm. That is, for any target function $g \in C(K)$ and error tolerance $\epsilon > 0$, there exists a parameter configuration θ such that

$$\sup_{x \in K} |g(x) - f_\theta(x)| < \epsilon.$$

While the theorem guarantees the existence of an approximating network, the geometric nature of the approximation is determined by the choice of activation function.

Modern architectures predominantly use ReLU activations. As shown by Montufar *et al.* (2014) and Arora *et al.* (2018), a ReLU network is a piecewise-affine estimator: it partitions the input domain into finitely many convex polytopes and is affine on each. Nonlinear targets are thus approximated by tiling them with flat facets, analogous to a polygonal mesh approximating a curved surface. In contrast, classical activation functions like the logistic

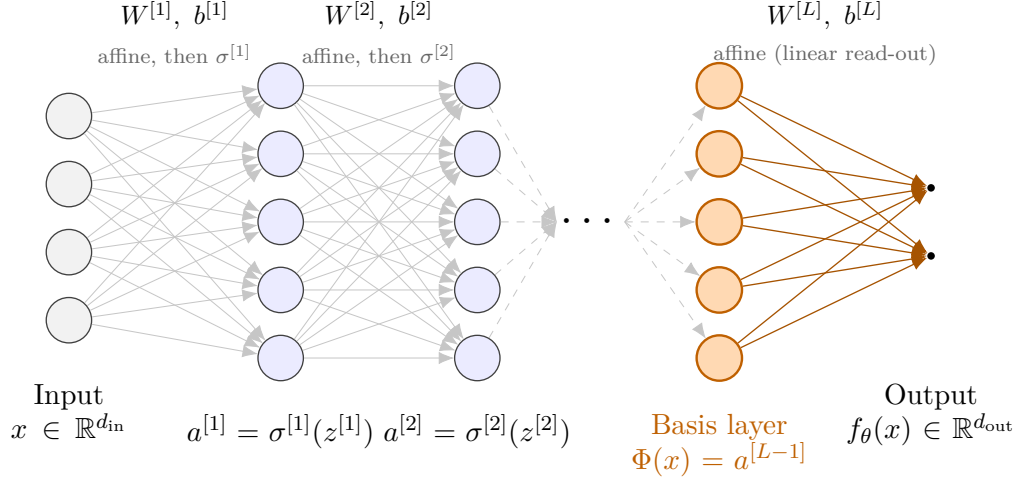


Figure 1: Architecture of an L -layer feedforward neural network. Each hidden layer applies an affine map $z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$ followed by the component-wise nonlinearity $\sigma^{[l]}$; the output layer is a linear read-out $f_{\theta}(x) = W^{[L]}a^{[L-1]} + b^{[L]}$. The penultimate layer $a^{[L-1]} = \Phi(x)$, shown in orange, is highlighted as a learned dictionary of *basis functions*: the network output is a linear combination of the entries of $\Phi(x)$, so the FNN realizes an adaptive basis expansion (Section 2.1.1). In the Transformer of Section 2.3, the query, key, and value projections are themselves FNN outputs of exactly this form, and the layer $\Phi(x)$ corresponds to the features on which attention then operates.

sigmoid, $\sigma(z) = (1 + e^{-z})^{-1}$, or the hyperbolic tangent, $\sigma(z) = \tanh(z)$, operate via smooth superposition. As established in the convergence rates derived by Barron (1993), networks using these smooth, bounded, sigmoidal functions approximate the target function by superimposing smooth ‘steps’ or ‘ridges’. Unlike the sharp boundaries of ReLU polytopes, these activations provide a differentiable approximation where nonlinearity arises from the smooth saturation of neurons (Cybenko, 1989).

We can regard the first hidden layer as learning a dictionary of feature maps denoted by $\Phi(x) = (\phi_1(x), \dots, \phi_{n_1}(x))^{\top}$, where each $\phi_i(x) := \sigma((W_i^{[1]})^{\top}x + b_i^{[1]})$ is a nonlinear basis function. The second hidden layer then computes weighted affine combinations of these features. Analytically, the network output can be written as a basis expansion given by

$$f_{\theta}(x) = c + \sum_{j=1}^{n_2} \beta_j \sigma\left((W_j^{[2]})^{\top} \Phi(x) + b_j^{[2]}\right),$$

where $W_j^{[2]}$ denotes the j -th row of the weight matrix $W^{[2]}$. This expression clarifies that the network is learning a new family of basis functions $\{\sigma((W_j^{[2]})^{\top} \Phi(x) + b_j^{[2]})\}_{j=1}^{n_2}$ built upon the first-layer dictionary, rather than using a fixed basis. Unlike classical nonparametric regression, where the basis (e.g., Fourier or spline) is fixed *a priori*, the FNN adaptively learns the basis parameters to represent the data structure best. This adaptive capability allows the FNN to emulate classical bases. Let $\mathcal{G} = \{g_1, \dots, g_p\} \subset C(K)$ be any finite collection of continuous functions (a prescribed basis). By the universal approximation

theorem, for any $\epsilon > 0$, there exist parameters such that

$$\sup_{x \in K} \left| f_\theta(x) - \sum_{j=1}^p \beta_j g_j(x) \right| < \epsilon.$$

Equivalently, there exists a partition of the hidden units into blocks $\{\mathcal{I}_j\}_{j=1}^p$ such that each block realizes an approximation:

$$\tilde{g}_j(x) = \sum_{i \in \mathcal{I}_j} \alpha_i^{(j)} \sigma((W_i^{[1]})^\top x + b_i^{[1]}), \quad \sup_{x \in K} |g_j(x) - \tilde{g}_j(x)| < \epsilon/p.$$

The second layer learns coefficients to form $\sum_j \beta_j \tilde{g}_j(x)$. Hence, even if the first layer (dictionary) is fixed, a two-hidden-layer network can approximate any finite linear combination of target basis functions on K .

This perspective allows us to map FNNs directly to classical kernel methods. To see this connection, we shift our focus from a trained deep network to a single-hidden-layer network with randomized first-layer parameters:

$$W_i^{[1]} \sim P_W, \quad b_i^{[1]} \sim P_b.$$

If we freeze these parameters at initialization and only train the output-layer weights, each hidden unit activation $\phi_i(x) = \sigma((W_i^{[1]})^\top x + b_i^{[1]})$ acts strictly as a random feature (Rahimi and Recht, 2007). The induced kernel

$$k(x, x') := \mathbb{E}_{W, b} \left[\sigma((W^{[1]})^\top x + b^{[1]}) \sigma((W^{[1]})^\top x' + b^{[1]}) \right]$$

is well-defined under P_W and P_b . Training the output coefficients $\{\beta_j\}_{j=1}^{n_1}$ with squared loss and an ℓ_2 -penalty is equivalent to kernel ridge regression with kernel $k(\cdot, \cdot)$. Moreover, Neal (1996) showed that if we place independent Gaussian priors on these output weights and let the hidden layer width $n_1 \rightarrow \infty$ with appropriate scaling, the induced prior over functions converges to a Gaussian Process (GP) with covariance $k(\cdot, \cdot)$. In this regime, the posterior mean predictor coincides with the GP regression estimator under kernel $k(\cdot, \cdot)$.

This basis-learning behavior is observable empirically. On algorithmic tasks such as modular addition ($a + b \bmod P$), networks initially memorize the training data and then, after extended training, transition to perfect generalization—a phenomenon known as *grokking* (Power *et al.*, 2022). In the generalizing phase, the embedding layers converge to Fourier-like bases (sines and cosines) over the cyclic group (Nanda *et al.*, 2023), an empirical confirmation that the network discovers a problem-appropriate basis rather than fitting a fixed dictionary.

2.2. Deep sets as the summary statistic machine

Standard FNNs are inherently limited when applied to statistical inference tasks involving sets of independent and identically distributed (IID) observations. A standard MLP requires a fixed-size input vector, yet datasets in inference tasks naturally possess variable

cardinality. Furthermore, the joint probability of an IID set, $p(x_1, \dots, x_N)$, is invariant to permutations of the indices. Consequently, any estimator $\hat{\theta}(x_1, \dots, x_N)$ derived from such data must respect this exchangeability. Standard architectures, which process inputs as ordered sequences, fail to capture this symmetry efficiently. To address this, Zaheer *et al.* (2017) proposed the ‘Deep Sets’ architecture, which provides a theoretically grounded framework for learning permutation-invariant functions.

Consider a set of inputs $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ where each element $x_j \in \mathbb{R}^{d_{\text{in}}}$. The Deep Sets architecture processes this set through a three-stage mechanism to ensure permutation invariance. First, an embedding function (encoder) $\phi : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{latent}}}$, typically parameterized as an MLP, maps each individual element to a latent representation:

$$h_j = \phi(x_j), \quad j = 1, \dots, N.$$

Crucially, this mapping is applied identically to every element in the set, ensuring that feature extraction is independent of the position of the element.

Subsequently, a permutation-invariant aggregation operator is applied to the set of latent codes $\{h_1, \dots, h_N\}$. While operators such as the mean or maximum are permissible, sum-pooling is often preferred in theoretical contexts as it preserves information regarding the set size N , which is vital for Bayesian inference. The aggregated summary statistic Z is computed as

$$Z = \sum_{j=1}^N h_j = \sum_{j=1}^N \phi(x_j).$$

Finally, a second neural network $\rho : \mathbb{R}^{d_{\text{latent}}} \rightarrow \mathbb{R}^{d_{\text{out}}}$, referred to as the decoder or regressor, processes the aggregated summary statistic to produce the final estimator

$$\hat{y} = \rho(Z).$$

Combining these steps, the complete Deep Sets estimator can be expressed in the compact form

$$f(\mathcal{X}) = \rho \left(\sum_{x \in \mathcal{X}} \phi(x) \right). \quad (3)$$

Figure 2 depicts this three-stage construction.

From a statistical perspective, this architecture acts as a neural generalization of the Method of Moments (MoM). In classical estimation, parameters θ are estimated by equating empirical moments to theoretical moments. This estimation involves computing a vector of sample moments, $\hat{m}_k = \frac{1}{N} \sum x_j^k$, and applying an inverse mapping g to solve for θ . The Deep Sets formulation mirrors this structure exactly: the encoder $\phi(x)$ learns to extract optimal ‘generalized moments’ (sufficient statistics) rather than relying on fixed polynomial powers; the summation $\sum \phi(x_j)$ accumulates these statistics into a fixed-size summary; and the outer network ρ approximates the inverse mapping g .

This analogy extends to the Moment Generating Function (MGF). The Taylor expansion of the MGF, $M_X(t) = \mathbb{E}[e^{t^\top X}]$, encapsulates all moments of the distribution. If the

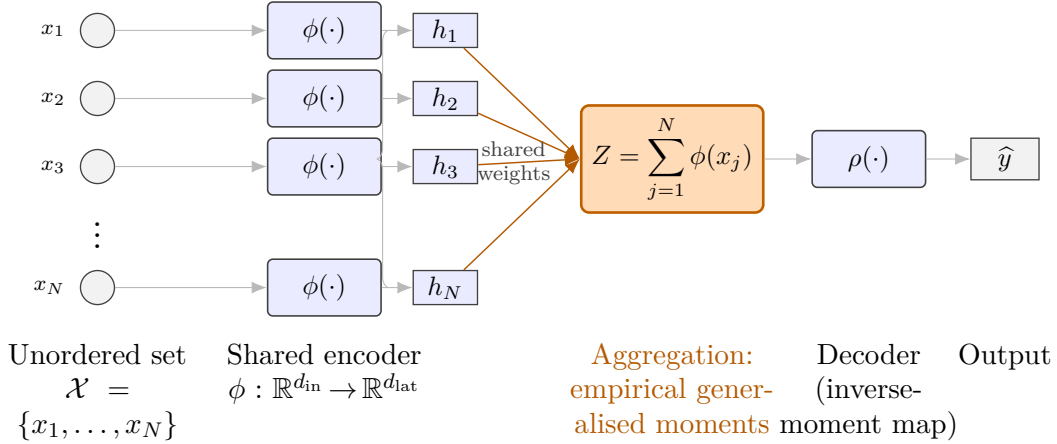


Figure 2: Deep Sets architecture. Each element x_j of an unordered set $\mathcal{X} = \{x_1, \dots, x_N\}$ is passed through the same encoder ϕ (shared parameters), producing latent codes $h_j = \phi(x_j)$. The codes are aggregated by a permutation-invariant sum, $Z = \sum_{j=1}^N \phi(x_j)$, shown in orange. This aggregation block acts as a vector of *empirical generalized moments*: it is the neural analog of the classical method-of-moments statistic $\hat{m}_k = \frac{1}{N} \sum_j x_j^k$, except that the fixed polynomial basis is replaced by the learned features ϕ . The decoder ρ then approximates the inverse moment-to-parameter map and returns $\hat{y} = \rho(Z)$.

encoder $\phi(x)$ possesses sufficient capacity, the aggregated vector $\sum \phi(x)$ effectively approximates the MGF of the underlying empirical distribution. Since the MGF uniquely characterizes the distribution under mild conditions, a Deep Set that learns a sufficient number of these moments captures a finite-dimensional representation of the entire probability measure (Wagstaff *et al.*, 2019). Theoretically, this principle is grounded in the Fisher-Neyman Factorization Theorem, which states that for Exponential Families, the likelihood factorizes through a sufficient statistic $T(x)$. Deep Sets effectively learn this statistic via ϕ , making them ‘Neural Sufficient Statistic Learners’.

2.3. The transformer as a statistical operator

Statistically, a Transformer layer can be read as a learned, data-dependent kernel smoother on the input set. The variant we use in the experiments is the Set Transformer of Lee *et al.* (2019), which applies the same construction described below to an *unordered* input set (no positional encoding), yielding a permutation-equivariant operator over samples. The kernel-smoothing interpretation developed in this subsection applies identically to both.

Let $X \in \mathbb{R}^{N \times d_{\text{model}}}$ denote the input sequence of token embeddings, where the row $x_j \in \mathbb{R}^{d_{\text{model}}}$ corresponds to the j -th token. (We use the term *token* for an indexed unit of the input set; for tabular regression, a token is one observation (x_n, y_n) .) Each Transformer layer (Vaswani *et al.*, 2017) projects every token through three learned feed-forward networks $\phi_Q, \phi_K, \phi_V : \mathbb{R}^{d_{\text{model}}} \rightarrow \mathbb{R}^{d_k}$, yielding queries, keys, and values:

$$q_j = \phi_Q(x_j), \quad k_j = \phi_K(x_j), \quad v_j = \phi_V(x_j),$$

where q_j, k_j, v_j form the j -th rows of $Q, K, V \in \mathbb{R}^{N \times d_k}$. In the original Transformer of

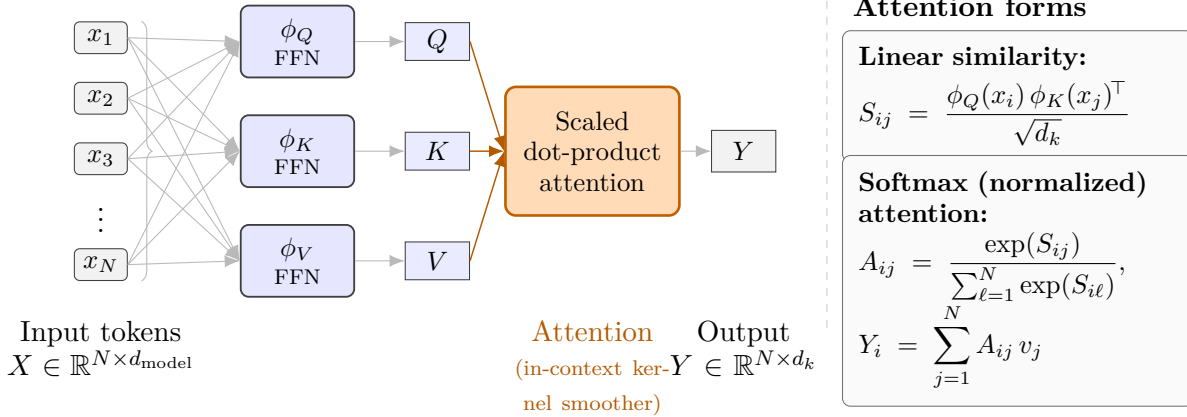


Figure 3: Transformer block. Every token x_j is fed through three independent feed-forward networks ϕ_Q, ϕ_K, ϕ_V —not bare matrix multiplications—producing per-token queries $q_j = \phi_Q(x_j)$, keys $k_j = \phi_K(x_j)$, and values $v_j = \phi_V(x_j)$. The scaled dot-product attention block (orange) aggregates value vectors using a data-dependent kernel induced by the query and key networks. The right panel lists the two attention strategies referenced in this section: the linear (unnormalized) similarity score S_{ij} , and the standard softmax-normalized attention A_{ij} used to mix the values into the output Y_i .

Vaswani *et al.* (2017), each of ϕ_Q, ϕ_K, ϕ_V is a single linear layer (*i.e.*, $\phi_Q(x_j) = x_j W_Q$ for a learned matrix W_Q , and similarly for K, V); modern variants frequently replace these with small multilayer FFNs to increase expressivity. The mechanism then computes a pairwise similarity matrix $S \in \mathbb{R}^{N \times N}$ with entries $S_{ij} = \phi_Q(x_i) \phi_K(x_j)^\top$, representing the unnormalized alignment between the i -th query and the j -th key. The scores are scaled by $1/\sqrt{d_k}$ and passed through a row-wise softmax to yield the attention matrix A ,

$$A_{ij} = \frac{\exp(S_{ij}/\sqrt{d_k})}{\sum_{\ell=1}^N \exp(S_{i\ell}/\sqrt{d_k})}.$$

The layer output is $Y = AV$, *i.e.*, $Y_i = \sum_{j=1}^N A_{ij} v_j$. In standard implementations, several such *heads* (parallel attention computations with independent ϕ_Q, ϕ_K, ϕ_V) are concatenated to capture diverse dependencies. Figure 3 summarises this pipeline and lists the two attention forms commonly used downstream.

The attention matrix A admits a direct interpretation as a Nadaraya–Watson kernel regression estimator (Nadaraya, 1964; Watson, 1964). Define a data-dependent kernel κ_η parameterized by the projection networks $\eta = \{\phi_Q, \phi_K\}$ as

$$\kappa_\eta(x_i, x_j) = \exp\left(\frac{\phi_Q(x_i) \phi_K(x_j)^\top}{\sqrt{d_k}}\right).$$

Unlike stationary kernels used in nonparametric regression (*e.g.*, the Radial Basis Function; see Wand and Jones, 1994) that depend only on the Euclidean distance $\|x_i - x_j\|$, this kernel is non-stationary: it learns a metric in which similarity is determined by alignment of the projected features.

Under this definition, A_{ij} is precisely the normalised kernel weight, and Y_i is the Nadaraya–Watson estimator of the values V at the query location x_i , conditioned on the context $\{x_j\}$:

$$Y_i = \mathbb{E}_{\hat{P}}[V \mid x_i] = \sum_{j=1}^N \frac{\kappa_\eta(x_i, x_j)}{\sum_{\ell=1}^N \kappa_\eta(x_i, x_\ell)} v_j. \quad (4)$$

Statistically, the self-attention layer performs in-context kernel smoothing: it integrates information across the value vectors V , with the smoothing bandwidth and shape dictated by the learned similarity structure κ_η .

This kernel-smoothing interpretation invites a comparison with Gaussian Process (GP) regression. For a GP with kernel $k(\cdot, \cdot)$ and noise variance σ_n^2 , the posterior mean at a test point x_* given training data $(X, \mathbf{y})$ is $m(x_*) = k(x_*, X)(K(X, X) + \sigma_n^2 I)^{-1} \mathbf{y}$. Both architectures form a weighted sum of targets, but their mixing mechanisms differ. The GP posterior requires inverting the kernel matrix, an $\mathcal{O}(N^3)$ operation; attention replaces this inversion with a row-wise softmax that scales as $\mathcal{O}(N^2)$. Softmax normalization is only an approximation to the inversion step, but it allows the Transformer to scale to substantially longer sequences while retaining the capacity to model complex dependencies (Rasmussen and Williams, 2006).

The query and key networks ϕ_Q, ϕ_K also admit a spectral interpretation via Mercer’s theorem. A Mercer kernel $\kappa(x, x')$ admits the eigenfunction expansion $\sum_r \lambda_r \psi_r(x) \psi_r(x')$, and the Transformer approximates this with a finite-rank inner product of feature maps,

$$\phi_Q(x_i) \phi_K(x_j)^\top = \langle \phi_Q(x_i), \phi_K(x_j) \rangle,$$

where ϕ_Q and ϕ_K are learned feature maps. Standard attention decouples these projections ($\phi_Q \neq \phi_K$)—a choice motivated by causal or temporal asymmetry in language modeling—so the resulting attention matrix is not a Mercer kernel in the classical (symmetric) sense. Variants that tie the projections, such as the Reformer (Kitaev *et al.*, 2020), recover symmetry by setting $\phi_Q = \phi_K$ and can then exploit locality-sensitive hashing (Andoni and Indyk, 2008).

A full Transformer stack consists of multiple attention layers rather than a single one. This composition has a natural analog in Deep Gaussian Processes (DGPs; Damianou and Lawrence, 2013), where the output of one Gaussian process serves as the input to the next, warping the data geometry layer by layer. Each attention layer computes a data-dependent kernel over the features extracted by the previous layer, so the depth of the stack enables the model to represent strongly non-stationary dependencies that a single shallow kernel cannot capture.

3. Amortized Bayesian inference

Amortized Bayesian inference is a computational strategy in which a substantial initial computational budget is allocated to training a global model, thereby making subsequent inference for individual test cases computationally efficient. This approach is particularly potent when combined with deep learning, replacing complex, iterative computational steps (such as MCMC or numerical optimization) with efficient forward passes of a neural network.

In many high-dimensional settings, such as image analysis or large-scale regression, the raw data x often contains redundant information. Neural networks allow us to compress x into lower-dimensional summary statistics, which classical statistical techniques can then leverage. In this work, we specifically focus on two complementary regimes of amortized inference: point estimation and uncertainty quantification.

3.1. Point estimation

In the first regime, the neural network acts as a direct function approximator for a statistical estimator. The goal is to learn a mapping $f_\theta : \mathcal{X} \rightarrow \Theta$, parameterized by network weights θ , that outputs a point estimate $\hat{\vartheta}$ given observed data x . Throughout this section, we use $\vartheta \in \Theta$ for the inferential target (a generic Bayesian parameter) and reserve θ for the trainable weights of the amortized estimator, matching the convention in Section 2. This formulation reduces inference to a supervised regression problem.

To ensure the network approximates a valid Bayesian estimator, we employ a decision-theoretic framework. We define a loss function $L(\vartheta, \hat{\vartheta})$ that quantifies the cost of estimating the true parameter ϑ as $\hat{\vartheta}$, and minimize the Bayes risk—the expected loss over the joint distribution of parameters and data. Let $\pi(\vartheta)$ denote the prior over parameters and $p(x | \vartheta)$ the likelihood. The Bayes risk $\mathcal{R}(\pi, f_\theta)$ is

$$\mathcal{R}(\pi, f_\theta) = \mathbb{E}_{\vartheta \sim \pi(\vartheta)} \left[\mathbb{E}_{x \sim p(x|\vartheta)} [L(\vartheta, f_\theta(x))] \right], \quad (5)$$

or equivalently

$$\mathcal{R}(\pi, f_\theta) = \int_{\Theta} \int_{\mathcal{X}} L(\vartheta, f_\theta(x)) p(x | \vartheta) \pi(\vartheta) dx d\vartheta.$$

Minimizing this risk with respect to θ yields the Bayes estimator under the chosen loss L . For instance, the squared-error loss $L(\vartheta, \hat{\vartheta}) = \|\vartheta - \hat{\vartheta}\|^2$ yields $f_{\theta^*}(x) \approx \mathbb{E}[\vartheta | x]$, the posterior mean.

3.1.1. Implicit topology learning as meta-learning

The capability of a neural network to perform statistical inference is not merely a curve-fitting exercise; it represents a form of meta-learning (Hospedales *et al.*, 2022). In traditional supervised learning, a model maps a specific input x to a corresponding label y . In the amortized inference regime, the network learns a functional mapping from an entire dataset $\mathcal{D}_n = \{x_1, \dots, x_n\}$ to a parameter space Θ .

This learning mechanism requires the network to internalize the topology of the parameter manifold. Rather than memorizing answers for specific datasets, the network must learn the inductive biases of the generative model—it learns *to estimate*, not to recall. For example, when estimating the variance of a distribution, the network must implicitly learn that this parameter is strictly positive and that it correlates with the input dispersion, regardless of the specific input values.

Empirically, the latent representations learned by the network form a low-dimensional manifold aligned with the true parameter space Θ . This geometric alignment allows the

network to generalize to unseen datasets generated from the same prior, effectively replacing the manually derived formulas of classical estimation (such as the ordinary least squares formula) with a learned nonlinear equivalent.

3.2. Uncertainty quantification

While point estimation provides a single optimal summary of the parameter space—typically corresponding to Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP) approximations (Bishop, 2006; Goodfellow *et al.*, 2016), it is inherently insufficient for capturing the full predictive or structural distribution. In complex modeling tasks, a single point estimate fails to capture the epistemic uncertainty arising from finite training data or the aleatoric uncertainty inherent in the data-generating process. Consequently, moving beyond deterministic predictions requires a coherent framework for full uncertainty quantification (UQ).

Bayesian neural networks: Among the foundational paradigms for incorporating probabilistic reasoning into deep architectures is the Bayesian Neural Network (BNN) framework (MacKay, 1992; Neal, 1996). Instead of learning a single deterministic point estimate for the connection weights, a BNN assigns a prior probability distribution over the network parameters, denoted as $p(\mathbf{w})$. Given a training dataset $\mathcal{D}$, the inferential goal shifts to computing the posterior distribution over the weights, $p(\mathbf{w} \mid \mathcal{D})$, via Bayes’ theorem. The predictive distribution for a novel test input $\mathbf{x}^*$ is subsequently evaluated by marginalizing over this parameter posterior:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \int p(y^* \mid \mathbf{x}^*, \mathbf{w})p(\mathbf{w} \mid \mathcal{D}) d\mathbf{w}$$

Despite their theoretical appeal, BNNs are computationally demanding and practically fragile. The true posterior over weights is analytically intractable because deep-network loss surfaces are high-dimensional and non-convex, so practitioners rely on structural or stochastic approximations. Sampling methods such as MCMC mix slowly in deep models and do not scale to architectures beyond shallow ones (Neal, 1996). Optimization-based approaches such as Variational Inference (VI; Graves, 2011; Blundell *et al.*, 2015) typically double the parameter count to model mean-field variances, introduce variance into gradient updates, and remain sensitive to hyperparameter tuning and initialization.

Deep ensembles: Another widely adopted paradigm for uncertainty quantification is the deep ensemble framework (Lakshminarayanan *et al.*, 2017). This approach involves training a collection of M independent neural networks, with each model initialized from a distinct random seed and exposed to different data shuffles or splits. At deployment, the epistemic uncertainty is quantified by evaluating the variance across the predictions or estimated quantities of the individual ensemble members. From a theoretical standpoint, because the optimization landscape of a deep neural network is highly non-convex and high-dimensional, varying the initialization trajectory forces each network to settle into distinct local minima. Consequently, the distributed outputs generated by these independent architectures can be viewed as empirical samples drawn from an implicit, multimodal posterior distribution, effectively capturing the geometric landscape of the model’s predictive uncertainty.

The principal bottleneck is optimization cost: training time and memory scale linearly with the ensemble size M , so deep ensembles are difficult to scale in large networks or resource-constrained deployment environments where repeatedly retraining the model is infeasible.

Amortized variational inference: A third major paradigm for capturing uncertainty focuses on explicitly approximating the posterior distribution by restricting it to a tractable parametric family. Rather than executing independent optimization or sampling routines for each individual dataset, amortized variational inference (AVI) trains a global neural network to learn a direct functional mapping from the observed data to the parameters of a chosen variational distribution, $q_\phi(\boldsymbol{\theta} \mid \mathbf{x})$ (Kingma and Welling, 2014; Gershman and Goodman, 2014). During the upfront training phase, the network is optimized by maximizing the Evidence Lower Bound (ELBO) or minimizing an explicit probabilistic divergence (such as the Kullback-Leibler divergence) across a wide distribution of tasks. While traditional implementations often rely on simple mean-field Gaussian approximations, modern formulations extend this framework by integrating highly flexible density estimators, such as normalizing flows, to represent complex, skewed, or multimodal posterior surfaces (Rezende and Mohamed, 2015; Papamakarios *et al.*, 2021).

A primary advantage of this methodology is its computational efficiency during deployment. It shifts the intensive inductive workload into a single offline training phase, spreading computational costs across all subsequent inference tasks. After optimizing the global network, evaluating the full approximate posterior or generating samples for a new dataset requires only one rapid forward pass through the network. This approach avoids the slow, iterative updates of classical MCMC methods and eliminates the large storage requirements of ensemble techniques.

Uncertainty methodologies utilized in this study: To systematically evaluate and quantify uncertainty within our experimental analysis without incurring the computational overhead of deep ensembles or the optimization volatility of standard BNNs, we utilize three distinct methodologies: Monte Carlo Dropout, Monte Carlo Stability Analysis, and Flow-Based Posterior Sampling.

Monte carlo dropout: Dropout (Srivastava *et al.*, 2014) was introduced as a regularization technique used strictly during training: a random fraction of each layer’s activations is zeroed at every step, which discourages co-adaptation of units and reduces overfitting. Gal and Ghahramani (2016) showed that retaining dropout at inference time yields a principled approximation to Bayesian inference. By keeping dropout active and repeating the forward pass on the same input, one obtains an empirical predictive distribution; the resulting stochastic forward passes are equivalent to variational inference in a Deep Gaussian Process, grounding a heuristic regularizer in formal inferential theory.

Formally, let $\widehat{\mathbf{W}}_l$ denote the trained weight matrix for layer $l \in \{1, \dots, L\}$. At inference time, for each stochastic forward pass $t \in \{1, \dots, T\}$, we sample a random binary mask vector from a Bernoulli distribution:

$$\mathbf{z}_{l,t} \sim \text{Bernoulli}(1 - q)$$

where q is the user-defined dropout probability. The active weights for the t -th realization are computed via the element-wise multiplication $\mathbf{W}_{l,t} = \widehat{\mathbf{W}}_l \cdot \text{diag}(\mathbf{z}_{l,t})$. Given a novel test instance $\mathbf{x}^*$, the predictive posterior distribution is empirically approximated by averaging across the collection of T stochastic model trajectories:

$$p(y^* | \mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^T p(y^* | \mathbf{x}^*, \mathbf{W}_t)$$

From this sampled distribution, the epistemic uncertainty can be quantified by calculating the empirical variance across the T predictions.

Despite its convenience, Monte Carlo (MC) dropout has well-known limitations. As a variational approximation, it inherits the standard failure modes of mean-field VI: a tendency to underestimate posterior variance and an inability to represent multimodal weight distributions, both of which are constrained by the rigid Bernoulli masking. MC dropout is also sensitive to model capacity: in small or shallow architectures, removing a fixed fraction of units degrades predictive capacity rather than calibrating uncertainty.

Monte carlo stability analysis: To evaluate the consistency of our neural estimator, we employ a simulation-based stability analysis. We leverage the generative nature of our experimental framework to assess estimator variance across repeated realizations of the data-generation process.

For a fixed ground-truth p -length parameter vector β_{true} and a specific sample size N , we generate B independent datasets $\mathcal{D}_1, \dots, \mathcal{D}_B$, where each $\mathcal{D}_b = \{(X^{(b)}, y^{(b)})\}$ for $b = 1, \dots, B$. These datasets represent potential realizations of the world under identical underlying laws but different noise instances. We pass each dataset through the trained Transformer model to obtain a set of parameter estimates $\{\hat{\beta}^{(1)}, \dots, \hat{\beta}^{(B)}\}$. It is essential to note that our model employs a hard-thresholding mechanism to enforce sparsity; the network produces both a magnitude vector and an inclusion probability. The final estimate is computed as the element-wise product of the magnitude and a binary mask derived from the probabilities (thresholded at 0.5). We quantify uncertainty by computing the standard deviation of these estimates across the B replications. The empirical stability metric is defined as the average standard deviation across all p coefficients:

$$\sigma_{\text{MC}}(N) = \frac{1}{p} \sum_{j=1}^p \sqrt{\frac{1}{B-1} \sum_{b=1}^B (\hat{\beta}_j^{(b)} - \bar{\beta}_j)^2}. \quad (6)$$

By evaluating the metric in (6) across varying sample sizes (*e.g.*, $N \in \{50, \dots, 1000\}$), we can empirically check the statistical consistency of the Transformer. A monotone decrease of σ_{MC} with N is consistent with a $\sqrt{N}$ -style information-aggregation rate.

Flow-based posterior sampling: While the Monte Carlo stability analysis quantifies the stability of the point estimate, it does not capture the whole geometry of the posterior distribution, particularly in multimodal settings. To address this, we employ a Flow Matching framework to train a neural network as a generative model that draws samples directly from the posterior $p(\theta | x)$. Further, to overcome the computational bottlenecks of traditional

sampling, we leverage Conditional Flow Matching (CFM, Lipman *et al.*, 2023). In contrast to MCMC samplers, which rely on local iterative updates to explore the parameter space, our flow-based mechanism learns a deterministic transformation from a simple base distribution to the complex target distribution.

The core premise is to identify a transformation capable of mapping samples from a tractable initial distribution p_0 (*e.g.*, a standard Gaussian) to the intractable target posterior p_1 . Flow matching models this transformation by identifying an ideal vector field that dictates the optimal movement of probability mass from the initial state to the final state over a time horizon $t \in [0, 1]$. We define a time-dependent vector field $v_t : \mathbb{R}^d \rightarrow \mathbb{R}^d$ parameterized by a neural network with parameters ϕ . The evolution of a sample z_t is governed by the ordinary differential equation

$$\frac{dz_t}{dt} = v_t(z_t; \phi).$$

By integrating this derivative from $t = 0$ to $t = 1$, the neural network effectively transports samples from the prior directly to the posterior.

During training, we regress the neural vector field v_t against a conditional target vector field u_t . We select the Optimal Transport path (Peyré and Cuturi, 2019), which creates a straight-line trajectory between a noise sample z_0 and a data sample z_1 . The loss function minimizes the discrepancy between the learned field and this ideal trajectory

$$\mathcal{L}_{\text{CFM}}(\phi) = \mathbb{E}_{t, z_0, z_1} [\|v_t(z_t) - (z_1 - z_0)\|^2]. \quad (7)$$

This approach effectively amortizes the inference cost: once the ideal vector field is learned, sampling becomes a fast, non-iterative forward pass through the ODE solver, avoiding the slow mixing and mode collapse issues often encountered with MCMC in complex topologies.

4. Experimental evaluation

4.1. Experiment I: Latent structure recovery

We study task heterogeneity through a clustered task prior and evaluate whether permutation-invariant neural estimators can amortize inference by learning shared latent structure across tasks. Concretely, we construct a meta-learning environment where each task corresponds to a linear regression problem with a task-specific coefficient vector $\beta_t \in \mathbb{R}^p$, but the collection $\{\beta_t\}$ is constrained to lie on a finite set of latent centroids. We compare two amortized estimators, Deep Sets and Set Transformer, against a classical per-task estimator.

Generative process. We fix the feature dimension $p = 20$ and the number of latent clusters K . We first sample K latent centroids following $\mu_k \sim \mathcal{N}(0, \tau^2 I_p)$, $k = 1, \dots, K$ with $\tau = 3.0$. These centroids remain fixed for the entire experiment and define a latent task mixture. For each task t , we sample its regression coefficient vector by selecting one centroid uniformly at random, *i.e.*, $\beta_t \sim \text{Uniform}(\{\mu_1, \dots, \mu_K\})$. Conditioned on β_t , we generate a task support set $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ with random support size

$N_t \sim \text{Discrete} - \text{Uniform}\{10, 11, \dots, 30\}$. We sample the inputs as $x_{t,n} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, I_p)$ and outputs follow the linear model

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n}, \quad \varepsilon_{t,n} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \quad \sigma = 1.$$

Training and test meta-datasets. For each choice of $K \in \{5, 10, 50\}$, we independently generate a training meta-dataset of T_{train} tasks and a test meta-dataset of $T_{\text{test}} = 200$ held-out tasks using the same generative process (*i.e.*, the same K and hyperparameters). The neural models are trained across tasks to learn an amortized mapping

$$f_\theta : \mathcal{D}_t \mapsto \hat{\beta}_t \in \mathbb{R}^p,$$

where f_θ is expressed as either a Deep Sets encoder or a Set Transformer encoder followed by a regression head. Both architectures operate on the set of paired observations $\{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ and are therefore permutation-invariant in the ordering of samples within a task.

Baselines and evaluation metric. As classical baselines, we compute a per-task Ordinary Least Squares (OLS) estimate $\hat{\beta}_t^{\text{OLS}}$ and a Ridge regression estimate $\hat{\beta}_t^{\text{Ridge}}$ using only the support set $\mathcal{D}_t$. We evaluate all estimators on the test tasks using parameter estimation error

$$\text{MSE}_\beta = \frac{1}{T_{\text{test}}} \sum_{t=1}^{T_{\text{test}}} \left\| \hat{\beta}_t - \beta_t \right\|_2^2, \quad (8)$$

and train neural models for 100 epochs by minimizing the same objective over training tasks.

Parameter estimation error. Table 1 reports parameter-estimation error in the latent-structure-recovery experiment. The amortized neural estimators achieve parameter MSE one to two orders of magnitude below the per-task Ridge baseline. Linear baselines (OLS and Ridge) cannot exploit the shared geometry across tasks, so they suffer high error regardless of the cluster count K . Ridge improves marginally over OLS through ℓ_2 regularization, but remains fundamentally per-task. In low-complexity regimes ($K = 5, 10$), the neural estimators closely approximate the Bayes-optimal estimator; at $K = 50$, both architectures continue to generalize, indicating that they exploit the finite cluster structure of the parameter space even as it becomes more diffuse.

4.2. Experiment II: Distributional robustness

We next study the behavior of amortized estimators as the meta-training-set size grows, and their robustness to distributional mismatch. This experiment jointly evaluates (i) sample complexity in the number of training tasks and (ii) stability under increasingly non-Gaussian noise distributions. Unlike Experiment I, where task heterogeneity is induced through a clustered prior, here tasks are independently generated but exhibit substantial within-task uncertainty due to high-variance parameters and complex residual structure.

Table 1: Parameter estimation error (MSE_β in (8)), standard error (SE), and 95% coverage (Cov95) in the latent structure recovery experiment. Results are averaged over 200 held-out test tasks. For Ridge, we consider per-task linear estimators trained independently on each task using only its support set. For Deep Sets and Set Transformer, we consider amortized estimators trained across tasks.

Latent complexity	Ridge (per-task)	Deep Sets		Set Transformer	
	MSE	MSE $\pm$ SE	Cov95	MSE $\pm$ SE	Cov95
Clusters ($K = 5$)	5.027	0.248 ± 0.016	0.980	0.043 ± 0.002	0.999
Clusters ($K = 10$)	5.245	1.031 ± 0.047	0.928	0.102 ± 0.007	0.980
Clusters ($K = 50$)	4.822	6.242 ± 0.169	0.315	2.370 ± 0.100	0.623

Generative process. We fix the feature dimension $p = 20$. For each task t , we draw regression coefficients independently from a high-variance prior $\beta_t \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 9^2 I_p)$. Conditioned on β_t , we generate a support set $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$ with inputs sampled as $x_{t,n} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, I_p)$. We generate outputs according to the linear model

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n},$$

where the noise distribution $\varepsilon_{t,n}$ varies across regimes described below. To rigorously evaluate robustness to distributional mismatch, the final evaluation is performed on a held-out test set of $T_{\text{test}} = 600$ independent tasks for each noise regime.

Noise regimes. To assess robustness to distributional mismatch, we consider four residual distributions: (I) Gaussian (symmetric): $\varepsilon_{t,n} \sim \mathcal{N}(0, 1)$, (II) Asymmetric: $\varepsilon_{t,n} \stackrel{d}{=} E - 1$, where $E \sim \text{Exp}(1)$, (III) Bimodal: $\varepsilon_{t,n}$ follows a two-component Gaussian mixture with weights $\{0.8, 0.2\}$, and (IV) Trimodal: $\varepsilon_{t,n}$ follows a three-component Gaussian mixture with weights $\{0.8, 0.1, 0.1\}$. For the multimodal cases, component-specific means and variances are chosen to ensure well-separated modes, inducing heavy-tailed and non-convex error distributions.

Training and evaluation protocol. We generate a meta-dataset of up to $T = 6000$ tasks using the generative process above. From this pool, we form meta-training sets of size $T_{\text{train}} \in \{5, 10, 30, 50, 80, 100, 300, 400\}$ and a held-out test set of $T_{\text{test}} = 600$ tasks. For each value of T_{train} , we train Deep Sets and the Set Transformer to convergence using the same fixed optimization budget per fit, and evaluate the resulting estimator on the held-out test tasks.

This protocol examines sample complexity as a function of the meta-training-set size. Robustness to distributional mismatch is assessed by evaluating each trained model under every residual regime (Gaussian, Asymmetric, Bimodal, and Trimodal) while keeping the task prior and the task sample-size distribution fixed.

Objective and metrics. As in Experiment I, Deep Sets and Set Transformer architectures learn an amortized mapping $f_\theta : \mathcal{D}_t \mapsto \hat{\beta}_t \in \mathbb{R}^p$. We evaluate the performance using

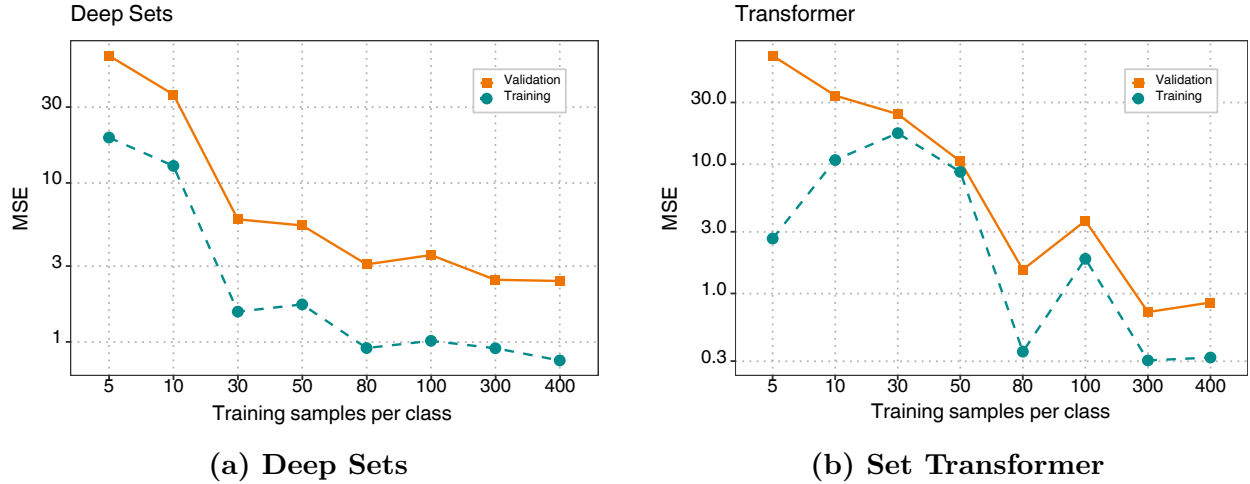


Figure 4: Sample complexity in the meta-training-set size: dashed lines report training-task fit and solid lines report held-out test error, as a function of the number of training tasks T_{train} . (a) Deep Sets achieves low error even with small $T_{\text{train}} < 50$ but plateaus quickly. (b) The Set Transformer requires more training tasks ($T_{\text{train}} > 80$) to stabilize, but reaches a substantially lower asymptotic error.

the parameter estimation error $\text{MSE}_\beta = \mathbb{E} [\|\hat{\beta}_t - \beta_t\|_2^2]$, computed over held-out tasks and averaged across noise regimes and support sizes.

Sample complexity in the meta-training-set size. Figure 4 reports the training-task fit and held-out test error of Deep Sets and the Set Transformer as the meta-training-set size T_{train} is varied over $\{5, 10, 30, 50, 80, 100, 300, 400\}$. The horizontal axis is the number of training tasks on which the amortized estimator was fit. It is distinct from the optimization budget (held fixed per fit) and from the per-task sample size N_t (drawn from a fixed distribution).

Deep Sets (left panel) attains low test error with only a small number of training tasks; both training and held-out error decrease sharply for $T_{\text{train}} \leq 30$ –50 and improve only marginally thereafter. The Set Transformer (right panel) has higher error and greater variance in the small- T_{train} regime, but its error continues to decrease as more training tasks are supplied. By $T_{\text{train}} \geq 300$, the Set Transformer reaches a substantially lower asymptotic error than Deep Sets. In summary, Deep Sets is sample-efficient in T_{train} but capacity-limited; the Set Transformer requires more training tasks to identify but attains a better solution.

Complex noise robustness. To systematically evaluate the robustness of the models, we injected different noise profiles into the dataset. The noise distributions in Table 2 are categorized into three distinct groups: *Clean* (symmetric, baseline Gaussian noise), *Skewed* (asymmetric, heavy-tailed noise that violates standard normality assumptions), and *Multimodal* (mixture models introducing distinct, unequally weighted error clusters).

Training the Set Transformer is more demanding than training simpler architectures: it requires a larger meta-training set and careful hyperparameter tuning, and we observed slower and less stable convergence under heavy-tailed and multimodal noise. The high coverage reported for the Set Transformer in Table 2 is partly explained by the wider predictive

Table 2: Parameter estimation error (MSE $\pm$ SE) and 95% coverage across clean, skewed, and multimodal noise distributions. *Clean Small* and *Clean High* denote Gaussian residuals with low and high variance ($\sigma = 1$ and $\sigma = 3$). *Right/Left Skew (r1)* and *(r2)* are shifted Exponential residuals with rate parameters $r_1 = 1$ and $r_2 = 0.5$ (heavier-tailed). *Bi-Unbalanced ($m_{\mu}\sigma$)* denotes a two-component Gaussian mixture with weights $\{0.8, 0.2\}$ whose minority component has mean μ and standard deviation σ . *Trimodal Tight* and *Trimodal Wide* denote three-component Gaussian mixtures with weights $\{0.8, 0.1, 0.1\}$ whose modes are closely or widely separated, respectively.

Noise Type	Deep Sets		Set Transformer	
	MSE $\pm$ SE	Cov95	MSE $\pm$ SE	Cov95
<i>Clean Distributions</i>				
Clean Small	6.86 $\pm$ 0.357	0.980	1.72 $\pm$ 0.100	0.984
Clean High	7.17 $\pm$ 0.341	0.974	1.56 $\pm$ 0.081	0.991
<i>Skewed Distributions</i>				
Right Skew (r1)	7.61 $\pm$ 0.393	0.972	1.65 $\pm$ 0.092	0.988
Left Skew (r1)	7.20 $\pm$ 0.340	0.975	1.63 $\pm$ 0.100	0.989
Right Skew (r2)	6.78 $\pm$ 0.339	0.977	1.54 $\pm$ 0.082	0.989
Left Skew (r2)	8.00 $\pm$ 0.404	0.974	1.73 $\pm$ 0.108	0.986
<i>Multimodal Distributions</i>				
Bi-Unbalanced (m3s1)	7.96 $\pm$ 0.414	0.972	2.21 $\pm$ 0.125	0.977
Bi-Unbalanced (m4s2)	9.99 $\pm$ 0.451	0.953	3.36 $\pm$ 0.201	0.955
Trimodal Tight	8.00 $\pm$ 0.377	0.975	1.95 $\pm$ 0.112	0.985
Trimodal Wide	7.42 $\pm$ 0.385	0.970	1.72 $\pm$ 0.089	0.990

variance due to this weight sensitivity, as reflected in the larger standard errors of its MSE estimates.

As shown in Figure 5, we assess the stability of the amortized estimators by measuring Monte Carlo variability as a function of the support set size N . Here, N denotes the number of observations provided to the model for a single task at inference time, and should be clearly distinguished from the meta-training-set size T_{train} or the optimization budget. For each task and noise regime, we draw replicated samples over the support set and compute the standard deviation of the resulting parameter estimates. This quantity captures the sensitivity of the estimator to sampling variability and provides a direct measure of predictive stability. Both architectures exhibit a clear monotonic decrease in Monte Carlo standard deviation as N increases, indicating that parameter estimates become more stable as additional evidence is provided. This behavior is consistent with that of well-behaved statistical estimators, where the uncertainty decreases as the information content of the observed data increases. Notably, the Set Transformer demonstrates substantially lower uncertainty across all noise regimes and support sizes. In particular, its Monte Carlo standard deviation remains below 0.6 even in low-data settings, whereas Deep Sets exhibits considerably higher variability, with deviations exceeding 1.5 for small N . Moreover, the Transformer’s uncertainty curves for Gaussian, asymmetric, and multimodal noise remain closely aligned, suggesting that its stability is largely insensitive to the form of the noise distribution. This observation indicates that

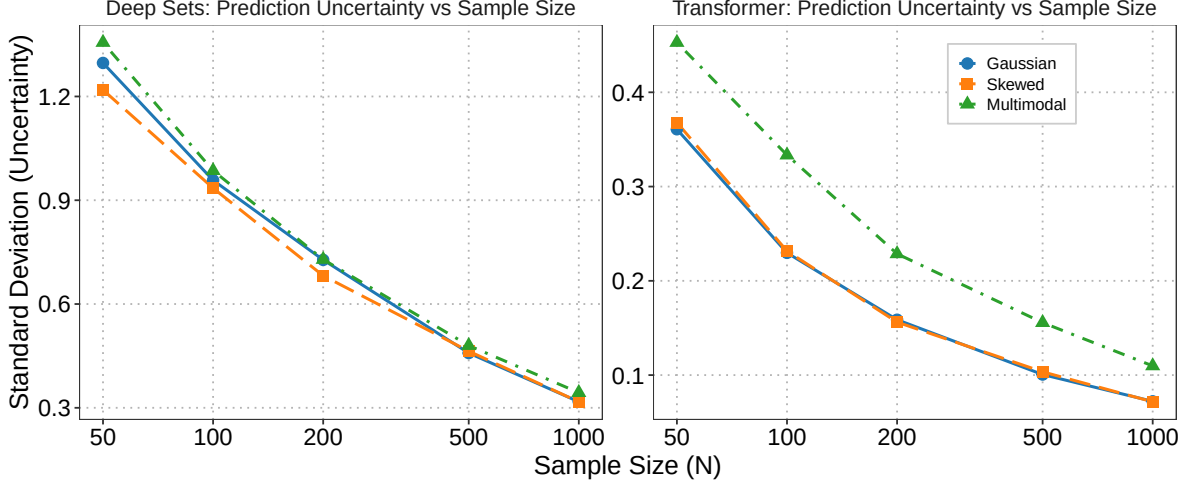


Figure 5: Monte Carlo stability analysis. Standard deviation of the estimated regression coefficients (computed via replicated sampling of the task support set) as a function of support set size N . Both architectures exhibit decreasing uncertainty as N increases.

the attention-based architecture produces more reliable and consistent parameter estimates under both data scarcity and distributional mismatch.

4.3. Experiment III: Sparse signal recovery

To further probe Regime II (the data-scarce frontier), we study the recovery of sparse signals in high-dimensional linear regression. Each task corresponds to estimating a task-specific sparse coefficient vector from a small set of observations. Our goal is to learn an amortized, permutation-invariant estimator that identifies both the support and magnitudes of the sparse signal.

Generative process. We fix the feature dimension $p = 100$. For each task t , we generate a sparse regression coefficient vector $\beta_t \in \mathbb{R}^p$ with a sparsity level k_t denoting the *percentage of zero elements*, so that

$$\|\beta_t\|_0 = p - k_t, \quad k_t \in \mathcal{K} := \{5, 10, 15, \dots, 100\}.$$

The active support $S_t \subset \{1, \dots, p\}$ with $|S_t| = p - k_t$ is sampled uniformly at random (*e.g.*, $k_t = 95$ leaves only 5 active variables). Conditional on S_t , the nonzero coefficients are drawn independently as $(\beta_t)_j \sim \mathcal{N}(0, 3)$, $j \in S_t$, with $(\beta_t)_j = 0$ for $j \notin S_t$. Conditioned on β_t , we generate a task-specific support set $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$, where covariates are sampled as $x_{t,n} \sim \mathcal{N}(0, I_p)$ and responses follow the linear model

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n}, \quad \varepsilon_{t,n} \sim \mathcal{N}(0, 1).$$

The number of observations per task is drawn uniformly as

$$N_t \sim \text{Discrete} - \text{Uniform}\{400, 401, \dots, 500\}.$$

Model. For each task t , we treat $\mathcal{D}_t$ as an unordered set of tokens $z_{t,n} = [x_{t,n}; y_{t,n}] \in \mathbb{R}^{p+1}$. Tokens are embedded into $\mathbb{R}^{d_{\text{model}}}$ and processed by an L -layer Transformer encoder ‘without positional encodings’, ensuring permutation equivariance across samples. A mean pooling operator produces a permutation-invariant task representation r_t , which is decoded by two heads: a magnitude head $\hat{\beta}_{\text{mag},t} \in \mathbb{R}^p$ and a sparsity gate $\hat{p}_t \in (0, 1)^p$. The final coefficient estimate is

$$\hat{\beta}_t = \hat{\beta}_{\text{mag},t} \odot \hat{p}_t.$$

Training objective. Given a batch of M tasks, we minimize prediction error using soft gating:

$$\mathcal{L} = \frac{1}{M} \sum_{t=1}^M \left(\frac{1}{N_t} \sum_{n=1}^{N_t} (y_{t,n} - x_{t,n}^\top \hat{\beta}_t)^2 \right).$$

Training meta-dataset construction. We generate a fixed meta-dataset of $T = 6000$ tasks spanning the grid of sparsity levels $\mathcal{K} = \{5, 10, 15, \dots, 100\}$ ($|\mathcal{K}| = 20$). For each $k \in \mathcal{K}$, we generate 300 independent tasks, yielding $T = 20 \times 300 = 6000$ tasks in total. Each task t is assigned a sparsity level $k_t \in \mathcal{K}$ and is generated according to the sparse linear model described above.

Per-task sampling. For a task with sparsity level k_t (percentage of zero coefficients), we sample an active support $S_t \subset \{1, \dots, p\}$ with $|S_t| = p - k_t$ uniformly at random, draw nonzero coefficients $(\beta_t)_j \sim \mathcal{N}(0, 1)$ for $j \in S_t$ (and $(\beta_t)_j = 0$ otherwise), and then generate a task-specific dataset $\mathcal{D}_t = \{(x_{t,n}, y_{t,n})\}_{n=1}^{N_t}$. The number of observations per task is drawn uniformly as $N_t \sim \text{Discrete} - \text{Uniform}\{400, 401, \dots, 500\}$, with covariates sampled as $x_{t,n} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, I_p)$ and responses generated as

$$y_{t,n} = x_{t,n}^\top \beta_t + \varepsilon_{t,n}, \quad \varepsilon_{t,n} \sim \mathcal{N}(0, 1).$$

Train–test split. After generating the full collection of $T = 6000$ tasks, we randomly shuffle task indices and split tasks into $T_{\text{train}} = 5400$ training tasks and $T_{\text{test}} = 600$ held-out test tasks. All models are trained only on the training tasks; evaluation is performed on the held-out tasks.

Performance. Figure 6 reports the across-epoch performance of the sparse recovery architecture under varying sparsity levels compared against the Lasso baseline. We measure cosine similarity between the predicted coefficient vector $\hat{\beta}$ and the ground-truth β , where the sparsity parameter k denotes the percentage of zero coefficients in β . The Lasso baseline demonstrates exceptional point-estimation accuracy, consistently achieving a cosine similarity close to 1 across all sparsity regimes. For our neural model, moderate sparsity levels (e.g., $k \leq 60\%$) yield high cosine similarity (> 0.85) even at early training stages (epoch 100), indicating accurate recovery of the dominant signal components. As sparsity increases (larger k), the recovery task becomes more challenging due to fewer informative features, leading to lower initial performance at epoch 100. However, continued optimization leads to consistent improvements, and by epoch 500, the model closely approaches the baseline

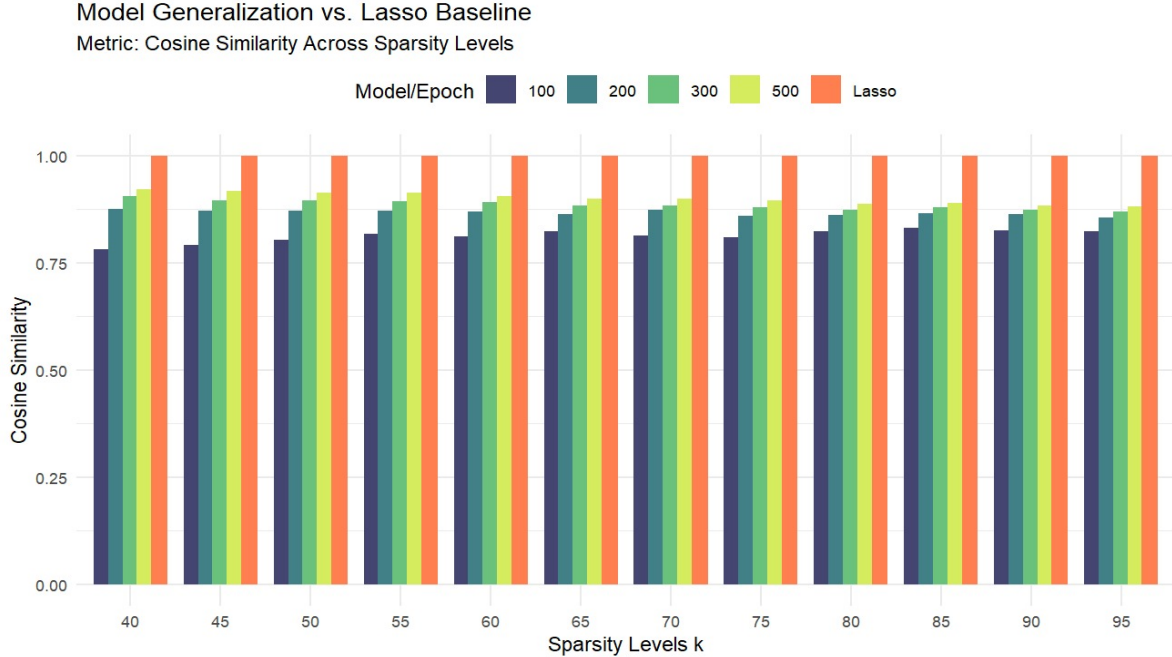


Figure 6: Cosine similarity between estimated and true regression coefficients across different sparsity levels k , where $k\%$ of the coefficients in the ground-truth parameter vector are zero. Bar clusters compare the neural network’s optimization progression at training epochs 100, 200, 300, and 500 against the baseline Lasso regression performance.

performance. Crucially, while Lasso solves an independent optimization problem for each task at inference time, the amortized model produces estimates in a single forward pass, making it well-suited to large-scale multi-task deployment.

While cosine similarity captures point-estimation accuracy, it does not reflect the stability of the recovered coefficients across different data realizations. To assess the reliability of the estimator beyond point estimates, we analyze the empirical standard deviation of the predicted coefficients, denoted as $\sigma(\hat{\beta})$. Table 3 reports this quantity as a function of the support set size N across sparsity percentages $k \in \{20, 50, 80, 95\}$. In low-sample regimes ($N = 50$), uncertainty is elevated across all sparsity levels, particularly when sparsity is high, reflecting the difficulty of disentangling signal from noise with limited observations. As the number of observations increases, the standard deviation decreases monotonically for all sparsity levels. At $N = 1000$, uncertainty is uniformly low, indicating stable coefficient recovery even under extreme sparsity.

4.4. Experiment IV: Topological mismatch in multimodality

We next consider a setting in which the posterior distribution over task parameters exhibits a substantial topological mismatch relative to standard unimodal assumptions. In particular, we study amortized inference when the parameter distribution is highly multimodal and supported on a non-convex manifold. This experiment evaluates whether neural inference models can identify and represent disconnected posterior geometry using only observed input-output samples.

Table 3: Standard deviation of predicted regression coefficients $\sigma(\hat{\beta})$ across varying support set sizes (N) and sparsity percentages (k). Here, k denotes the percentage of zero coefficients in the true parameter vector. The values reported here are calculated based on averaging the standard deviations obtained from 30 replications.

Support set size (N)	Sparsity percentage (k)			
	20	50	80	95
50	0.530	0.498	0.445	0.412
100	0.392	0.355	0.310	0.285
200	0.258	0.220	0.185	0.150
500	0.140	0.125	0.095	0.085
1000	0.062	0.055	0.048	0.042

Generative model. Let $\beta \in \mathbb{R}^2$ denote the task-specific regression parameter. We define a multimodal prior distribution consisting of $K = 8$ Gaussian components arranged uniformly on a circle of radius $R = 5.0$:

$$p(\beta) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(\beta \mid \mu_k, \sigma^2 I_2), \quad \mu_k = R \begin{bmatrix} \cos\left(\frac{2\pi k}{K}\right) \\ \sin\left(\frac{2\pi k}{K}\right) \end{bmatrix}.$$

Conditioned on β , observations are generated according to the same linear model used throughout the paper

$$y_n = x_n^\top \beta + \varepsilon_n, \quad x_n \sim \mathcal{N}(0, I_2), \quad \varepsilon_n \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

yielding a task-specific dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$.

Inference objective. Given $\mathcal{D}$, the goal is to approximate the posterior distribution $p(\beta \mid \mathcal{D})$, which inherits the multimodal structure of the prior while being shaped by the likelihood induced by the observed (x, y) pairs. Notably, the inference model has access only to the observed samples and must identify the full posterior geometry without explicit knowledge of the prior components.

Amortized conditional flow. We employ a conditional normalizing flow trained via Conditional Flow Matching to learn a mapping from observed datasets to posterior distributions. The input dataset $\mathcal{D}$ is first embedded into a fixed-dimensional context vector

$$\mathbf{r} = f_\psi(\mathcal{D}),$$

using a permutation-invariant encoder consistent with earlier experiments (Deep Sets). This representation summarizes the information contained in the observed (x, y) samples and conditions the inference model.

Let $\beta_t \in \mathbb{R}^2$ denote the state of the flow at time $t \in [0, 1]$, initialized from a base distribution $\beta_0 \sim \mathcal{N}(0, I_2)$. The evolution of β_t is governed by a time-dependent velocity field

$$\frac{d\beta_t}{dt} = v_\phi(t, \beta_t, \mathbf{r}), \quad (9)$$

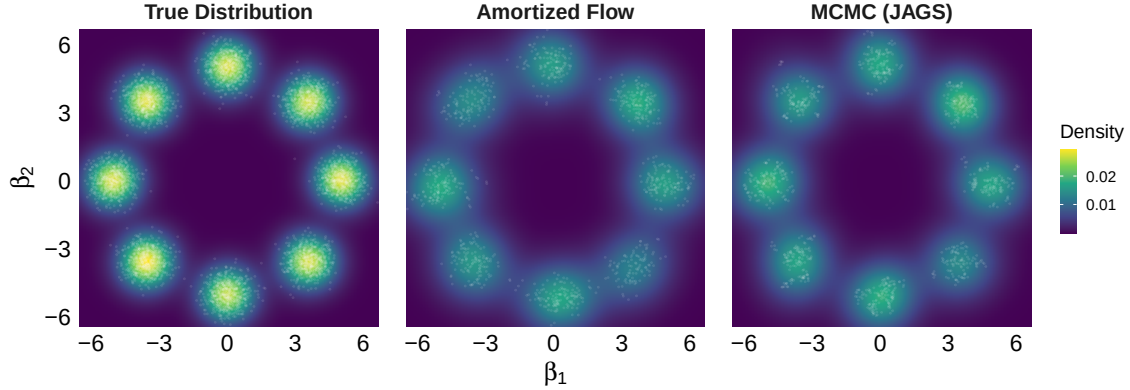


Figure 7: Posterior Sampling Benchmark: Visual comparison of the true multimodal distribution (Left), the Amortized Normalizing Flow approximation (Center), and the MCMC baseline (Right). The Amortized Flow successfully captures the complex multimodal geometry and separates the modes significantly faster than the baseline.

where v_ϕ is a neural network parameterized by ϕ . Integrating this ordinary differential equation yields

$$\beta_1 = \beta_0 + \int_0^1 v_\phi(t, \beta_t, \mathbf{r}) dt, \quad (10)$$

which represents a sample from the learned approximation to $p(\beta \mid \mathcal{D})$.

Posterior geometry and transport dynamics. Figure 7 compares samples from the true posterior distribution, the amortized flow approximation, and an MCMC baseline. The amortized model successfully allocates probability mass across all eight modes, demonstrating that it recovers the disconnected geometry of the posterior using only the observed (x, y) samples. To quantitatively compare the posterior approximations, we compute the Kullback–Leibler (KL) and Jensen–Shannon (JS) divergences between the true posterior and each approximation. The amortized flow yields a KL divergence of 0.2728 and a JS divergence of 0.0587, while the corresponding values for the MCMC approximation are 0.2544 and 0.0528, respectively.

Figure 8 visualizes the transport induced by the learned vector field. Samples drawn from the isotropic Gaussian base distribution are smoothly transformed over time into the multimodal posterior structure, confirming that the model learns a continuous deformation that respects the topology of the target distribution.

Computational efficiency. In addition to capturing complex posterior geometry, the amortized flow provides substantial computational benefits. Posterior samples are generated in 0.82 seconds per task, compared to 2.76 seconds for the MCMC baseline. Although the MCMC approximation is marginally closer to the true posterior according to these divergence measures, the differences are relatively small. In contrast, the amortized flow provides posterior samples through a single forward pass after training, resulting in substantially lower computational cost than running an MCMC algorithm for each new dataset.

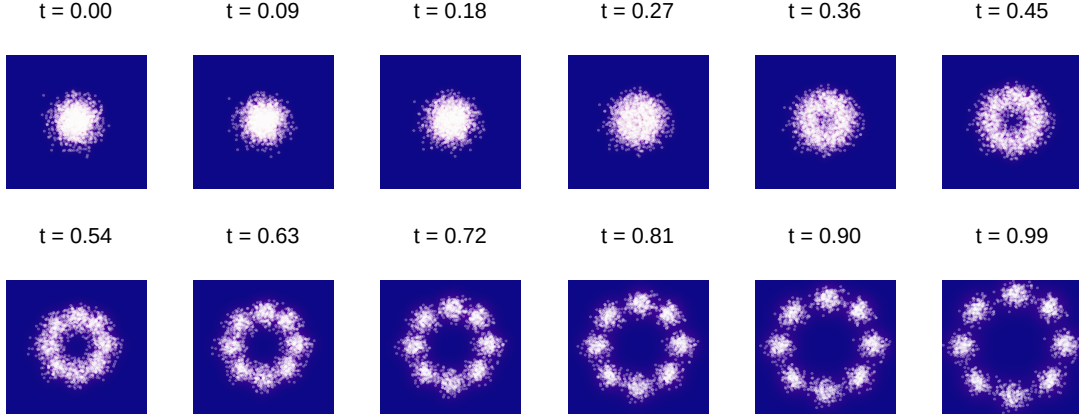


Figure 8: Visualization of the flow trajectory evolution. The horizontal panels illustrate the transformation of the initial particle distribution ($\theta_0 \sim \mathcal{N}(0, I)$ at $t = 0.00$) under the learned vector field over time. By $t = 1.00$, the particles have successfully converged to the target 8-mode posterior geometry.

5. Discussion and future research

Our experimental evaluation highlights the practical trade-offs of using permutation-invariant neural architectures for amortized inference. The primary advantage of these models is their computational efficiency during inference. As demonstrated in our sparse signal recovery (Section 4.3) and multimodal posterior experiments (Section 4.4), amortized models replace computationally expensive per-task optimizations, such as Lasso or MCMC, with a single forward pass. This allows for highly scalable inference, reducing posterior sampling time from 2.76 seconds to 0.82 seconds per task.

Furthermore, these architectures excel when tasks share a hidden structural geometry. In the latent structure recovery experiment (Section 4.1), both Deep Sets and the Set Transformer successfully pooled information across tasks to uncover hidden clusters, achieving parameter estimation errors substantially lower than those of classical per-task linear baselines (*e.g.*, Ridge regression) that fail to leverage cross-task information.

However, our results also emphasize that model selection must be carefully tailored to the specific application, as significant differences exist in training dynamics and robustness. Deep Sets provides rapid training convergence, reaching stable validation errors in fewer than 50 epochs (Section 4.2). In contrast, the Set Transformer requires extended optimization (over 80 epochs) to stabilize but ultimately achieves a lower asymptotic error. While the Transformer demonstrates superior Monte Carlo stability as the support set size increases, its architecture inherently requires larger data volumes and more intricate hyperparameter tuning. As observed in our distributional robustness evaluations, the Transformer is highly sensitive to its weights, leading to high prediction variance and difficulty generalizing across diverse, complex noise profiles.

Finally, these findings indicate that amortized neural inference is particularly promising for domains where classical distributional assumptions fail. The amortized flow model

successfully identified and represented a disconnected, non-convex, multimodal posterior without relying on unimodal approximations (Section 4.4). Future research will focus on scaling these architectures to higher-dimensional, real-world datasets characterized by extreme data sparsity and complex topological mismatches, investigating methods to stabilize Transformer training under heavy-tailed and asymmetric noise. Specifically, the use of MCMC is common in the literature focusing on large spatial and spatiotemporal data modeling using Gaussian processes, *e.g.*, under geostatistics (Gelfand and Schliep, 2016; Hazra *et al.*, 2025b), multi-type processes (Cisneros *et al.*, 2023; Mukherjee *et al.*, 2026), spatially varying coefficients models (Gelfand *et al.*, 2003; Bhowmik and Hazra, 2026), heavy-tailed processes (Hazra *et al.*, 2020, 2025a), nonstationary processes (Hazra and Huser, 2021), longitudinal datasets (Hazra *et al.*, 2019), and extreme value analysis (Jóhannesson *et al.*, 2022; Hazra *et al.*, 2023); amortized Bayesian inference is an emerging area of Bayesian statistics potentially suitable to handle such computationally challenging problems efficiently.

Acknowledgement

The authors would like to thank the Handling Editor and an anonymous Reviewer for their detailed constructive feedback, which helped to improve the quality and flow of the manuscript. The authors acknowledge the use of ChatGPT (OpenAI) and Grammarly for assistance with grammar correction, language refinement, and sentence restructuring. The scientific content, analysis, results, and conclusions presented in this paper are solely the responsibility of the authors.

Conflict of interest

The authors do not have any financial or non-financial conflicts of interest to declare for the research work included in this article.

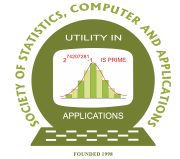
References

- Andoni, A. and Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, **51**, 117–122.
- Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. (2018). Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*.
- Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, **39**, 930–945.
- Bhowmik, S. and Hazra, A. (2026). A Bayesian latent Gaussian conditional autoregressive copula model for analyzing spatially-varying trends in rainfall. *Japanese Journal of Statistics and Data Science*, **9**, 99–131.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622.
- Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, **16**, 199–231.

- Cisneros, D., Gong, Y., Yadav, R., Hazra, A., and Huser, R. (2023). A combined statistical and machine learning approach for spatial prediction of extreme wildfire frequencies and sizes. *Extremes*, **26**, 301–330.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**, 303–314.
- Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian processes. In *Artificial Intelligence and Statistics (AISTATS)*, pages 207–215. PMLR.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059.
- Gelfand, A. E., Kim, H.-J., Sirmans, C., and Banerjee, S. (2003). Spatial modeling with spatially varying coefficient processes. *Journal of the American Statistical Association*, **98**, 387–396.
- Gelfand, A. E. and Schliep, E. M. (2016). Spatial statistics and Gaussian processes: A beautiful marriage. *Spatial Statistics*, **18**, 86–104.
- Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, **85**, 398–409.
- Gershman, S. J. and Goodman, N. D. (2014). Amortized inference in probabilistic reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
- Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356.
- Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR.
- Hazra, A. and Huser, R. (2021). Estimating high-resolution Red Sea surface temperature hotspots, using a low-rank semiparametric spatial model. *The Annals of Applied Statistics*, **15**, 572–596.
- Hazra, A., Huser, R., and Bolin, D. (2025a). Efficient modeling of spatial extremes over large geographical domains. *Journal of Computational and Graphical Statistics*, **34**, 795–811.
- Hazra, A., Huser, R., and Jóhannesson, Á. V. (2023). Bayesian latent Gaussian models for high-dimensional spatial extremes. In *Statistical Modeling Using Bayesian Latent Gaussian Models: With Applications in Geophysics and Environmental Sciences*, pages 219–251. Springer.
- Hazra, A., Nag, P., Yadav, R., and Sun, Y. (2025b). Exploring the efficacy of statistical and deep learning methods for large spatial datasets: A case study. *Journal of Agricultural, Biological and Environmental Statistics*, **30**, 231–254.
- Hazra, A., Reich, B. J., Reich, D. S., Shinohara, R. T., and Staicu, A.-M. (2019). A spatio-temporal model for longitudinal image-on-image regression. *Statistics in Biosciences*, **11**, 22–46.
- Hazra, A., Reich, B. J., and Staicu, A.-M. (2020). A multivariate spatial skew-t process for joint modeling of extreme precipitation indexes. *Environmetrics*, **31**, e2602.

- Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). NeuTra-lizing Bad Geometry in Hamiltonian Monte Carlo Using Neural Transport. *arXiv preprint arXiv:1903.03704*, .
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, 251–257.
- Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2022). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**, 5149–5169.
- Jóhannesson, Á. V., Siegert, S., Huser, R., Bakka, H., and Hrafnkelsson, B. (2022). Approximate Bayesian inference for analysis of spatiotemporal flood frequency data. *The Annals of Applied Statistics*, **16**, 905–935.
- Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.
- Kitaev, N., Kaiser, &., and Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30.
- Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.
- Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In *International Conference on Learning Representations*.
- Luccioni, A. S., Viguier, S., and Ligozat, A.-L. (2023). Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *Journal of Machine Learning Research*, **24**, 1–15.
- MacKay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, **4**, 448–472.
- Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, volume 27.
- Mukherjee, A., Hazra, A., and Vats, D. (2026). Scalable Bayesian inference for high-dimensional mixed-type multivariate spatial data. *Spatial Statistics*, **74**, 101020.
- Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, **9**, 141–142.
- Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, .
- Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, **22**, 1–64.
- Peyré, G. and Cuturi, M. (2019). Computational optimal transport with applications to data sciences. *Foundations and Trends® in Machine Learning*, **11**, 355–607.

- Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, .
- Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Köthe, U., and Bürkner, P.-C. (2024). Neural methods for amortised parameter inference. *arXiv:2404.12484*, .
- Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20.
- Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA.
- Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538.
- Richards, J., Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2024). Neural Bayes estimators for censored inference with peaks-over-threshold models. *Journal of Machine Learning Research*, **25**, 1–49.
- Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **71**, 319–392.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.
- Sainsbury-Dale, M., Zammit-Mangion, A., and Huser, R. (2024). Likelihood-free parameter estimation with neural Bayes estimators. *The American Statistician*, **78**, 1–14.
- Sainsbury-Dale, M., Zammit-Mangion, A., Richards, J., and Huser, R. (2025). Neural Bayes estimators for irregular spatial data using graph neural networks. *Journal of Computational and Graphical Statistics*, **34**, 1153–1168.
- Shmueli, G. (2010). To explain or to predict? *Statistical Science*, **25**, 289–310.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, **15**, 1929–1958.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.
- Wagstaff, E., Fuchs, F., Engelcke, M., Posner, I., and Osborne, M. A. (2019). On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pages 6487–6494. PMLR.
- Wand, M. P. and Jones, M. C. (1994). *Kernel Smoothing*. CRC press.
- Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, **26**, 359–372.
- Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30.
- Zammit-Mangion, A., Sainsbury-Dale, M., and Huser, R. (2025). Neural methods for amortized inference. *Annual Review of Statistics and Its Application*, **12**, 311–335.



Bayesian Feature Extraction using Gaussian and Diffused-gamma Priors for High Dimensional Spatio-Temporal Data

Garrett Frady¹, Dipak K. Dey¹ and Shariq Mohammed^{2,3}

¹*Department of Statistics, University of Connecticut, Storrs, Connecticut, U.S.A.*

²*Department of Biostatistics, Boston University, Boston, Massachusetts, U.S.A.*

³*Rafik B. Hariri Institute for Computing, Computational Science and Engineering, Boston University, Boston, Massachusetts, U.S.A.*

Received: 21 December 2025; Revised: 29 May 2026; Accepted: 28 July 2026

Abstract

High-dimensional data with sparse structure and spatio-temporal dependence arise in many scientific domains. We develop a Bayesian feature-extraction framework for spatio-temporal settings that employs Gaussian and Diffused-gamma priors to induce structured sparsity. The modeling framework specifies a general likelihood via Bregman divergence, enabling compatibility with a range of loss functions and measurement models. Posterior computation is carried out via Markov chain Monte Carlo (MCMC), and we introduce a two-stage feature-extraction procedure based on posterior samples to stabilize selection across space and time. We illustrate the method with a multi-subject electroencephalography (EEG) case study examining the relationship between chronic alcohol exposure and activity in different brain regions. We first fit binary classification models at each time point, then use false discovery rate-controlled screening and subsequent clustering in a two-stage feature-extraction pipeline to identify active brain regions. The analysis demonstrates that our proposed priors improve recovery of sparse features and enhance interpretability in the presence of spatio-temporal dependence. The framework is broadly applicable to high-dimensional, structured problems where accurate feature selection and inference are required. The code to implement the model is publicly available via GitHub.

Key words: Bregman divergence; False discovery rate; Markov chain Monte Carlo (MCMC); Variable selection; Shrinkage priors.

AMS Subject Classifications: 62F15, 62H30, 62P10, 62-08

1. Introduction

Due to vast improvements in technology and data collection methods, performing statistical analyses commonly involves modeling massive amounts of data containing a significantly larger number of parameters than outcomes. In such cases, classical statistical models are inadequate and some form of regularization, dimension reduction, or variable selection is required. Additionally, data often includes matrix covariates with correlations extending across multiple dimensions. For example, predicting natural disasters and discovering global warming trends require analyzing three-dimensional data capturing dynamic spatial patterns changing over time. Beyond the evolution of environmental phenomena, spatio-temporal data analysis can provide valuable insights into distribution and determinants of public health outcomes. Measurements from neuroimaging procedures, such as electroencephalography (EEG) and functional magnetic resonance imaging (fMRI), may be collected in a spatio-temporal pattern to examine statistical dependencies between electrical signals at different regions of the brain. Understanding how locations of the brain communicate plays a major role in detecting early onset of mental-related illnesses, including dementia, opioid addiction, alcoholism, and attention deficit hyperactivity disorder (ADHD). Although spatio-temporal data arises across a wide variety of scientific domains, this paper develops the proposed framework exclusively using EEG data as the motivating example. The minimum data structure required for the framework to apply is a three-dimensional array of measurements over n subjects, L spatial locations, and τ time points, paired with a subject-level response variable, where $n \ll L \times \tau$.

Designing a supervised predictive model of subject-level behavior using neuroimaging data comes with its difficulties. Brain imaging techniques generally produce tens of thousands to hundreds of thousands of measurements for each subject, acting as covariates, and only consist of tens to hundreds of subjects because of the invasive nature of brain imaging procedures. Different natural biological behavior results in high subject-to-subject variability. The spatio-temporal structure yields two-dimensional predictors and correlation exists across multiple dimensions. Such a correlation structure may not be fully captured in a computationally efficient manner through traditional statistical or machine learning methods. The sheer size of the data will likely be too large for the memory available on a single computer node. Conducting analysis in an efficient manner requires efficient and scalable strategies.

We present a case study utilizing EEG data as our motivating example. The primary goal of this paper is to identify active brain regions that are associated with cognitive or behavioral states implying potential chronic exposure to alcohol. Then, classify subjects as at risk, or not, of alcoholism according to their EEG measurements. There have been approaches in literature that address one or both of these objectives. We conduct a thorough literature review to examine prior methodologies address one or both of these objectives, and highlight new contributions in our approach. We propose a computationally efficient Bayesian-feature extraction method utilizing the Gaussian and Diffused-gamma (GD) prior (Goh and Dey, 2018) by constructing models locally at each time point. Covariates in each model are electrical signals observed at locations. Estimation is carried out using the Markov chain Monte Carlo (MCMC) algorithm and the Metropolis-Hastings algorithm within Gibbs sampler. MCMC samples obtained from the GD model are used to develop a two-stage feature extraction process to make final estimations and select active locations,

and a subject-level weighted prediction process to accurately classify subjects.

A substantial body of research has been allocated to developing accurate and implementable methods to handle data with the complex characteristics previously described. The most common approach to reduce the dimension of a problem is vectorizing the matrix covariates. Since the number of covariates, p , in neuroimaging data is enormous, vectorizing the covariates will not bypass the high dimensional issue. To address this and the difficulties described in Section A.1 of the Annexure, Hu and Allen (2015) developed a local-aggregate modeling approach applied to EEG data. By constructing models at each location, the dimension of the predictor vectors is equivalent to the number of time points, tending to the order of hundreds up to thousands. Thus, the local modeling approach requires additional effort to reduce the dimension of the supervised learning problem.

1.1. Sparsity, feature extraction, penalized loss function estimator

Even with large p , in several real-world applications, only a small portion of the predictors may have a significant impact on the response (leading to sparsity within high dimensional settings). For example, in neuroimaging research, only a few of the brain locations may be significantly correlated to predicting early onset of a mental-related illness. Dimension reduction is a matter of removing irrelevant predictors from the model by forcing corresponding coefficients to zero. To this end, multiple studies (Tibshirani, 1996; Fan and Li, 2001; Zou, 2006) have employed the penalized loss function (PL) estimator to achieve simultaneous estimation and feature extraction. The PL estimator is defined as

$$\hat{\beta}_{PL} = \underset{\beta}{\operatorname{argmin}} [L(\mathbf{y}, \mathbf{h}(\mathbf{X}\beta)) + Pe(\beta|\lambda)], \quad (1)$$

where $L(\cdot, \cdot)$ denotes a loss function, $Pe(\cdot|\lambda)$ is a penalty function, and $\lambda \geq 0$ is a regularization parameter intended to control the degree of penalization. An abundance of research has been devoted to developing sparsity inducing penalty functions. First formally developed by Akaike (1974) and further investigated by Schwarz (1978), the ℓ_0 -norm penalty, $\ell_0(\beta|\lambda) = \lambda \sum_{j=1}^p \mathbb{1}_{(\beta_j \neq 0)}$, where $\mathbb{1}_{(\cdot)}$ is an indicator function, was introduced to induce sparsity into $\hat{\beta}_{PL}$. Quantifying $\hat{\beta}_{PL}$ remains computationally cumbersome caused by the discontinuous nature of indicator functions. Many methods followed in attempt to improve previous methods (see examples in Section A.1 of the Annexure).

Although PL point estimators are attractive, Casella *et al.* (2010) pointed out that frequentist LASSO methods are unable to produce valid standard errors, rendering them unable to accurately quantify uncertainty in the estimators. In turn, Bayesian techniques became popular in sparse, high dimensional literature. An advantage of our local Bayesian modeling approach over the frequentist local modeling approach in Hu and Allen (2015) is that we are able to quantify the uncertainty in the estimators through the posterior distribution of β . A potential limitation of Bayesian approaches is that for large values of p , the MCMC sampling algorithm proves to be computationally inefficient and ineffective (Ročková and George, 2014). To overcome this deficiency in the analysis of EEG data, Mohammed *et al.* (2019) proposed a Bayesian local-aggregate modeling strategy by constructing continuous spike-and-slab models at each time point. Building models locally at time points instead of locations reduces the dimension further, since EEG data has high temporal resolution but low spatial resolution. An advantage of our GD model over spike-and-slab models is that we

have less hyperparameters to tune and do not require extensive computational resources or sensitivity analysis to find optimal values.

Novelty of the proposed methods. Existing approaches, such as Goh and Dey (2018), rely on a component-wise iterated conditional modes (ICM) algorithm (Besag, 1986) within MCMC and impose sparsity through a hard threshold determined by a regularization parameter, λ . Our method addresses these limitations by introducing a fully Bayesian framework that avoids ad hoc thresholding and manual tuning. Specifically, we (i) place a non-informative gamma prior on λ rather than performing a grid search as in Goh and Dey (2018), reducing computational burden and improving robustness; and (ii) develop a novel two-stage feature extraction pipeline based on posterior samples, combining FDR-controlled screening with spatial-temporal clustering and weighted subject-level prediction. This integration of structured priors, adaptive sparsity induction, and posterior-driven feature selection represents a significant advancement over existing methods, enabling more reliable recovery of sparse features in high-dimensional spatio-temporal settings.

The outline for the remaining sections in this paper is as follows. In Section 2, we describe the EEG data used in our case study, the procedure and characteristics of its measurements. In Section 3, we explain our local Bayesian modeling framework. We define the GD prior and its relationship with a continuous-and-differentiable ℓ_0 -norm approximation, discuss the role of the class of Bregman divergence measures in our approach, present a computationally feasible MCMC algorithm, and give background theory for posterior computation under a specified model with the GD prior. Additionally, we describe our two-stage feature extraction process and subject-level weighted prediction process. In Section 4, we conduct a simulation study and an EEG case study. We compare the performance of our GD method with existing methods in the literature. Finally, we provide concluding remarks and potential future developments in Section 5. Additional details and proofs are provided in the Annexure.

2. Electroencephalography data

EEG is a popular non-invasive brain-imaging technique administered as a way to record electrical brain signals over a specified interval of time via electrodes attached to different locations of a subject's scalp. Consequently, researchers commonly conduct EEG analysis to examine the functional connectivity between different parts of the brain in response to a stimulus. From a statistical viewpoint, detecting associated brain regions in such a manner may be viewed as a feature extraction process. The non-invasive aspect of EEG leads to a low signal-to-noise ratio, because the subjects' skull, connective tissue, and body fluid between the brain and EEG electrodes significantly debilitates the electrical signal (Tam *et al.*, 2019).

In our case study, we use open access EEG data obtained from <https://archive.ics.uci.edu/ml/datasets/eeg+database>. A clean version of this data set is available in an R package *SpikeSlabEEG* on GitHub (Mohammed *et al.*, 2019). The data surfaces from a large study of 120 trials for 122 subjects to investigate EEG correlates of genetic predisposition to alcoholism. Measurements were taken from 64 electrodes placed on the subjects' scalp by sampling at 256 Hz for 1 second (3.9-ms epochs). Photos from Snodgrass and Vanderwart (1980) were shown to each subject as a stimulus. In our analysis, we consider

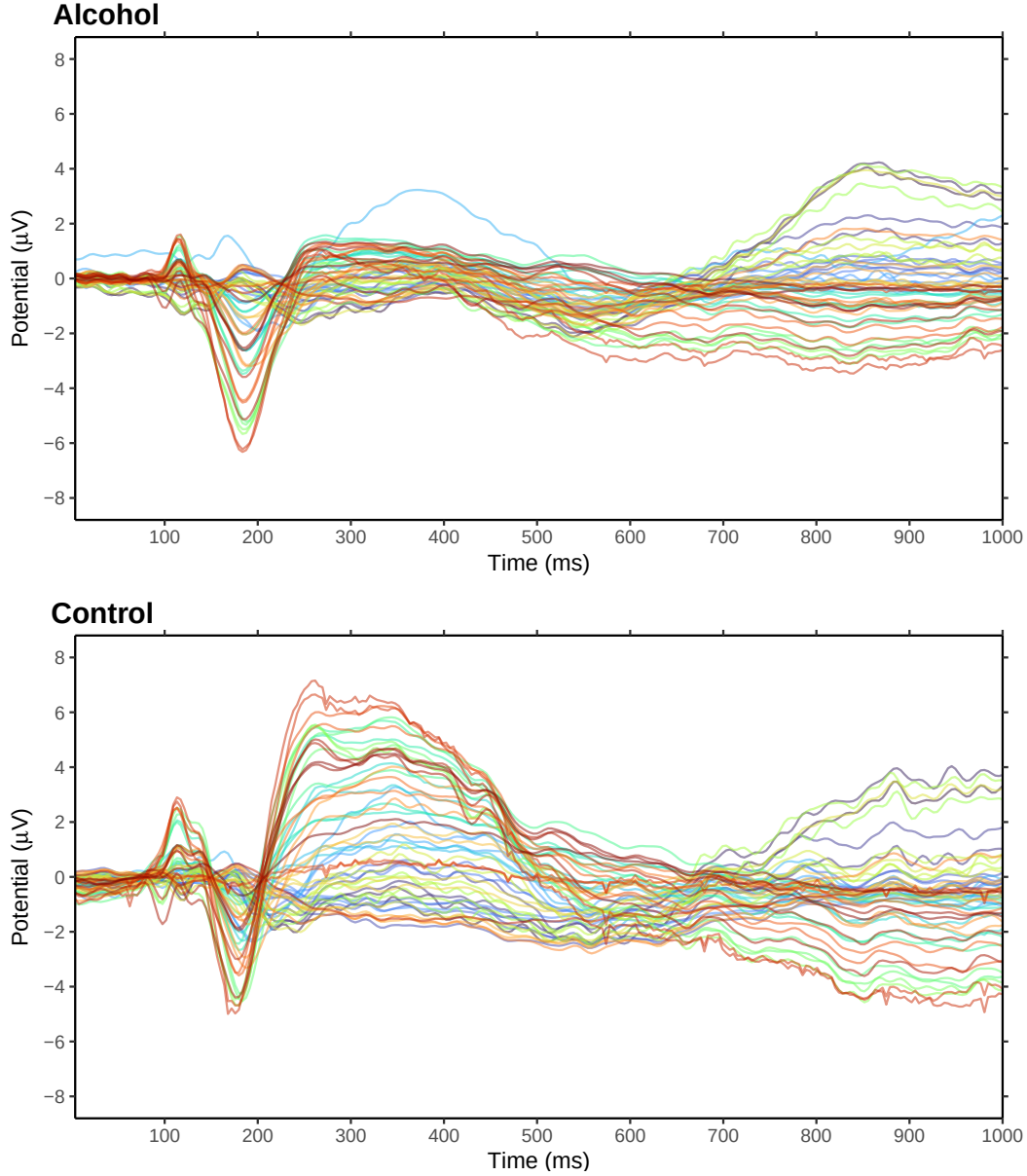


Figure 1: ERP plots for the alcohol and control group

57 of the 64 locations as they are the only locations with coordinates associated to them. Hence, the predictor tensor, χ , has dimensions $122 \times 57 \times 256$. As EEG captures alterations in voltage resulting from the movement of electrically charged ions across neuronal cell membranes, the measurements are continuous. In order to visualize the spatio-temporal behavior of the data, we present event related potential (ERP) plots and scalp maps. For more information on these visualization techniques, see Section A.4 of the Annexure.

Figure 1 shows the ERP plots for the alcoholic group (77 subjects) and the control group (45 subjects). The two plots consist of 57 time series curves representing the average potential of subjects within a group at 57 locations over 256 time points. Each of the 57 locations correspond with the same unique colored curve in both plots. The scalp maps in Figure 2 represent the distribution of the average EEG signals at three specified time points.

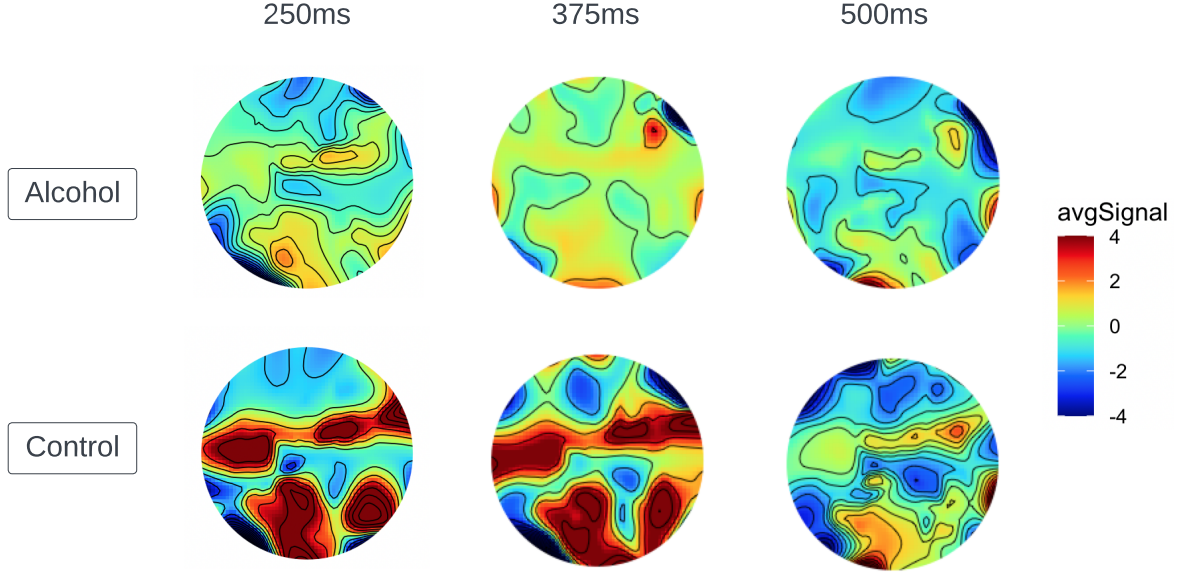


Figure 2: Scalp maps at 250ms, 375ms, and 500ms for the alcohol group and the control group

Since we employ local models at each time point, visualizing the localized distribution of mean potential across all subjects at a specific time point is beneficial. Looking at Figure 1 and Figure 2, the magnitude of the average signals for the control group within the 225-500ms time interval are distinctively larger than that of the alcohol group, especially in the outer regions of the brain in locations illustrated with red shading on the scalp maps. The observed disparities serve as both inspiration and a crucial prerequisite, illuminating the potential not only to decipher functional connections between brain regions but also to classify subjects into either the alcohol or control group. Furthermore, the ERP plots and scalp maps in Figure 1 and Figure 2 show that the set of locations with higher magnitudes of average potential varies over time. That is, different locations are active at different time points.

3. Local Bayesian modeling framework

As a way of bypassing the curse of dimensionality in high dimensional spatio-temporal data, we construct subject-level Bayesian classification models through logistic regression, locally at each time point. The GD prior is assigned to model coefficients, which correspond to locations. Although the GD prior inherently possesses shrinkage and selection properties, the GD prior cannot produce sparse solutions on its own. As a result, we implement a two-stage feature extraction process, with the first stage being a hard thresholding technique based on controlling the false discovery rate (FDR). The second stage consists of clustering locations based on the area under the curve computed from plotting the estimated coefficient vectors of each location. We aggregate the predictions of the local models over time to make final predictions. In the scope of this modeling perspective, we develop a general likelihood function, define the GD prior distribution, specify priors on the hyperparameters, and demonstrate the derivation of the posterior distribution.

3.1. GD prior specification and posterior computation

Following the formulation of $\hat{\beta}_{PL}$ in Eq. (1), we choose the wide-ranging class of Bregman divergence measures as our loss function. Depending on the form of the convex function, Bregman divergence measures reduce to well-known loss functions (Goh and Dey, 2018). See Table A.1 in Section A.2.1 of the Annexure for examples. To develop a general likelihood function, we highlight the *duality property* (explained in Section A.2.1 of the Annexure) between the loss function and the likelihood function. We define our likelihood function as $f_\psi(\mathbf{y}|\boldsymbol{\beta}) \propto \exp\{-BD_\psi(\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta}))\}$, with the assumption that $f_\psi(\mathbf{y}|\boldsymbol{\beta})$ is integrable. Banerjee *et al.* (2005) showed that there is a bijection between the natural exponential family and the class of Bregman divergence measures. The flexibility of the class of Bregman divergence measures allows us to handle a variety of response variables and likelihood functions. See Table A.2 in Section A.2.1 of the Annexure for examples. With the EEG data analysis, one of our goals is binary classification of subjects. Thus, we use the Bregman divergence measure uniquely defined for the Bernoulli family of distributions, $z_1 \log(\frac{z_1}{z_2}) + (1 - z_1) \log(\frac{1-z_1}{1-z_2})$, which leads to our likelihood function being proportional to the Bernoulli likelihood function (see Section A.2.1 of the Annexure).

The GD prior is defined as follows (Goh and Dey, 2018)

$$\pi(\boldsymbol{\beta}, \mathbf{d}) \propto \pi_G(\boldsymbol{\beta}|\mathbf{d})\pi_D(\mathbf{d}), \quad (2)$$

$$\pi_G(\boldsymbol{\beta}|\mathbf{d}) \propto \prod_{j=1}^L \left[d_j^{1/2} \exp \left\{ -\frac{d_j}{2} \beta_j^2 \right\} \right], \quad (3)$$

$$\pi_D(\mathbf{d}) \propto \prod_{j=1}^L \left[d_j^{\lambda-1/2} \exp \left\{ -\frac{\tau_0^2}{2\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\}} d_j \right\} \right], \quad (4)$$

where $\tau_0 > 0$ is chosen to be sufficiently small. A critical piece of the GD prior is that it leads to a continuous and differentiable approximation of the ℓ_0 -norm penalty, which acts as the penalty function in Eq. (1). Goh and Dey (2018) introduced an ℓ_0 -norm approximation, referred to as the GD penalty, $\tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0) = \lambda \sum_{j=1}^L \frac{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2}{\tau_0^2 + \{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2}$, where $\mathbf{X}_{[j]}$ denotes the j^{th} column of the design matrix $\mathbf{X}$. We can see that for given values of the tuning parameter λ and coefficient vector $\boldsymbol{\beta}$, $\tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0) \rightarrow \lambda \sum_{j=1}^L \mathbb{1}_{(\beta_j \neq 0)}$ as $\tau_0 \rightarrow 0$. Furthermore, $\tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0)$ is continuous and differentiable everywhere. Of the many shrinkage and selection operators developed to induce sparsity into the PL estimator, Dicker *et al.* (2013) introduced a continuous approximation of the ℓ_0 -norm penalty called the seamless- ℓ_0 (SELO) penalty. However, it has restrictions in that it is not differentiable everywhere, lending an advantage to the GD penalty over the SELO penalty. We define the SELO penalty and provide a numerical demonstration of the advantage the GD penalty has over the SELO penalty in Section A.2.2 of the Annexure. A disadvantage of the GD penalty function is that it can't induce sparsity into the estimator unless $\tau_0 = 0$. We overcome this difficulty through our two-stage feature extraction process.

In our GD approach, the hyperparameter λ governs the degree of sparsity in the estimator of $\boldsymbol{\beta}$. Prior work, such as Goh and Dey (2018), selects λ via a grid search based on marginal likelihood, which introduces substantial computational overhead and relies on a frequentist tuning procedure within an otherwise Bayesian model. We propose a fully

Bayesian alternative by assigning a gamma prior to λ , with density $\pi(\lambda) \propto \lambda^{\alpha_1-1} \exp\{-\alpha_2\lambda\}$, where $\alpha_1 > 0$ and $\alpha_2 > 0$ are shape and rate parameters, respectively. This prior is chosen to be weakly informative—flat enough to avoid imposing strong assumptions, yet structured to reflect the practical insight that larger λ induces greater sparsity. By centering the prior's mode near zero (but not at zero), we encourage moderate sparsity without collapsing all coefficients. This innovation eliminates the need for computationally expensive grid searches, preserves Bayesian coherence, and improves robustness by integrating uncertainty about λ directly into posterior inference.

When latent variables are introduced into regression models, common practice is to integrate out the latent variable from the likelihood function and conduct inference only on the parameter of interest. In Bayesian regression models, the latent variable is integrated out of the posterior distribution and posterior inference proceeds only for the parameter of interest. Rather than integrating out $\mathbf{d}$, an essential component of our GD approach is to substitute $\mathbf{d}$ with its MAP estimator. Without this aspect of our approach, we lose the crucial connection between the GD prior and the GD penalty function described in Lemma 2, Remark 1, and their proofs in Section A.5 of the Annexure. Using these justifications, we can see that the GD estimator is a nice approximation of the PL estimator with a Bregman divergence measure as the loss function and $\tilde{\ell}_0(\cdot|\lambda, \tau_0)$ as the penalty function. Tibshirani (1996) pointed out that in a Bayesian framework, the PL estimator in Eq. (1) can be viewed as the MAP estimator of the posterior distribution given by $\pi(\boldsymbol{\beta}|\mathbf{y}) \propto f(\mathbf{y}|\boldsymbol{\beta})\pi(\boldsymbol{\beta}) \propto \exp\{-[L(\mathbf{y}, h(\mathbf{X}\boldsymbol{\beta})) + Pe(\boldsymbol{\beta}|\lambda)]\}$.

Since $\pi_{GD}(\boldsymbol{\beta}, \mathbf{d}) \propto \pi_G(\boldsymbol{\beta}|\mathbf{d})$ with respect to $\boldsymbol{\beta}$, the posterior distribution of $\boldsymbol{\beta}$ can be viewed as $\pi(\boldsymbol{\beta}|\mathbf{y}, \hat{\mathbf{d}}) \propto f_\psi(\mathbf{y}|\boldsymbol{\beta})\pi_{GD}(\boldsymbol{\beta}, \hat{\mathbf{d}}) \propto f_\psi(\mathbf{y}|\boldsymbol{\beta})\pi_G(\boldsymbol{\beta}|\hat{\mathbf{d}})$, where $\hat{\mathbf{d}}$ is the MAP estimator of $\mathbf{d}$. We outline the propriety of the GD posterior in Section A.2.3 of the Annexure. Due to the high dimensionality of our parameter space, we utilize the univariate full conditional distributions, given by $\pi(d_j|\text{others}) \propto d_j^\lambda \exp\{-\frac{\tau_0^2/[(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}] + \beta_j^2}{2} d_j\}$ and $\pi(\beta_j|\text{others}) \propto \exp\{-BD_\psi(\mathbf{y}, h(\mathbf{X}\boldsymbol{\beta})) - \frac{d_j}{2}\beta_j^2\}$, $j = 1, \dots, L$. Note, it is possible to express $\pi(d_j|\text{others})$ in closed form, and due to the smoothness of $\pi(\beta_j|\text{others})$, we can implement the Newton-Raphson algorithm to find its mode. Prior knowledge about β_j is represented through $\beta_j|\hat{d}_j \stackrel{\text{ind.}}{\sim} N(0, 1/\hat{d}_j)$. Lemma 2 in Section A.5 of the Annexure shows the form of $\hat{d}_j$. The posterior distribution of lambda is $\pi(\lambda|\text{others}) \propto \lambda^{\alpha_1-1} \exp\{-(\alpha_2 - \sum_{j=1}^L \log(d_j))\lambda\}$. For the remainder of this section, we demonstrate how we utilize the Metropolis-Hastings algorithm within a component-wise Gibbs sampler to carry out the MCMC algorithm.

MCMC sampling procedure. We implement the MCMC algorithm to obtain samples from our target posterior. Goh and Dey (2018) implement a component-wise ICM algorithm within the MCMC algorithm to compute the MAP estimate of $\boldsymbol{\beta}$. For each iteration of the algorithm, the cutoff is defined as $\xi_j = \frac{2\lambda}{\tau_0^2/[(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}] + \hat{\beta}_j^2}$, $j = 1, \dots, L$, which corresponds to a 95% credible interval when $\hat{\beta}_j = 0$. As a result of the hard thresholding, $\hat{\boldsymbol{\beta}}$ contains exact zeros. However, we found their approach induces strong sparsity into the estimator. Consequently, we propose an entirely Bayesian MCMC sampling approach, inducing sparsity into the estimator in the first stage of feature extraction.

Modified version of Metropolis-Hastings algorithm. Due to the high dimensionality of the parameter space, we additionally implement a component-wise Gibbs sampler by

iteratively generating β_j 's from their component-wise full conditionals. Since a closed form of the full conditional distribution of β_j may not exist, we propose to use the Metropolis-Hastings algorithm within the Gibbs sampler, where the move from the current state $\beta_j^{(t)}$ to a new state $\beta_j^{(t+1)}$ is generated from the proposal density $p(\beta_j^{(t+1)}|\beta_j^{(t)})$. Our Metropolis-Hastings algorithm differs from Goh and Dey (2018), because we don't implement the ICM algorithm. Let $\boldsymbol{\beta}_{(j)}^{(t)} = (\beta_1^{(t+1)}, \dots, \beta_{j-1}^{(t+1)}, \beta_j^{(t)}, \beta_{j+1}^{(t)}, \dots, \beta_L^{(t)})$ denote the coefficient vector at the current state and $\boldsymbol{\beta}_{(j)}^{(t+1)} = (\beta_1^{(t+1)}, \dots, \beta_{j-1}^{(t+1)}, \beta_j^{(t+1)}, \beta_{j+1}^{(t)}, \dots, \beta_L^{(t)})$ denote the coefficient vector at the next state. We define the proposal density to be Gaussian with mean $\beta_j^{(t)}$ and variance $\delta[\frac{\partial^2}{\partial \beta_j^2} BD_\psi(\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta}))|_{\boldsymbol{\beta}=\boldsymbol{\beta}_{(j,L)}^{(t)}} + \hat{d}_j]^{-1}$, owing to the form of the full conditional of β_j . Here, $\delta > 0$ is a predetermined constant to control the acceptance rate. For all $1 \leq j \leq L$, the proposed moves from state to state are accepted with probability

$$\alpha = \min \left\{ 1, \frac{f_\psi(\mathbf{y}|\boldsymbol{\beta}_{(1,j)}^{(t+1)}) \pi_{GD}(\boldsymbol{\beta}_{(1,j)}^{(t+1)}) p(\beta_j^{(t)}|\beta_j^{(t+1)})}{f_\psi(\mathbf{y}|\boldsymbol{\beta}_{(j,L)}^{(t)}) \pi_{GD}(\boldsymbol{\beta}_{(j,L)}^{(t)}) p(\beta_j^{(t+1)}|\beta_j^{(t)})} \right\}.$$

3.2. Two-stage feature extraction

A major goal of our Bayesian feature extraction process is to identify active regions of the brain according to the magnitude of their electrical signal. We propose a two-stage feature extraction process to obtain final estimates and select active locations. Although the MCMC sampling approach doesn't produce exact 0 estimates, the inherent feature extraction nature of the GD prior shrinks coefficients of smaller magnitude, while preserving coefficients of larger magnitude. Sparsity is induced into the estimator through a false discovery rate (FDR) thresholding approach in stage one. Coefficients which were preserved through shrinkage will remain preserved, whereas those which were reduced toward 0 will be dropped. Since the covariates in the local models are locations, performing the first stage in this way accounts for the fact that the set of locations selected is not uniform over all time points. In the second stage, we use a k-means clustering approach on the area under the curve formed by the estimated coefficient vectors of locations obtained from stage one.

First-stage feature extraction with FDR control. The first stage of the feature extraction process is a FDR-based thresholding approach involving Bayesian model averaging and MAP estimates (Morris *et al.*, 2008; Mohammed *et al.*, 2021). We consider N MCMC samples of $\boldsymbol{\beta}_t = (\beta_{1t}, \dots, \beta_{Lt})^T$, $\{\boldsymbol{\beta}_t^{(s)} : s = 1, \dots, N\}$, for $t = 1, \dots, \tau$ from the GD posterior distribution. For each location $l = 1, \dots, L$, we can calculate posterior inclusion probabilities $p_{lt} = \frac{1}{N} \sum_{s=1}^N \mathbb{1}_{(|\beta_{lt}^{(s)}| \leq c)}$ from the observed MCMC samples for some constant $c > 0$. This pooled statistic can be interpreted as the local FDR (Morris *et al.*, 2008), and $(1 - p_{lt})$ is the probability that location l at time point t is active. For coefficients β_{lt} close to 0, p_{lt} is large so that the probability of the coefficient being significant is small. On the other hand, p_{lt} is small for coefficients of higher magnitude, indicating a high probability of the coefficient being significant. The major question is, what values of p_{lt} are considered small, and what values are considered large? We choose a threshold ψ_α such that the overall average FDR is controlled at level α . The coefficient β_{lt} is said to be significant if $p_{lt} < \psi_\alpha$. By controlling the overall average FDR at level α , we only allow for $100\alpha\%$ of the pairs in $\{(l, t) : p_{lt} < \psi_\alpha\}$ to be false-positive inclusions. The threshold ψ_α is computed by first sorting the posterior

inclusion probabilities p_{lt} over all locations and time points. We denote the sorted probabilities by $p_{(k)}$, for $k = 1, \dots, L\tau$, and let $u = \{K | \frac{1}{K} \sum_{k=1}^K p_{(k)} \leq \alpha\}$. We define $\psi_\alpha = p_{(u)}$, and according to our approach at level α , the set $\{(l, t) : p_{lt} < \psi_\alpha\}$ is considered active.

Second-stage feature extraction with k-means clustering. After the first stage, we have a new vector of coefficient estimates at each time point, $\hat{\beta}_t^*$, which contains exact zero values. Looking at all time points, we have a matrix of zero and nonzero coefficient estimates, say $\hat{\beta}^*$. Specifically, $\hat{\beta}_{lt}^* = 0$ implies location l at time point t is inactive, and $\hat{\beta}_{lt}^* \neq 0$ implies location l at time point t is active. In the second stage of the feature extraction process, we are focusing on the individual locations, represented by rows of $\hat{\beta}^*$, say $\hat{\beta}_l^*$. As a consequence of inducing sparsity into the estimator in the first stage, active locations will have distinctively more non-zero entries than inactive locations. To extract active locations, we start by creating a line plot where the x-axis represents each $\hat{\beta}_{lt}^*$ over time and the y-axis represents the magnitude of the estimates. From here, we compute the area of the geometric shapes created between each pair of points in the line plot. We refer to the area under the curve formed by $\hat{\beta}_l^*$ as the sum of the areas of the geometric shapes. The area under the curve formed by active locations will be notably larger than that of inactive locations. Active locations will demonstrate longer streaks of successive nonzero estimates over time. Conversely, inactive locations will show a flatter pattern, emerging from several streaks of 0 over time. Locations with substantially large magnitude coefficients for only a short period of time will also have larger areas under the curve. Finally, we apply k-means clustering to the areas under curve formed by location vectors. Although k-means is sensitive to outliers, particularly when $k = 2$, there are no outliers after the first stage. We conclude locations in the cluster with the larger centroid are active. Since the first stage initiates separation between active and inactive locations, k-means with $k = 2$ clusters is more likely to converge to centroids that accurately represent the centers of the two clusters.

3.3. Subject-level prediction

Considering we construct models locally at time points, each local model is able to produce its own predicted binary response. Having all τ models predict the same binary response is extremely improbable. At each time point, the i th subject is assigned a prediction probability, $\hat{p}_{it} = h(\mathbf{x}_{it}^T \hat{\beta}_t)$, where $\hat{\beta}_t$ is the estimated coefficient vector at time point t , and represents the likeliness of the i th subject classified as a 1 over a 0. Specifically, $\hat{p}_{it}$ close to 1 is strong indication to classify the subject as $\hat{y}_{it} = 1$ at time point t and $\hat{p}_{it}$ close to 0 is strong indication to classify the subject as $\hat{y}_{it} = 0$ at time point t .

Final predictions for each subject in such scenarios have been made using the length of longest run of the local predictions $\hat{y}_{it}$ (Mohammed *et al.*, 2019). However, by assigning subject i as 1 with probability $\hat{p}_{it}$ and 0 with probability $1 - \hat{p}_{it}$, additional randomness is incorporated into the prediction process, hampering evaluations based on longest runs. For example, if $\hat{p}_{it} = 0.8$, then there is still 0.2 probability of subject i being classified as a 0. The effectiveness of the length of the longest run is significantly compromised by misclassification in local predictions. More so, clear differences in electrical signal, between the control and alcohol group, exist within a smaller interval of time than the final stretch of time where the two groups show extremely similar behavior (see Figure 1).

Instead, we construct local weights, w_{it} , with the goal of allocating higher priority

to local models with more certainty of classifying the subject as a 0 or 1. Local models corresponding to values of $\hat{p}_{it}$ closer to either extreme are granted increased emphasis over those corresponding to $\hat{p}_{it}$ in the vicinity of 0.5. Local models yielding $\hat{p}_{it}$ near 0.5 have a high degree of uncertainty resulting from the considerable amount of noise debilitating the electrical signals. These time points are practically negligible and should not play a major role in the prediction process. For all t , the weights for subject i are defined as $w_{it} = \frac{(\hat{p}_{it}-0.5)^2}{\sum_{t=1}^{\tau} (\hat{p}_{it}-0.5)^2}$. Owing to the heterogeneity in the temporal cross-section of the data and accounting for the subject-to-subject variability in the data, the final predicted response for subject i is $\hat{y}_i = 1$ if $\sum_{t=1}^{\tau} w_{it}\hat{p}_{it} > 0.5$; and $\hat{y}_i = 0$ otherwise. In addition to decreasing the leverage of the local models at which the signal is insignificant, our approach accounts for the fact that there are time points where the selected locations show minimal electrical activity.

4. Numerical studies

In this section, we investigate the performance of our GD approach in estimation, feature extraction, and subject-level prediction. Results of our GD method are compared to existing shrinkage and selection operators employed in PL regression: LASSO, adaptive LASSO (a-LASSO), elastic net (E-Net), mini-max concave penalty (MCP), smoothly clipped absolute deviation (SCAD). We include details surrounding the run time of the GD model in Section 5.

For all analyses reported in this paper, the model was fitted using the R package `rstan` (Stan Development Team, 2023); the Gibbs sampler derivation is provided in Section 3.1 for methodological context. The `rstan` implementation employs Hamiltonian Monte Carlo (HMC) (Betancourt and Girolami, 2015; Neal, 2011). The MH-within-Gibbs sampler presented in Section 3.1 and the Stan HMC implementation target the same GD posterior distribution; they differ only in the mechanism used to generate proposals. HMC uses gradient information from the log-posterior to construct efficient proposals that traverse the posterior geometry in fewer steps than a random-walk MH scheme. The smoothness and differentiability of the GD posterior, a property that follows directly from the continuous and differentiable nature of the GD penalty (see Section A.2.2), makes it particularly amenable to gradient-based sampling. The `rstan` implementation was adopted for all reported results on account of its superior computational efficiency, particularly for larger simulation settings and the EEG case study.

Prior elicitation. A generalized least squares estimator of β is used as the initial value in the simulations and case study. We use $\tau_0 = 10^{-5}$, because it is sufficiently small and works well in the numerical studies. In consideration of the low signal-to-noise ratio in the data, the constant used in the FDR approach to calculate the posterior inclusion probabilities is chosen to be $c = 10^{-3}$. A sensitivity analysis across a small range of c values is reported in Supplementary Section A.3. The FDR approach will not recognize any of the coefficients as significant for too large of a value of c , which results in the first stage of feature extraction producing an overly sparse coefficient matrix. We choose the shape and rate hyperparameters of the gamma distribution on λ to be $\alpha_1 = 0.1$ and $\alpha_2 = 0.2$, respectively. In other words, the prior mean is 0.5 and the prior variance is 2.5. We have prior belief that the model tends to perform better when λ is between 0 and 1, which is why

we chose the prior mean to be 0.5. However, choosing a fairly large prior variance allows for more flexibility in the way we present our prior knowledge and allows for a wider exploration of the parameter space.

4.1. Simulation study

In order to demonstrate the effectiveness and validity of our GD approach, we fit our model to multiple simulated data sets with different settings. The simulated data sets are sparse, high dimensional and aim to resemble the data structure used in the case study. In longitudinal simulation studies, only a single time point is considered, so one can simply generate the data, $\mathbf{X}_t$, which can be used along with the true coefficient vector, $\boldsymbol{\beta}_t$, and known inverse link function, $h(\cdot)$, to generate the response vector $\mathbf{y}$. A major challenge in simulating data for our framework is that $\mathbf{X}_t$ and $\boldsymbol{\beta}_t$ vary over time, while the response vector remains the same, i.e., $\mathbf{y}_t = \mathbf{y}$, for all $t = 1, \dots, \tau$. See Section A.3 of the Annexure for details on how we simulate our data and coefficients.

Since we want to assess the selection of active locations, we report the proportion of correctly and incorrectly selected locations using true positive rate (TPR) and false positive rate (FPR). We report the root mean square error (rMSE) of estimation after the first stage to illustrate the models ability to accurately estimate coefficients, although, sparsity in the final estimator causes estimation error rates to inflate. With $p = L \times \tau$, we define $rMSE = \sqrt{\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|^2/p}$. We present the receiver operating characteristics (ROC) curves and the area under the ROC curve (AUC) values as well. Additionally, we report three measures of prediction error rates: TPR, FPR, and prediction error (PE).

Three different values of L (25, 60, 75) and three different values of τ (100, 175, 250) were considered, resulting in nine total simulation settings. In addition, we consider three different σ_{noise} values, 1, 1.5, and 2, where σ_{noise} is the standard deviation of the normal distribution used to generate random noise. Let $\text{SNR} = (\max |\beta_j|)/\sigma_{\text{noise}}$ denote the signal-to-noise ratio, and $\max |\beta_j|$ represents the largest absolute regression coefficient. A larger value of σ_{noise} decreases the SNR, whereas a smaller value increases it. Consequently, simulation results depend on the chosen SNR, which is controlled through σ_{noise} . In the analysis reported below, we consider $\text{SNR} = 1/1.5$. See Section A.3 of the Annexure for more details on the different simulation settings. Each setting was replicated 100 times. All of the rates in the tables below are averages, along with standard deviations, over the 100 replications. Table 1 contains results from estimation and feature extraction, and Table 2 contains results from subject-level prediction.

As the number of locations, L , increases, the accuracy of correctly selected indices decreases, while the proportion of incorrectly selected locations rises. This trend makes sense since the number of predetermined active locations increases with L . Conversely, as the number of time points, τ , increases, the proportion of correctly selected locations increases, and the proportion of incorrectly selected locations decreases. Since the length of the location vectors is τ , for a large τ , the frequency of 1's will be much greater than the frequency of 0's in predetermined active locations, making it easier for the second stage of feature extraction to differentiate between active and inactive locations. Estimation accuracy also decreases with larger L , which is not surprising as the number of coefficients in each local model is L . Taking into account that the MCMC sampling algorithm is conducted at

Table 1: Estimation and feature extraction - simulation results

L	τ	rMSE	corrInd	incorrInd
25	100	0.542 (0.009)	0.860 (0.165)	0.110 (0.122)
	175	0.541 (0.006)	0.946 (0.110)	0.057 (0.079)
	250	0.541 (0.005)	0.952 (0.090)	0.038 (0.052)
60	100	0.574 (0.037)	0.676 (0.165)	0.254 (0.101)
	175	0.575 (0.030)	0.778 (0.158)	0.271 (0.100)
	250	0.571 (0.024)	0.812 (0.125)	0.234 (0.100)
75	100	0.966 (0.142)	0.611 (0.138)	0.334 (0.083)
	175	0.971 (0.092)	0.691 (0.139)	0.322 (0.098)
	250	0.979 (0.072)	0.735 (0.131)	0.342 (0.104)

Table 2: Prediction rates and AUC - simulation results

L	τ	TPR	FPR	PE	AUC
25	100	0.908 (0.076)	0.105 (0.098)	0.096 (0.063)	0.963 (0.041)
	175	0.953 (0.052)	0.037 (0.065)	0.043 (0.038)	0.993 (0.011)
	250	0.979 (0.040)	0.025 (0.044)	0.022 (0.029)	0.998 (0.006)
60	100	0.879 (0.092)	0.123 (0.116)	0.123 (0.079)	0.951 (0.059)
	175	0.945 (0.055)	0.062 (0.080)	0.057 (0.049)	0.984 (0.032)
	250	0.968 (0.043)	0.036 (0.055)	0.033 (0.036)	0.993 (0.018)
75	100	0.884 (0.083)	0.113 (0.094)	0.114 (0.066)	0.955 (0.043)
	175	0.935 (0.058)	0.055 (0.071)	0.061 (0.048)	0.986 (0.023)
	250	0.979 (0.036)	0.026 (0.053)	0.023 (0.032)	0.996 (0.012)

each local model, increasing L makes it more difficult for the algorithm to converge. There is no clear pattern that arises in the rMSE over τ , which is understandable considering the dimension of the local coefficient vectors is L and irrelevant of τ .

The general trends observed in the results for prediction follow similar to those for estimation and feature extraction. As L increases, prediction accuracy tends to decrease, and as τ increases, prediction accuracy increases. If the coefficient estimates are less accurate for large L , then there will also be more variance in the local prediction probabilities. By conducting the weighted prediction process as described in Section 3.3, the prediction accuracy decreases when there is more variability in local prediction probabilities. In the same way, we anticipate any additional uncertainty in the prediction process to decrease when τ is larger. Overall, the model appears to perform the best in estimation and feature extraction for large values of τ and small values of L . For prediction, the model appears to perform the best for large values of τ , mostly irrelevant of L . In the real data, there are 57 locations and 256 time points. Given the findings in the simulation setting with $L = 60$ and $\tau = 250$, we expect the model to perform fairly well in the EEG case study. We understand that the high subject-to-subject variability and low signal-to-noise ratio in the real data will likely reduce overall performance, but we anticipate results comparable to previous methods. Tables A.3-A.6 in Section A.4 of the Annexure contain results for different values of SNR.

Table 3: Prediction error rates and AUC - EEG results

Method	TPR	FPR	PE	AUC
GD	0.7143	0.2222	0.2623	0.7856
LASSO	0.8571	0.3778	0.2295	0.7616
a-LASSO	0.7662	0.2222	0.2295	0.7968
E-Net	0.8052	0.4000	0.2705	0.7723
MCP	0.8182	0.4444	0.2787	0.7752
SCAD	0.8571	0.3778	0.2295	0.7651

4.2. EEG case study

In this section, we provide results obtained from applying our GD method to a multi-subject EEG case study. Our goal is to classify subjects according to their EEG measurements to predict genetic predisposition to alcoholism. Details surrounding run time of the model are in Section 5. Due to the low signal-to-noise ratio and several other complex characteristics of the data, high prediction accuracy is hard to achieve. In light of this, we report ROC curves and AUC values on top of the prediction error rates which were used in the simulation studies. Results from the competing methods (LASSO, a-LASSO, E-Net, MCP, SCAD) are obtained by vectorizing the model coefficients into a $L\tau$ -dimensional vector and matricizing the three-dimensional array of data into an $n \times L\tau$ -dimensional matrix. On the other hand, our GD method constructs local models which actually tackles the high dimensional problem and allows us to recognize spatial and temporal patterns observed in the data. Table 3 provides the results and Figure 3 contains ROC curves for each method. We do not include comparisons with the spike-and-slab approach in Mohammed *et al.* (2019), because a function from an R package that was used to fit the model is no longer supported. For context, Mohammed *et al.* (2019) reported a prediction accuracy of 70.03% (equivalently, a prediction error of 29.97%) with $\text{TPR} = 0.7374$ and $\text{FPR} = 0.3600$ on the same EEG dataset using fivefold cross-validation with the Wcut thresholding method. Our GD method achieves an improved prediction error of 26.23% (prediction accuracy of 73.77%), with $\text{TPR} = 0.7143$ and $\text{FPR} = 0.2222$ using LOOCV. The two methods are broadly comparable in overall prediction accuracy, with our GD method achieving a lower prediction error and a notably lower FPR, suggesting that the GD prior with FDR-controlled screening induces more controlled sparsity in the selected locations.

Our final subject-level predictions were based on the coefficient estimates obtained after the first-stage feature selection, $\hat{\beta}^*$. Although all of the competing methods have high TPR, the GD method has the lowest FPR (tied with a-LASSO). Also, the GD method has the second highest AUC behind a-LASSO. Since the a-LASSO model was fit using ridge weights which were found using cross-validation, it makes sense that it performed the best here in terms of prediction rates. Even with that, the results from the GD method are comparable to a-LASSO and outperforms the other competing methods. Over and above that, our GD method determines which locations of the brain are active, identifying patterns of functional connectivity associated with cognitive or behavioral states. The competing methods can only determine whether an individual (l, t) pair is significant, i.e., whether a coefficient is zero or nonzero. They are unable to determine active locations as our framework allows.

We implemented leave-one-out cross-validation (LOOCV). With $n = 122$ subjects in

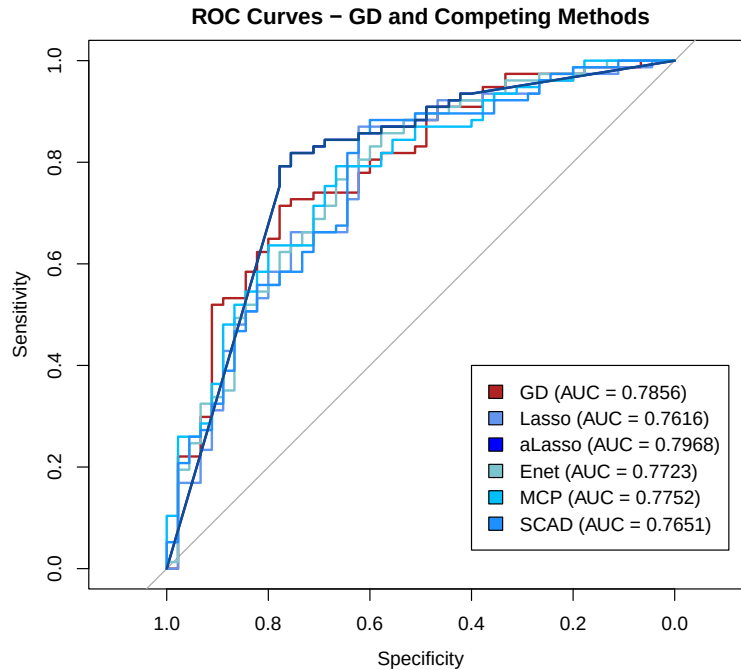


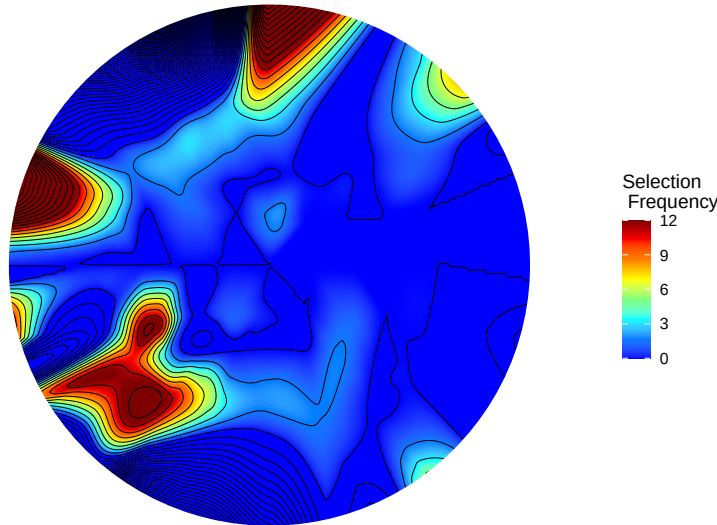
Figure 3: ROC curves - EEG results

the data, LOOCV can be thought of as a k -fold cross-validation where $k = 122$. Using this approach, an average of approximately 3.87 locations were selected over the 122 folds, which accounts for approximately 6.79% of the 57 locations. Similar to what is seen in Figure 2 of Hu and Allen (2015) and the plots in Section 2, the bulk of the activity is detected at the outer locations of the brain. We found that majority of the folds resulted in the model selecting the same 3 electrodes, CP5, P7, and P07, as active. It is worth noting these electrodes are located in the parietal lobe of the brain, which plays a crucial role in a wide range of sensory and cognitive processes. Note that, to empirically validate the choice of number of clusters in the second-stage feature extraction with k-means clustering, we considered cluster sizes $k = 2$ and $k = 3$ on the full EEG data set and fit the local models. We found that the average silhouette widths were 0.21 and 0.02 for $k = 2$ and $k = 3$, respectively. Although 0.21 ($k = 2$) reflects modest separation it is substantially higher than 0.02 ($k = 3$), supporting $k = 2$ as the empirically appropriate choice.

The parietal lobe is responsible for processes such as sensory integration and spatial awareness, and has been referred to as the association region of the brain Ackerman (1992). It has also been shown to be essential for visual tracking (Battelli *et al.*, 2009). Aside from the parietal lobe, other electrodes selected multiple times were located in the frontal lobe, shown to have fast visual responses, thought to reflect mostly low-level visual processing, and delayed responses that correlate with perceptual reports (Libedinsky and Livingstone, 2011). Table 4 provides the names of electrodes along with their associated frequency of selection. Only 18 out of 57 of the locations were selected at least once. To demonstrate where these electrodes are located relative to different regions of the brain, we included a scalp map in Figure 4 based on the frequency of selection for all locations over the 122 folds. We used a square root transformation on the data because of the large range of frequencies from 0 to 122. The darker red seen on the bottom left demonstrates the activity in the parietal lobe, and the darker red on the upper left and top of the plot demonstrate the activity observed

Table 4: Electrodes frequency of selection over the 122 folds in LOOCV.

Electrode	FPZ	AFz	F7	F5	F3	F8	FT7	FC3	FCZ
Frequency	39	2	7	6	2	1	20	1	3
Electrode	CP5	CP1	CP4	P7	P5	P4	PO7	POz	O2
Frequency	121	1	1	122	15	3	122	4	2

**Figure 4: Scalp map showing the frequency (square root) of selection for the 57 locations**

in different cortical areas of the frontal lobe. See Figure A.2 in Section A.4 of the Annexure for the exact spatial locations of the electrodes.

5. Conclusion

We introduce a novel Bayesian approach, utilizing a continuous-and-differentiable approximation of the ℓ_0 -norm penalty, to address the challenge of simultaneous feature extraction and estimation in handling sparse, high dimensional spatio-temporal data. To reduce the dimension of the problem, we employ a local modeling approach, offering the advantage of parallelizing local models. The Gaussian and gamma mixture representation allows for estimation via traditional Bayesian methods such as the MCMC sampling algorithm to extract information regarding the target posterior distribution. Although our method is being applied to EEG measurements to examine the effects of chronic alcohol exposure on specific brain regions, its adaptability extends to other domains with a similar spatio-temporal structure and dependencies across multiple dimensions.

Over and above the frequentist PL methods, Bayesian methods are able to quantify uncertainty in the PL estimator. We assign a gamma prior to the regularization parameter as a computationally efficient alternative to frequentist grid search, on top of other potential modeling benefits to explore. BD measures as loss functions allows our method to handle various likelihood functions. Our two-stage feature extraction process induces sparsity into the estimator and determines active locations. FDR thresholding induces sparsity in the first stage, while k-means clustering on the areas under the curve formed by estimated coefficient vectors selects active locations, avoiding a fixed threshold. While our GD approach doesn't

explicitly incorporate spatial-temporal structure, it demonstrates computational efficiency.

Spatio-temporal characteristics of the data and compute time. Bypassing the spatio-temporal structure of the data in our local modeling approach is a strategic choice, allowing us to avoid the need of extensive computational power. We still acknowledge the spatio-temporal behavior of the data. In the second stage, we employ k-means clustering to group locations exhibiting similar behavior over time. We account for the varying signal-to-noise ratio at different time points, giving less weight to those with an extremely low signal-to-noise ratio during our weighted prediction process. Despite alternative approaches explicitly incorporating spatial-temporal structure, our method yields computational efficiency and comparable results. For example, fitting a single local model on the EEG data with 1000 MCMC iterations, $n = 122$ subjects, and $L = 57$ locations on a computer with 32 GB of memory, 10 cores, and a Apple M1 Max chip, yields a run time of approximately 35.23 seconds, on average, with a standard deviation and range of approximately 11.54 seconds and 57.44 seconds, respectively. Fitting all $\tau = 256$ local models individually yields a run time of approximately 150.33 minutes. When we run 10 local models at a time, in parallel, on the 10 cores, the run time to fit all $\tau = 256$ local models is reduced to approximately 36.19 minutes. To further characterize scaling with n and L , we timed a single local model (1,000 MCMC iterations, 300 warmup, one chain) for $(n, L) \in \{(100, 25), (100, 60), (200, 25)\}$ on an Intel Xeon E5-2660v3 node with 256 GB RAM, yielding mean sampling times of 13.2, 26.5, and 18.0 seconds, respectively. Run time approximately doubles when L increases from 25 to 60 with n fixed, while increasing n from 100 to 200 with L fixed produces a more modest increase, indicating that run time is more sensitive to the number of locations than the number of subjects.

Although we avoided fixing a threshold, k-means clustering with two clusters comes with its own disadvantages. The weighted prediction process is based on a fixed threshold, leading to an arbitrary separation between subjects classified as alcoholic versus otherwise. Extending the framework to another real-data application (e.g., fMRI data or spatio-temporal climate measurements) which share the same three-dimensional array structure required by the method, remains an important and natural direction for future work. A major focus moving forward will be to improve the local modeling approach by incorporating the temporal and spatial structures explicitly through the prior. Beyond that, we also plan to extend the GD prior to three-dimensional array models, exploring new techniques to improve our feature extraction process, enhancing the prediction process, and assessing the model under flexible link functions.

Acknowledgments

SM was partially supported by Institutional Research Funds from Boston University and the Framingham Heart Study Brain Aging Program supported by U19-AG068753 from the National Institute of Health.

Code availability statement.

The code to implement the model using Stan is provided as an R package at <https://github.com/garrett-frady/gdprior>.

Conflict of interest

The authors do not have any financial or non-financial conflict of interest to declare for the research work included in this article.

References

- Ackerman, S. (1992). *Discovering the Brain*. National Academies Press.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**, 716–723.
- Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with Bregman divergences. *Journal of Machine Learning Research*, **6**, 1705–1749.
- Battelli, L., Alvarez, G. A., Carlson, T., and Pascual-Leone, A. (2009). The role of the parietal lobe in visual extinction studied with transcranial magnetic stimulation. *Journal of Cognitive Neuroscience*, **21**, 1946–1955.
- Beckmann, C. F. and Smith, S. M. (2005). Tensorial extensions of independent component analysis for multisubject fMRI analysis. *NeuroImage*, **25**, 294–311.
- Bernardo, J. M. and Smith, A. F. M. (2009). *Bayesian Theory*. Wiley Series in Probability and Statistics. Wiley.
- Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B (Methodological)*, **48**, 259–279.
- Betancourt, M. J. and Girolami, M. (2015). Hamiltonian Monte Carlo for hierarchical models. In Upadhyay, S. K., Singh, U., Dey, D. K., and Loganathan, A., editors, *Current Trends in Bayesian Methodology with Applications*, chapter 4, pages 79–102. Chapman and Hall/CRC Press, Boca Raton, FL.
- Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, **7**, 200–217.
- Caffo, B. S., Crainiceanu, C. M., Verduzco, G., Joel, S., Mostofsky, S. H., Bassett, S. S., and Pekar, J. J. (2010). Two-stage decompositions for the analysis of functional connectivity for fMRI with application to Alzheimer’s disease risk. *NeuroImage*, **51**, 1140–1149.
- Calhoun, V. D., Adali, T., Pearlson, G. D., and Pekar, J. J. (2001). A method for making group inferences from functional MRI data using independent component analysis. *Human Brain Mapping*, **14**, 140–151.
- Casella, G., Ghosh, M., Gill, J., and Kyung, M. (2010). Penalized regression, standard errors, and Bayesian Lassos. *Bayesian Analysis*, **5**, 369–411.
- Delorme, A. and Makeig, S. (2004). EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, **134**, 9–21.
- Dicker, L., Huang, B., and Lin, X. (2013). Variable selection and estimation with the seamless- L_0 penalty. *Statistica Sinica*, **23**, 929–962.
- Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, **96**, 1348–1360.

- Goh, G. and Dey, D. K. (2018). Bayesian MAP estimation using Gaussian and diffused-gamma prior. *Canadian Journal of Statistics*, **46**, 399–415.
- Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. (1999). Bayesian model averaging: a tutorial. *Statistical Science*, **14**, 382–417.
- Hu, Y. and Allen, G. I. (2015). Local-aggregate modeling for big-data via distributed optimization: applications to neuroimaging. *Biometrics*, **71**, 905–917.
- Huang, J., Shen, H., and Buja, A. (2009). The analysis of two-way functional data using two-way regularized singular value decompositions. *Journal of the American Statistical Association*, **104**, 1609–1620.
- Kim, Y., Choi, H., and Oh, H.-S. (2008). Smoothly clipped absolute deviation on high dimensions. *Journal of the American Statistical Association*, **103**, 1665–1673.
- Lee, M., Shen, H., Huang, J. Z., and Marron, J. S. (2010). Biclustering via sparse singular value decomposition. *Biometrics*, **66**, 1087–1095.
- Libedinsky, C. and Livingstone, M. (2011). Role of prefrontal cortex in conscious visual perception. *Journal of Neuroscience*, **31**, 64–69.
- Lukic, A. S., Wernick, M. N., Hansen, L. K., Anderson, J., and Strother, S. C. (2002). A spatially robust ICA algorithm for multiple fMRI data sets. In *Proceedings IEEE International Symposium on Biomedical Imaging*, pages 839–842.
- Mitchell, T. J. and Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, **83**, 1023–1032.
- Mohammed, S., Bharath, K., Kurtek, S., Rao, A., and Baladandayuthapani, V. (2021). RADIOHEAD: radiogenomic analysis incorporating tumor heterogeneity in imaging through densities. *The Annals of Applied Statistics*, **15**, 1808–1830.
- Mohammed, S., Dey, D. K., and Zhang, Y. (2019). Bayesian variable selection using spike-and-slab priors with application to high dimensional electroencephalography data by local modelling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **68**, 1305–1326.
- Morris, J. S., Brown, P. J., Herrick, R. C., Baggerly, K. A., and Coombes, K. R. (2008). Bayesian analysis of mass spectrometry proteomic data using wavelet-based functional mixed models. *Biometrics*, **64**, 479–489.
- Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, chapter 5, pages 113–162. Chapman & Hall/CRC, Boca Raton, FL.
- Ou, W., Hämmäläinen, M. S., and Golland, P. (2009). A distributed spatio-temporal EEG/MEG inverse solver. *NeuroImage*, **44**, 932–946.
- Ročková, V. and George, E. I. (2014). EMVS: the EM approach to Bayesian variable selection. *Journal of the American Statistical Association*, **109**(506), 828–846.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**, 461–464.
- Snodgrass, J. G. and Vanderwart, M. (1980). A standardized set of 260 pictures: norms for name agreement, image agreement, familiarity, and visual complexity. *Journal of Experimental Psychology: Human Learning and Memory*, **6**, 174–215.
- Stan Development Team (2023). RStan: the R interface to Stan. R package version 2.26.23.

- Svensén, M., Kruggel, F., and Benali, H. (2002). ICA of fMRI group study data. *NeuroImage*, **16**, 551–563.
- Tam, W.-k., Wu, T., Zhao, Q., Keefer, E., and Yang, Z. (2019). Human motor decoding from neural signals: a review. *BMC Biomedical Engineering*, **1**, 12.
- Tian, T. S., Huang, J. Z., Shen, H., and Li, Z. (2012). A two-way regularization method for MEG source reconstruction. *Annals of Applied Statistics*, **6**, 1021–1046.
- Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288.
- Wedderburn, R. W. M. (1974). Quasi-likelihood functions, generalized linear models, and the Gauss–Newton method. *Biometrika*, **61**, 439–447.
- Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, **10**, 515–534.
- Zhang, C., Jiang, Y., and Shang, Z. (2009). New aspects of Bregman divergence in regression and classification with parametric and nonparametric estimation. *Canadian Journal of Statistics*, **37**, 119–139.
- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, **38**, 894–942.
- Zhou, H. and Li, L. (2014). Regularized matrix regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **76**, 463–483.
- Zhou, H., Li, L., and Zhu, H. (2013). Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, **108**, 540–552.
- Zou, H. (2006). The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, **101**, 1418–1429.
- Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**, 301–320.

ANNEXURE

A.1. Literature review

In this section of the supplemental material, we highlight previously proposed methods found in literature to handle feature extraction in high dimensional data with complex characteristics. Specifically, we outline techniques introduced to reduce the dimension of the problem and provide a detailed review on shrinkage and selection penalty functions used to induce sparsity into the penalized loss function (PL) estimator.

Several procedures use a combination of principal component analysis (PCA) and independent component analysis (ICA) or an extension of them to reduce the dimension of the covariate structure. Calhoun *et al.* (2001) proposed a factor-analytic group decomposition technique applied to functional MRI (fMRI) data, which decomposes the spatial structure using independent components analysis (ICA). Lukic *et al.* (2002) and Svensén *et al.* (2002) developed methods using ICA and eigen-decomposition on the covariate structure to map spatial and temporal patterns of active brain regions. Beckmann and Smith (2005) proposed spatio-temporal decomposition through a tensorial extension of ICA applied to multi-subject fMRI data. Caffo *et al.* (2010) proposed a two-stage method to reduce the dimension of the covariates using subject-level singular-value decomposition (SVD), followed by population-level principal component analysis (PCA). Since there is no clear and concise interpretation of the reduced dimensions obtained from PCA and ICA, the goal of conducting variable selection on associated brain regions is not achieved.

To highlight the spatio-temporal correlations ever-present in neuroimaging measurements, several methods were proposed involving a combination of roughness and sparse penalties imposed on the spatial and temporal domains. Ou *et al.* (2009) used a one-way $\ell_1\ell_2$ -regularization method and Huang *et al.* (2009) used two-way regularized SVD methods imposing roughness penalties on both the spatial and temporal domains Witten *et al.* (2009) and Lee *et al.* (2010) imposed sparse penalties on both domains, Tian *et al.* (2012) placed a rough penalty and a sparse penalty separately on the two domains. Other techniques impose regularization penalties on a family of tensor GLMs, imposing a low rank approximation on the predictor tensor for dimension reduction (Zhou *et al.*, 2013; Zhou and Li, 2014). However, such methods are computationally unattractive and lack effective statistical inference.

Sparsity inducing penalty functions. Although the ℓ_0 -norm penalty induces sparsity, quantifying $\hat{\beta}_{PL}$ remains computationally cumbersome caused by repercussions from the jumpy (discontinuous) nature of indicator functions. As a way of removing the discontinuities established by the indicator function in the ℓ_0 -norm penalty, Tibshirani (1996) makes use of the ℓ_1 -norm penalty to propose a new convex optimization technique - *LASSO* (least absolute shrinkage and selection operator), $\text{LASSO}(\beta|\lambda) = \lambda \sum_{j=1}^p |\beta_j|$, where $\lambda \geq 0$ is the tuning parameter. However, there are limitations surrounding variable selection when making use of the LASSO penalty. Zou and Hastie (2005) pointed out that the convex nature of the LASSO problem, when $p \gg n$, allows for no more than n variables to be selected, and further, struggles selecting more than one variable from a group with high pairwise correlations. In response to the preceding restrictions faced with LASSO, Zou and Hastie (2005) developed an elastic net penalty to induce sparsity, $\text{e-net}(\beta|\lambda_1, \lambda_2) = \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2$, where $\lambda_1, \lambda_2 \geq 0$ are tuning parameters for the LASSO component

and ridge component, respectively. The elastic net penalty is a mixture of the LASSO penalty and the ridge penalty introduced to inherit the positive characteristics of LASSO while addressing its limitations. Zou and Hastie (2005) mentions that the naiveness of the elastic net penalty leads to biased estimation and classification when it does not closely resemble the LASSO or ridge penalty. It was additionally shown by Zou (2006) that employing a common regularization parameter λ leads to inconsistent estimates of $\hat{\beta}_{PL}$. To overcome this obstacle, Zou (2006) developed an alternative to LASSO by introducing a weighted ℓ_1 -norm penalty, called adaptive LASSO (a-LASSO), $a\text{-LASSO}(\beta|\lambda, \mathbf{w}) = \lambda \sum_{j=1}^p w_j |\beta_j|$, where $\mathbf{w} = (w_1, \dots, w_p)^T$ is the weight vector and $\lambda \geq 0$ is the regularization parameter. With an appropriate choice of weights based on $\sqrt{n}$ -consistent estimators for components of $\hat{\beta}_{PL}$, it can be shown that the a-LASSO penalty removes the bias issue by dropping insignificant coefficients (Zou, 2006). Selecting suitable weights to achieve the formerly stated advantage of a-LASSO is computationally infeasible as we are not aware of the insignificant predictors beforehand. Furthermore, using the data to construct weights is unrealistic in the presence of high dimensions. Kim *et al.* (2008) remedy the constraints attached to the a-LASSO method with an iterative concave convex procedure coupled with the smoothly clipped absolute deviation (SCAD) penalty suggested by Fan and Li (2001)

$$\begin{aligned} \text{SCAD}(\beta|\lambda, \gamma) = & \lambda \sum_{j=1}^p |\beta_j| \mathbb{1}_{(|\beta_j| < \lambda)} + \sum_{j=1}^p \left(\frac{\gamma\lambda(|\beta_j| - \lambda) - (|\beta_j|^2 - \lambda^2)^2/2}{\gamma - 1} + \lambda^2 \right) \mathbb{1}_{(\lambda \leq |\beta_j| \leq \gamma\lambda)} \\ & + \sum_{j=1}^p \left(\frac{(\gamma - 1)\lambda^2}{2} + \lambda^2 \right) \mathbb{1}_{(|\beta_j| \geq \gamma\lambda)} \end{aligned}$$

where $\lambda \geq 0$ and $\gamma > 1$ is a shrinkage parameter. The drawback of the SCAD penalty used for this purpose is the level of computational and analytical difficulty in finding $\hat{\beta}_{PL}$ coming from the nonconvexity of the optimization problem (Zhang, 2010). As an alternative, Zhang (2010) developed a nearly unbiased estimation and selection method by introducing the minimax concave penalty (MCP)

$$\text{MCP}(\beta|\lambda, \gamma) = \lambda \sum_{j=1}^p \left[\int_0^{|\beta_j|} (1 - x/(\gamma\lambda))_+ dx \right], \quad (5)$$

where $\lambda \geq 0$, $\gamma > 0$, and subscript $+$ denotes the positive portion of a function.

A.2. Details on GD prior and GD posterior

In this section, we elaborate on the details of the GD prior and posterior distribution. As mentioned in the main paper, the GD prior is constructed in a way that the GD estimator acts as a Bayesian version of the PL estimator. In particular, the GD estimator can be shown to be proportional to the PL estimator with the class of BD measures as the choice of loss function and an ℓ_0 -norm approximation as the penalty function.

A.2.1. Bregman divergence measures as loss functions

Here, we further discuss the wide-ranging application and flexibility of the class of BD measures, the duality property, and the bijective relationship between the class of BD

Table A.1: Bregman divergence measures as known loss functions

$\psi(\mathbf{z})$	$BD_\psi(\mathbf{z}_1, \mathbf{z}_2)$	Loss Function
$\ \mathbf{z}\ _2^2$	$\ \mathbf{z}_1 - \mathbf{z}_2\ _2^2$	Squared Euclidean loss
$\mathbf{z}^T \mathbf{W} \mathbf{z}$	$(\mathbf{z}_1 - \mathbf{z}_2)^T \mathbf{W} (\mathbf{z}_1 - \mathbf{z}_2)$	Mahalanobis distance
$\sum_{i=1}^n z_i \log z_i$	$\sum_{i=1}^n z_{1i} \log \left(\frac{z_{1i}}{z_{2i}} \right) - (z_{1i} - z_{2i})$	Kullback-Leibler divergence
$\sum_{i=1}^n -\log z_i$	$\sum_{i=1}^n \left\{ \frac{z_{1i}}{z_{2i}} - \log \left(\frac{z_{1i}}{z_{2i}} \right) - 1 \right\}$	Itakura-Saito distance
$\sum_{i=1}^n e^{cz_i}$	$\sum_{i=1}^n \left\{ e^{cz_{2i}} \left(e^{c(z_{1i} - z_{2i})} - c(z_{1i} - z_{2i}) - 1 \right) \right\}$	Weighted Linex loss

measures and natural exponential family. Let $\psi : \Omega \rightarrow \mathbb{R}$ denote a strictly convex and differentiable function on a nonempty convex set $\Omega \subseteq \mathbb{R}^n$, and let $\mathbf{z}_1, \mathbf{z}_2 \in \Omega$. The Bregman divergence for $\mathbf{z}_1$ and $\mathbf{z}_2$ with respect to ψ is defined to be $BD_\psi(\mathbf{z}_1, \mathbf{z}_2) = \psi(\mathbf{z}_1) - \psi(\mathbf{z}_2) - (\mathbf{z}_1 - \mathbf{z}_2)^T \nabla \psi(\mathbf{z}_2)$, where $\nabla \psi$ represents the gradient vector of ψ (Bregman, 1967).

Duality property. Originally discussed in Bernardo and Smith (2009), the duality property affirms that the negative log-likelihood function can be viewed as a loss function. Extending the dual relationship between the likelihood function and loss function to the data-generating distribution, we assume that the negative log-likelihood function is a BD measure. Consider the standard normal likelihood function $f(\mathbf{y}|\boldsymbol{\beta}) \propto \exp\{-\frac{1}{2}L_2(\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta}))\}$, where $L_2(\mathbf{z}_1, \mathbf{z}_2) = \|\mathbf{z}_1 - \mathbf{z}_2\|_2^2$ denotes the squared Euclidean loss. Hence, $-\log f(\mathbf{y}|\boldsymbol{\beta}) \propto L_2(\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta}))$, i.e., the negative log-likelihood function for the standard normal distribution is proportional to the squared Euclidean loss function. Table A.1 below shows some examples of Bregman divergence measures as loss functions for specified strictly convex functions.

Bijection with natural exponential family. An advantage of using the class of BD measures as our choice of loss function is the flexibility they bring into the approach as a result of the bijective relationship existing with the natural exponential family. For instance, if we consider the strictly convex function defined by $\psi(z) = \sum_{i=1}^n z_i \log z_i$, then the BD measure can be written as proportional to the Poisson likelihood function using as follows:

$$\begin{aligned}
 f(\mathbf{y}|\boldsymbol{\beta}) &\propto \exp \left\{ - \sum_{i=1}^n \left[y_i \log \left(\frac{y_i}{h(\mathbf{x}_i^T \boldsymbol{\beta})} \right) - (y_i - h(\mathbf{x}_i^T \boldsymbol{\beta})) \right] \right\} \\
 &\propto \prod_{i=1}^n \left(e^{-h(\mathbf{x}_i^T \boldsymbol{\beta})} [h(\mathbf{x}_i^T \boldsymbol{\beta})]^{y_i} \right).
 \end{aligned}$$

Table A.2 provides examples of the bijective relationship between BD measures and the natural exponential family. Using the BD measure corresponding to the Bernoulli distribution, we have

$$\begin{aligned}
 BD_\psi(\mathbf{y}, h(\mathbf{X}\boldsymbol{\beta})) &= \sum_{i=1}^N \left\{ y_i \log \left(\frac{y_i}{h(\mathbf{x}_i^T \boldsymbol{\beta})} \right) + (1 - y_i) \log \left(\frac{1 - y_i}{1 - h(\mathbf{x}_i^T \boldsymbol{\beta})} \right) \right\} \\
 &= \sum_{i=1}^N \left\{ \log \left(\left[\frac{y_i}{h(\mathbf{x}_i^T \boldsymbol{\beta})} \right]^{y_i} \left[\frac{1 - y_i}{1 - h(\mathbf{x}_i^T \boldsymbol{\beta})} \right]^{1 - y_i} \right) \right\},
 \end{aligned}$$

and the likelihood function can be written as

$$f_\psi(\mathbf{y}|\boldsymbol{\beta}) \propto \exp\{-BD_\psi(\mathbf{y}, h(\mathbf{X}\boldsymbol{\beta}))\} \propto \prod_{i=1}^N \left([h(\mathbf{x}_i^T \boldsymbol{\beta})]^{y_i} [1 - h(\mathbf{x}_i^T \boldsymbol{\beta})]^{1 - y_i} \right),$$

Table A.2: Bregman divergence measures and their corresponding distributions in the natural exponential family along with the associated strictly convex function

$\psi(z)$	$BD_\psi(z_1, z_2)$	Distribution
$\frac{1}{2\sigma^2}z^2$	$\frac{1}{2\sigma^2}(z_1 - z_2)^2$	Gaussian
$z \log z$	$z_1 \log\left(\frac{z_1}{z_2}\right) - (z_1 - z_2)$	Poisson
$-\log z$	$\frac{z_1}{z_2} - \log\left(\frac{z_1}{z_2}\right) - 1$	Exponential
$z \log z + (1 - z) \log(1 - z)$	$z_1 \log\left(\frac{z_1}{z_2}\right) + (1 - z_1) \log\left(\frac{1 - z_1}{1 - z_2}\right)$	Bernoulli

which is the Bernoulli likelihood function. It is worth mentioning that a more general distribution family, relative to the natural exponential family, can be generated by Bregman divergence measures. For instance, Zhang *et al.* (2009) demonstrated that the quasi-likelihood function (Wedderburn, 1974) can be generated by BD measures.

A.2.2. GD penalty

Here, we emphasize the advantage of the GD penalty over the SELO penalty. In particular, we demonstrate numerically that the GD penalty is differentiable everywhere and that the SELO penalty is not differentiable at $\beta_j = 0$. Dicker *et al.* (2013) defined the SELO penalty as $\text{SELO}(\beta|\lambda, \tau_0) = \frac{\lambda}{\log(2)} \sum_{j=1}^p \log\left(\frac{|\beta_j|}{|\beta_j| + \tau_0} + 1\right)$, where $\lambda \geq 0$ is a tuning parameter and $\tau_0 > 0$ is a deterministic constant. Note, as $\tau_0 \rightarrow 0$, $\log\left(\frac{|\beta_j|}{|\beta_j| + \tau_0} + 1\right) \rightarrow \log(2)$ for $|\beta_j| \neq 0$. Hence, the SELO penalty converges to the ℓ_0 -norm penalty as τ_0 tends to 0.

Advantage of GD penalty over SELO penalty. Although the SELO penalty is also a continuous approximation of the ℓ_0 -norm penalty, it has restrictions. At $\beta_j = 0$, the SELO penalty is not differentiable. Whereas, the GD penalty is differentiable everywhere. Due to the smoothness of the GD penalty function, applying the Newton-Raphson algorithm is stable and convergence is fast. We provide a numerical demonstration of this advantage the GD penalty has over the SELO penalty. Observe,

$$\frac{\delta \text{SELO}(\beta|\lambda, \tau_0)}{\delta \beta_j} = \frac{\lambda}{\log(2)} \times \frac{\tau_0 \beta_j}{|\beta_j|(2|\beta_j| + \tau_0)(|\beta_j| + \tau_0)^2},$$

which does not exist at $\beta_j = 0$. On the other hand,

$$\frac{\delta \tilde{\ell}_0(\beta|\lambda, \tau)}{\delta \beta_j} = \lambda \times \frac{2\{X_{[j]}^T X_{[j]}\} \beta_j^2 \tau^2}{(\tau^2 + \{X_{[j]}^T X_{[j]}\} \beta_j^2)^2},$$

which is 0 at $\beta_j = 0$. This shows that the GD penalty is continuous and differentiable everywhere, and the SELO penalty is continuous, but not differentiable everywhere.

A.2.3. GD posterior distribution

It is known that if the prior distribution is proper and the likelihood function is non-degenerate, then the set of points leading to an improper posterior distribution has Lebesgue measure zero, i.e., the posterior distribution is proper almost everywhere. Further,

$$\begin{array}{ccc}
\boxed{\text{Time 1}} & \boxed{\text{Time 2}} & \boxed{\text{Time } \tau} \\
\hline
\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \sim (\mathbf{X}_1)(\boldsymbol{\beta}_1) & \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \sim (\mathbf{X}_2)(\boldsymbol{\beta}_2) \cdots & \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \sim (\mathbf{X}_\tau)(\boldsymbol{\beta}_\tau)
\end{array}$$

Figure A.1: Illustration of local Bayesian models

if the likelihood function is uniformly bounded, then the posterior distribution is proper everywhere. Notice, for all $\boldsymbol{\beta} \in \mathbb{R}^L$, the GD prior, $\pi_G(\boldsymbol{\beta}|\hat{\mathbf{d}}_{MAP})$, is proper, and the likelihood function, $f_\psi(\mathbf{y}|\boldsymbol{\beta})$, is non-degenerate and bounded. One can see that the GD prior is proper from Lemma 1 and by observing the form of $\tilde{\ell}_0(\cdot|\lambda, \tau)$. Also, to verify that the likelihood function is non-degenerate and bounded, one can recall the form of the likelihood, which uses the duality property, and note that BD measures are non-negative. Thus, the GD posterior distribution is proper. However, posterior computation in high dimensional problems requires more careful attention.

We have several mechanisms to summarize the information obtained from the MCMC samples. We may compute the posterior mode (MAP estimate), posterior mean, or posterior median, and conduct inference based on point estimates. The posterior distribution of $\boldsymbol{\beta}$ can be used to assess the uncertainty associated with our estimation procedure. Thus, we are able to calculate credible intervals and highest posterior density (HPD) intervals for the model parameters. The posterior distribution can be used to generate posterior probabilities to determine the “best” model or set of models, and generate further probabilities of interest in variable selection (Mitchell and Beauchamp, 1988). We may also use Bayesian model averaging (Hoeting *et al.*, 1999) to incorporate multiple scenarios explored by the MCMC sampler.

A.3. Simulation settings and details

In longitudinal simulation studies, only a single time point is considered, so one can simply generate the data, $\mathbf{X}_t$, which can be used along with the true coefficient vector, $\boldsymbol{\beta}_t$, and known link function, $h(\cdot)$, to generate the response vector $\mathbf{y}$. A major challenge in simulating data for our framework is that $\mathbf{X}_t$ and $\boldsymbol{\beta}_t$ vary over time, while the response vector remains the same, i.e., $\mathbf{y}_t = \mathbf{y}$, for all $t = 1, \dots, \tau$. To overcome this hurdle, we generate a Bernoulli response vector $\mathbf{y} \in \mathbb{R}^n$ with a fixed probability, we set $P(y_i = 1) = 77/122$ to imitate the real data. The three-dimensional data array $\chi \in \mathbb{R}^{n \times L \times \tau}$ is constructed by first generating $\mathcal{U} \in \mathbb{R}^{N \times L \times \tau}$ from a multivariate normal distribution with mean $\mathbf{0}$, such that N is chosen to be sufficiently larger than n . Taking into account the high subject-to-subject variability in the data, the covariance matrix is constructed by $\Sigma_{ij} = \rho^{|i-j|}$, $i, j = 1, \dots, L$, where $0 < \rho < 1$ is chosen to incorporate correlations between locations. In our simulations, we choose $\rho = 0.5$. For $t = 1, \dots, \tau$, let $\mathbf{X}_t \in \mathbb{R}^{n \times L}$ denote the matrix along the third dimension of χ and $\mathbf{U}_t \in \mathbb{R}^{N \times L}$ denote the matrix along the third dimension of $\mathcal{U}$. We generate rows of $\mathbf{X}_t$ from the larger collection of rows of $\mathbf{U}_t$ so that $\mathbf{X}_t \boldsymbol{\beta}_t$ corresponds to $\mathbf{y}$, for all t . See Figure A.1 for a demonstration of our local modeling approach.

We generate the true coefficient matrix $\boldsymbol{\beta} = (\beta_{lt})_{l=1, \dots, L, t=1, \dots, \tau}$ by sampling entries

Table A.3: Simulation results: estimation and feature extraction; SNR = 1

L	τ	$rMSE_{est}$	corrInd	incorrInd
25	100	0.516 (0.010)	0.946 (0.094)	0.035 (0.047)
	175	0.514 (0.007)	0.980 (0.060)	0.008 (0.012)
	250	0.514 (0.006)	0.990 (0.044)	0.004 (0.017)
60	100	0.573 (0.045)	0.793 (0.131)	0.184 (0.094)
	175	0.565 (0.023)	0.860 (0.107)	0.140 (0.078)
	250	0.566 (0.025)	0.903 (0.093)	0.117 (0.085)
75	100	0.826 (0.101)	0.714 (0.125)	0.246 (0.103)
	175	0.825 (0.088)	0.789 (0.126)	0.242 (0.094)
	250	0.841 (0.070)	0.841 (0.108)	0.226 (0.097)

Table A.4: Simulation results: prediction; SNR = 1

L	τ	TPR	FPR	PE	AUC
25	100	0.990 (0.022)	0.016 (0.038)	0.012 (0.019)	0.999 (0.003)
	175	0.997 (0.013)	0.002 (0.015)	0.003 (0.010)	1.000 (0.000)
	250	1.000 (0.000)	0.001 (0.007)	0.000 (0.003)	1.000 (0.000)
60	100	0.970 (0.048)	0.034 (0.059)	0.031 (0.036)	0.995 (0.011)
	175	0.991 (0.026)	0.005 (0.022)	0.007 (0.017)	0.999 (0.003)
	250	0.998 (0.012)	0.000 (0.000)	0.001 (0.006)	1.000 (0.001)
75	100	0.967 (0.040)	0.033 (0.057)	0.032 (0.034)	0.994 (0.011)
	175	0.995 (0.018)	0.003 (0.018)	0.005 (0.013)	1.000 (0.002)
	250	1.000 (0.005)	0.003 (0.020)	0.001 (0.007)	1.000 (0.001)

from a Bernoulli distribution and randomly assigning 20% of the locations as active. To differentiate between active and inactive locations, entries in active rows are assigned lower probability of being 0 compared to inactive locations. Entries in inactive locations are 0 with probability 0.8 and active locations are 0 with probability 0.2. In other words, active locations should have more 1 entries, whereas inactive locations should have more 0 entries. We then randomly select 30% of the coefficients to be multiplied by -1 to account for negative electrical signals. After we generate the coefficient matrix, we fix the response vector and generate the data array as described above.

Due to the low signal-to-noise ratio observed in EEG data, caused by the non-invasiveness of the procedure, we add random noise to the data from a normal distribution centered at 0. By adding random noise to the data, we also account for the sporadic behavior seen in EEG measurements. Without doing this, the simulation settings are infeasible and the results will not accurately depict the performance of the model when applied to real data. We explored three different values of *noiseSD*, 1, 1.5, and 2. Table A.3 and Table A.4 contain simulation results from using a standard deviation of 1, i.e., SNR = 1. Table A.5 and Table A.6 contain simulation results from using a standard deviation of 2, i.e., SNR = 1/2.

For SNR = 1, we can see that the model performs extremely well. In particular, when τ is large, the model achieves near perfect prediction accuracy. Although we observe a jump in rMSE when L changes from 60 to 75 in the SNR = 1 case, we observe a distinctively larger jump in SNR = 1/2 case. It is reasonable to see this behavior due to the fact that

Table A.5: Simulation results: estimation and feature extraction; SNR = 1/2

L	τ	$rMSE_{est}$	corrInd	incorrInd
25	100	0.553 (0.009)	0.818 (0.166)	0.174 (0.124)
	175	0.552 (0.006)	0.876 (0.153)	0.144 (0.129)
	250	0.552 (0.005)	0.876 (0.147)	0.096 (0.112)
60	100	0.581 (0.040)	0.613 (0.168)	0.305 (0.089)
	175	0.582 (0.030)	0.637 (0.181)	0.285 (0.099)
	250	0.583 (0.028)	0.682 (0.166)	0.301 (0.094)
75	100	0.966 (0.142)	0.611 (0.138)	0.334 (0.083)
	175	0.971 (0.092)	0.691 (0.139)	0.322 (0.098)
	250	0.979 (0.072)	0.735 (0.131)	0.342 (0.104)

Table A.6: Simulation results: prediction; SNR = 1/2

L	τ	TPR	FPR	PE	AUC
25	100	0.787 (0.107)	0.215 (0.136)	0.211 (0.093)	0.872 (0.090)
	175	0.875 (0.078)	0.116 (0.099)	0.122 (0.063)	0.950 (0.043)
	250	0.902 (0.074)	0.105 (0.098)	0.100 (0.066)	0.966 (0.035)
60	100	0.793 (0.116)	0.220 (0.127)	0.213 (0.090)	0.865 (0.088)
	175	0.832 (0.114)	0.153 (0.112)	0.162 (0.089)	0.919 (0.072)
	250	0.895 (0.079)	0.120 (0.100)	0.110 (0.070)	0.957 (0.052)
75	100	0.884 (0.083)	0.113 (0.094)	0.114 (0.066)	0.955 (0.043)
	175	0.935 (0.058)	0.055 (0.071)	0.061 (0.048)	0.986 (0.023)
	250	0.979 (0.036)	0.026 (0.053)	0.023 (0.032)	0.996 (0.012)

lower SNR will have a greater impact when the number of locations is large.

The overall trends observed in estimation, feature extraction, and prediction are similar to those observed in Table 3 and 4 when SNR = 1/1.5. As expected, the overall performance and accuracy of the model dwindle as the SNR decreases. With a low SNR, it is challenging for the model to distinguish the meaningful signal from the noise, which leads to decreased accuracy in classifying subjects and increased difficulty in identifying active locations. Also, low SNR can lead to increased sensitivity to minor fluctuations or variations in the data.

Sensitivity of the stage 1 FDR threshold c . To assess whether Stage 1 variable selection is sensitive to the FDR constant c used to compute posterior inclusion probabilities, we repeated the representative simulation setting (SNR = 1/2) under three choices, $c \in \{10^{-4}, 10^{-3}, 10^{-2}\}$. The results show robustness across this range. In particular, the true-positive rate is identical for $c = 10^{-4}$ and $c = 10^{-3}$ (TPR = 0.64 in both cases), while the false-positive rate remains zero and AUC is 0.944 throughout all three choices of c . The only noticeable difference is a small reduction in TPR at the largest threshold considered (TPR = 0.60 at $c = 10^{-2}$), without any increase in false positives or loss in AUC. Overall, these findings indicate that the Stage 1 screening and downstream prediction are stable over two orders of magnitude around our chosen value. Selecting $c = 10^{-3}$ therefore preserves the best observed recovery among the tested values while maintaining a deliberately conservative stance (zero false positives), which is appropriate given that the true signal-to-noise ratio in

the EEG application is unknown and may be lower than in the simulation setting.

A.4. EEG analysis and visualization

In this section, we describe the graphical tools used to visualize the EEG data, expand on the programming approach used to fit the EEG data, and provide additional interpretations of the results.

ERP plots and scalp maps. Although more advanced approaches to signal processing of biophysical signals in the brain are available, the use of amplitude and latency measures of peaks in EEG trial averages, known as event related potentials (ERPs), are still heavily utilized in electrophysiology research (Delorme and Makeig, 2004). An ERP-image plot contains several time series curves (one for each location) which represent the average “potential” (amongst all subjects) at each latency. Delorme and Makeig (2004) additionally described a data visualization method referred to as scalp maps, which show the topographical behavior of the distribution of average potential at a specified time point.

Fitting EEG data. Prior to using LOOCV, we were averaging the results over 100 replications as we did in the simulations. That is, we randomly generated 100 different training and testing sets, fit the model on each of the training sets, analyzed the results using the corresponding testing sets, and averaged the results from each of the 100 replications. It was clear that certain seeds resulted in poor splits of the data. We could see that some splits returned high tpr’s and low fpr’s, while others had values of tpr’s and fpr’s close to 0.5. It followed that the AUC values below 0.5 corresponded to the same bad splits. To avoid “bad” splits, we decided to use LOOCV. The results from LOOCV were much nicer and proved that the bad splits were the issue before. Out of all 122 folds, an average of approximately 3.87 locations were selected, which accounts for approximately 6.79% of the 57 locations being considered in the data set. We tried using a union of all of the locations selected at least once, but this only amounted to 18/57 ($\approx 31.58\%$) of the locations. We found that majority of the folds resulted in the model selecting 3 of the 57 locations as active. To be specific, 74/122 ($\approx 60.66\%$) of the folds selected exactly 3 locations as active, and the indices of the selected locations were exactly the same: 35 (CP5), 43 (P7), 52 (P07). Interesting enough, all 122 of the folds selected electrodes P7 and P07, and 121 folds selected CP5; that is, CP5, P7, and P07 were all selected in 121/122 folds. Aside from those three, location 2 (FPZ) was selected in 39/122 ($\approx 31.97\%$), location 16 (FT7) was selected in 20/122 ($\approx 16.39\%$), and location 44 (P5) was selected in 15/122 ($\approx 12.30\%$). For reference to the scalp map pertaining to the frequency of selection of locations, we provide a topographic image of the scalp in Figure A.2 which specifies the precise locations of the electrodes.

A.5. Proofs of useful results

Lemma 1. *If $g(\tau_0|x) = \frac{x^2}{\tau_0^2 + x^2}$, then $\lim_{\tau_0 \rightarrow 0} g(\tau_0|x=0) = 0$ and $\lim_{\tau_0 \rightarrow 0} g(\tau_0|x \neq 0) = 1$.*

The proof for Lemma 1 is omitted as the result is elementary.

Lemma 2. *Let $\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})$ denote the GD prior. Then, for any $\lambda \geq 0$ and $\tau_0 > 0$,*

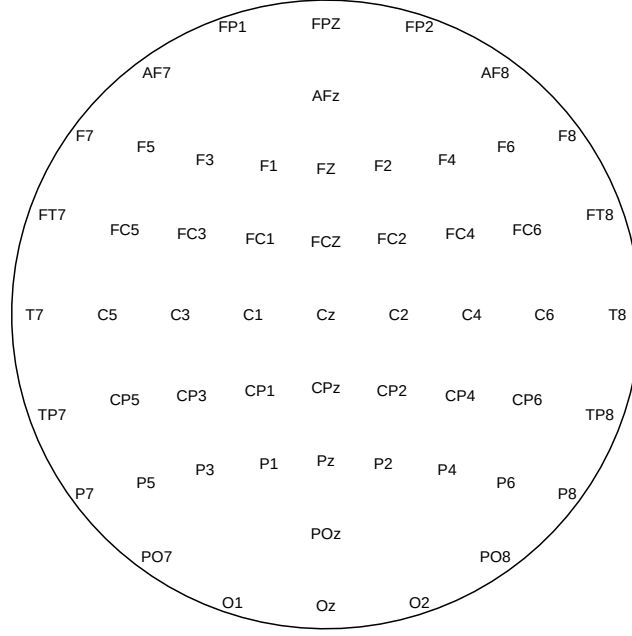


Figure A.2: Showing names of electrodes at their corresponding locations of the brain

$$\arg \max_{\mathbf{d}} \{\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})\} = \left(\frac{2\lambda}{\tau_1^2 + \beta_1^2}, \dots, \frac{2\lambda}{\tau_p^2 + \beta_p^2} \right)$$

$$\arg \max_{\mathbf{d}} \{\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})\} = \left(\frac{2\lambda}{\tau_1^2 + \beta_1^2}, \dots, \frac{2\lambda}{\tau_p^2 + \beta_p^2} \right),$$

where $\tau_j^2 = \tau_0^2 / \{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\}$, $j = 1, \dots, p$. Further, if $\hat{\mathbf{d}} = \arg \max_{\mathbf{d}} \{\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})\}$, then

$$\pi_{GD}(\boldsymbol{\beta}, \hat{\mathbf{d}}) \propto \exp \left\{ -\tilde{\ell}_0(\boldsymbol{\beta} | \lambda, \tau_0) \right\}.$$

Proof of Lemma 2: We will find the arg max of the GD prior $\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})$ with respect to a single d_j for simplicity, as the form of $\hat{d}_j$ will be similar for each j . Recall,

$$\pi_{GD}(\boldsymbol{\beta}, \mathbf{d}) \propto \pi_G(\boldsymbol{\beta} | \mathbf{d}) \pi_D(\mathbf{d}),$$

where $\pi_G(\boldsymbol{\beta} | \mathbf{d})$ and $\pi_D(\mathbf{d})$. Note, the arg max of a function and its natural log are equivalent. We will work with the log of the GD prior for easier derivation. Observe,

$$\frac{\partial \log \pi_{GD}(\boldsymbol{\beta}, \mathbf{d})}{\partial d_j} = \frac{1}{2d_j} - \frac{\beta_j^2}{2} + \frac{2\lambda - 1}{2d_j} - \frac{\tau_0^2}{2\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\}}.$$

Then,

$$\left. \frac{\partial \log \pi_{GD}(\boldsymbol{\beta}, \mathbf{d})}{\partial d_j} \right|_{d_j = \hat{d}_j} = 0 \implies 2\lambda = \hat{d}_j \beta_j^2 + \hat{d}_j \frac{\tau_0^2}{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\}}$$

$$\begin{aligned} \implies 2\lambda &= \hat{d}_j(\beta_j^2 + \frac{\tau_0^2}{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\}}) \\ \implies \hat{d}_j &= \frac{2\lambda}{\tau_j^2 + \beta_j^2}, \end{aligned}$$

where $\tau_j^2 = \frac{\tau_0^2}{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\}}$, $j = 1, \dots, p$. Further,

$$\frac{\partial^2 \log \pi_{GD}(\boldsymbol{\beta}, \mathbf{d})}{\partial d_j^2} \Big|_{d_j = \hat{d}_j} = -\frac{1}{2\hat{d}_j^2} - \frac{2\lambda - 1}{2\hat{d}_j^2} = -\frac{\lambda}{\hat{d}_j^2} \leq 0.$$

Therefore,

$$\arg \max_{\mathbf{d}} \{\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})\} = (\frac{2\lambda}{\tau_1^2 + \beta_1^2}, \dots, \frac{2\lambda}{\tau_p^2 + \beta_p^2}).$$

Now, let $\hat{\mathbf{d}} = \arg \max_{\mathbf{d}} \{\pi_{GD}(\boldsymbol{\beta}, \mathbf{d})\}$, $BD_\psi(\cdot, \cdot)$ denote a Bregman divergence measure with ψ denoting a strictly convex function, and $\tilde{\ell}_0(\cdot|\lambda, \tau_0)$ denote the ℓ_0 -norm approximation. Since $\pi_{GD}(\boldsymbol{\beta}, \mathbf{d}) \propto \pi_G(\boldsymbol{\beta}|\mathbf{d})$ with respect to $\boldsymbol{\beta}$, for τ_0 small enough,

$$\begin{aligned} \pi_{GD}(\boldsymbol{\beta}, \hat{\mathbf{d}}) &\propto \prod_{j=1}^p \{\hat{d}_j^{1/2} \exp(-\frac{\hat{d}_j}{2} \beta_j^2)\} \\ &\propto \prod_{j=1}^p \{(2\lambda \frac{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2}{\tau_0^2 + \{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2})^{1/2} \exp(-\lambda \frac{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2}{\tau_0^2 + \{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2})\} \\ &\propto \exp\{-\lambda \sum_{j=1}^p (\frac{\{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2}{\tau_0^2 + \{(\mathbf{X}_{[j]})^T \mathbf{X}_{[j]}\} \beta_j^2})\} \\ &= \exp\{-\tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0)\}, \end{aligned}$$

which gives us the desired result.

Remark 1. Let $\hat{\mathbf{d}} = \arg \max_{\mathbf{d}} \left\{ \max_{\boldsymbol{\beta}} f_\psi(\mathbf{y}|\boldsymbol{\beta}) \pi_{GD}(\boldsymbol{\beta}, \mathbf{d}) \right\}$. Then, for any $\lambda \geq 0$ and $\tau_0 > 0$,

$$\arg \max_{\boldsymbol{\beta}} \{f_\psi(\mathbf{y}|\boldsymbol{\beta}) \pi_{GD}(\boldsymbol{\beta}, \hat{\mathbf{d}})\} = \arg \min_{\boldsymbol{\beta}} \{BD_\psi\{\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta})\} + \tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0)\}.$$

Proof of Remark 1: Observe that by applying the dual property between the likelihood function and loss function, we get

$$\begin{aligned} f_\psi(\mathbf{y}|\boldsymbol{\beta}) \pi_{GD}(\boldsymbol{\beta}, \hat{\mathbf{d}}) &\propto f_\psi(\mathbf{y}|\boldsymbol{\beta}) \exp\{-\tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0)\} \\ &\propto \exp\{-BD_\psi\{\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta})\}\} \exp\{-\tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0)\} \\ &= \exp\{-(BD_\psi\{\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta})\} + \tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0))\}. \end{aligned}$$

Hence,

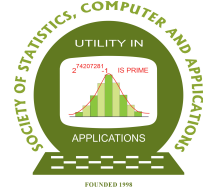
$$\arg \max_{\boldsymbol{\beta}} \{f_\psi(\mathbf{y}|\boldsymbol{\beta}) \pi_{GD}(\boldsymbol{\beta}, \hat{\mathbf{d}})\} = \arg \max_{\boldsymbol{\beta}} [\exp\{-(BD_\psi\{\mathbf{y}, \mathbf{h}(\mathbf{X}\boldsymbol{\beta})\} + \tilde{\ell}_0(\boldsymbol{\beta}|\lambda, \tau_0))\}].$$

Since the natural log is a monotonic strictly increasing function and the arg max of a function is the same as the arg min of the negative of that function for strictly convex (or concave) functions,

$$\begin{aligned} \underset{\beta}{\operatorname{argmax}}[\exp\{-(BD_{\psi}\{\mathbf{y}, \mathbf{h}(\mathbf{X}\beta)\} + \tilde{\ell}_0(\beta|\lambda, \tau_0))\}] &= \underset{\beta}{\operatorname{argmax}}\{-(BD_{\psi}\{\mathbf{y}, \mathbf{h}(\mathbf{X}\beta)\} + \tilde{\ell}_0(\beta|\lambda, \tau_0))\} \\ &= \underset{\beta}{\operatorname{argmin}}\{BD_{\psi}\{\mathbf{y}, \mathbf{h}(\mathbf{X}\beta)\} + \tilde{\ell}_0(\beta|\lambda, \tau_0)\}. \end{aligned}$$

Therefore, it follows that

$$\underset{\beta}{\operatorname{argmax}}\{f_{\psi}(\mathbf{y}|\beta)\pi_{GD}(\beta, \hat{\mathbf{d}})\} = \underset{\beta}{\operatorname{argmin}}\{BD_{\psi}\{\mathbf{y}, \mathbf{h}(\mathbf{X}\beta)\} + \tilde{\ell}_0(\beta|\lambda, \tau_0)\}.$$



Bayesian Predictive Inference in Logistic Regression with Covariates and Survey Weights

Lingli Yang¹ and Balgobin Nandram²

¹*Department of Statistics and Quantitative Sciences
Takeda Pharmaceutical Company Limited, MA, USA*

²*Department of Mathematical Sciences
Worcester Polytechnic Institute, MA, USA*

Received: 09 November 2025; Revised: 31 July 2026; Accepted: 04 August 2026

Abstract

In this study, we present a comprehensive Bayesian predictive inference framework for binary responses, using of probability survey samples with auxiliary data. The proposed framework integrates survey weights into a logistic regression model. We explore six distinct models comprising both unnormalized and normalized weighted likelihoods, with three variations of adjusted survey weights: original, trimmed, and calibrated. The logistic regression model is implemented utilizing the Metropolis-Hastings sampler. Subsequently, we employ the surrogate sampling and stratification approach to make predictive inferences on finite population parameters. The comparative performance and efficacy of the six models are assessed via a simulation study and a real-world dataset on body mass index. Our results reveal that models incorporating normalized density functions with adjusted trimmed weights demonstrate superior estimation accuracy while maintaining adherence to Bayesian principles, particularly in the presence of survey weight outliers. These findings underscore the significance of the proposed framework and the excellence of adjusted trimmed weights for effective predictive inference in survey-based studies.

Key words: Sampling theory; Survey weights; Bayesian inference; Pseudo-likelihood; Generalized linear models.

AMS Subject Classifications: 62D05, 62F15, 62J12

1. Introduction

In statistics, survey sampling is a vital methodology for obtaining valid and reliable data from a representative subset of a population. The goal of survey sampling is to reduce cost and obtain a sample with similar characteristics to the population in terms of key variables of interest. By carefully selecting a sample that is representative of the population, researchers can make predictions and inferences based on the sample data. Also, survey

sampling is utilized to make accurate estimates and predictions about the population as a whole.

In survey sampling, survey weights are used to adjust for any sampling biases that may be present in the sample data. They are used to ensure that the sample is representative of the population from which it was drawn. The use of survey weights is essential to ensure that inferences and estimates made from the sample data are accurate and unbiased (Lohr, 2009). However, incorrect or improper use of survey weights can lead to biased estimates and inaccurate inferences. The opening sentence in Gelman (2007) is, “Survey weighting is a mess. It is not always clear how to use weights in estimating anything more complicated than a simple mean or ratio, and standard errors are tricky even with simple weighted means.” Lohr (2007) responded in her comments, “I do not think that survey weighting is a mess, but I do think that many people ask too much of the weights.” We agree with Lohr (2007) that there is a limit to how much can be gained from survey weights, but we should make every effort to do so.

The process of weighting involves the calculation of weight for each unit in the sample, which is done by taking the inverse of the probability of selection. This allows for a correction of bias that may be present in the sample data due to the sampling design. For example, if some groups in the population have a lower probability of being selected, then the weight for each unit in those groups will be adjusted upward to ensure that the sample is representative of the population. In data sets from population surveys, the weights attached to the units can include adjustments for unit non-response, and post-stratification to match the distributions of auxiliary variables with known distributions in the population (Chen *et al.*, 2017). Also, it is possible to generate survey weights by using a single nonprobability sample. Elliott and Valliant (2017) used propensity scores to obtain surrogates for survey weights for nonprobability sample. Chen *et al.* (2020) used weighted likelihoods to obtain propensity scores, and Robbins *et al.* (2021) concentrated on design-based methods such as calibration and propensity score weighting (without parametric models). Nandram and Rao (2021) and Nandram *et al.* (2021) showed a full Bayesian approach to incorporate nonprobability sample and a probability sample. However, our concern is not about how to estimate survey weights, but rather to show the comparisons of different types of adjusted weights into a parametric model. Note that Yang *et al.* (2023) discussed the performance of nine methods but they did not take auxiliary information into consideration.

When the study variable is binary, Bayesian logistic regression is a widely used probabilistic model that holds the relationship between covariates and the study variable. It utilizes Bayesian inference to calculate the posterior distribution of the model parameters given the observed data. In contrast to traditional (frequentist) logistic regression, Bayesian logistic regression explicitly models the uncertainty in the model parameters and produces a distribution over the possible parameter values. This distribution can be used to make probabilistic predictions and to qualify the uncertainty in the model’s predictions. Although it is practical to weight units in order to help mitigate the effects of differential inclusion into the sample, the estimates that result can often be very ineffective.

Auxiliary information refers to additional data or variables that are not the primary focus of a statistical analysis but can provide valuable support or enhancement to the analysis (Srivastava, 1967). In our context, covariates (independent variables) are often considered

auxiliary information. While the response variable is the main focus, including covariates in the model can help account for additional variability and better capture the relationships within the data. The covariates not only provide auxiliary information, but also are an essential component in the modeling that focuses on regression and the estimation of the corresponding regression coefficients. Roberts *et al.* (1987) and Archer and Lemeshow (2006) discussed different test statistics for logistic regression analysis that are adapted to include the survey design. Rader *et al.* (2017) proposed weighted estimating equations for logistic regression models to correct bias of estimates. Chen and Nandram (2023) described a hierarchical Bayesian logistic regression model to capture the variation in the binary data. The same idea is shown in the work of Barasa and Muchwanju (2015), but they only analyzed the regression coefficients, rather than making predictions and inferences about a finite population quantity.

Predictive inference can be obtained using surrogate sampling. This model is adjusted using the survey weights to account for selection bias. After this adjustment, we obtain the posterior distribution of the model parameters, which are interpreted as population parameters. At this stage, the original sample data and their weights are no longer needed. Using the posterior parameters, we then generate the full population from the population model, which is referred to as surrogate sampling (Nandram, 2007). For example, suppose we have a population model with the population size N ,

$$y_1, \dots, y_N | \theta \stackrel{\text{ind}}{\sim} f(y | \theta).$$

With the observed sample data $(w_i, y_i), i = 1, \dots, n$ and a prior on θ , we are able to obtain the posterior density $\pi(\theta | y_s)$. Suppose the focus is on the mean of the finite population, denoted as $\bar{Y}$. Subsequently, Bayesian predictive inference regarding the mean of the finite population is acquired through the following process,

$$\pi(\bar{Y} | y_s) = \int f(\bar{Y} | \theta) \pi(\theta | y_s) d\theta,$$

where $f(\bar{Y} | \theta)$, the population model, does not depend on y_s ; see Nandram and Rao (2021). We simply draw θ from the posterior density $\pi(\theta | y_s)$ and draw $\bar{Y}$ from the population model. In general, the available information in this study is the study variable and the covariates of the survey sample. Based on the surrogate sampling method, we incorporate the survey weights into the population model to obtain the sample distribution.

The main goal of this study is to evaluate the performance of six models to do Bayesian predictive inference of finite population interests. In Section 2, we explain three types of adjusted survey weights: adjusted original survey weights, adjusted trimmed survey weights and adjusted calibrated survey weights. Then we incorporate weights into the unnormalized model and normalized model to access Bayesian predictive inference. In Section 3, a simulation study is carried out to further understand these six models by the absolute relative bias (ARB), the posterior standard deviation (PSD), the posterior root mean squared error (PRMSE), the coverage probability (CP), and the width of highest posterior density interval (Wid). In Section 4, we present an application using body mass index (BMI) data from the Third National Health and Nutrition Examination Survey (NHANES III) and compare the posterior means and posterior standard deviations of these six models. Finally, Section 5 gives some recommendations for further research after reviewing the study's advantages and disadvantages in greater depth.

2. Bayesian methodology

In this section, we establish a general framework for statistical inference of a binary variable from probability survey samples when pertinent auxiliary data are available. In particular, we demonstrate how to include survey weights in a logistic regression model. The survey weights are included using six different methods, which use either unnormalized or normalized weighted likelihoods. We employ three different kinds of adjusted survey weights, based on the original survey weights, trimmed survey weights, and calibrated survey weights.

2.1. Adjusted survey weights

Assuming that the selection probability only depends on the covariates that are observed (*i.e.*, ignorable selection), for unit i , let y_i be a vector of response; x_i be a vector of observed covariates. When a sampling plan is used in a finite population of size N to draw a sample of size n , with given survey weights in the sample, $W_1, \dots, W_n$, Horvitz-Thompson estimators of population total and population size are,

$$\hat{T} = \sum_{i=1}^n W_i y_i, \quad \hat{N} = \sum_{i=1}^n W_i.$$

These estimators are typically applied in a similar ways and are design-unbiased.

Weight trimming is a technique used to handle outliers in a dataset by replacing extreme values with less extreme ones (Haziza and Beaumont, 2017). This can reduce the impact of outliers on statistical analyses. Specifically, weight trimming replaces extreme values with either the next smallest or next largest non-outlier value in the dataset, up to a certain percentile or quantile threshold. It can help improve the robustness and reliability of statistical analyses; see Rao (1966) and Basu (2010).

In our work, $Q_3 + 1.5(Q_3 - Q_1)$ is considered as the threshold, where Q_1 is the first quartile, Q_3 is the third quartile. Let W^* be weights after trimming and W_0 be the threshold, where $W_0 = Q_3 + 1.5(Q_3 - Q_1)$, then

$$W_i^* = \begin{cases} W_0, & W_i \geq W_0 \\ aW_i, & W_i < W_0 \end{cases}, \quad (1)$$

where a is a rescaling parameter so that $\sum_{i=1}^n W_i^* = \sum_{i=1}^n W_i = \hat{N}$, ensuring the summation of weights would not be changed after the trimming. This rescaling step is important to keep the HT estimator of population size the same, but we can not trim too much. The choice of W_0 has been discussed by Potter (1990); see also Chen *et al.* (2017) for an explanation of weight trimming processes. In our case, we use $Q_3 + 1.5(Q_3 - Q_1)$, which is the common rule in boxplot construction to detect outliers. It is true that weight trimming can lead to biased estimates, but it will increase the effective sample size.

Calibrated survey weights were used to adjust for the under-representation of certain population groups in the sample (Haziza and Beaumont, 2017). This was necessary to improve the accuracy and reliability of the study's estimates of obesity prevalence in the finite population. Calibrated survey weights are a common technique used in survey research to adjust for non-representative samples or under-representation of specific subgroups.

Assume we can find the totals of each covariate, $\underline{t}$, through a census, government documents, or online scraping. By the calibration equations, $\sum_{j=1}^n \tilde{W}_j x_j = \underline{t}$, survey weights are calibrated as $\tilde{W}$. As proposed by Haziza and Beaumont (2017), a distance function $G(u)$ is used to make sure $\tilde{W}$ are as close as possible to W , where $G(u) \geq 0$ and $G(1) = 0$; $G(u)$ is differentiable, say $g(u) = G'(u)$, $g(1) = 0$, and strictly convex. The coefficient q_j is the scale factor indicating the importance of unit j in the distance calculation. In most practical situations, the factor q_j is set to 1. Let the q_j be set to 1, then the calibrated weights are

$$\begin{aligned} \underset{\tilde{W}_j}{\operatorname{argmin}} \quad & \sum_{j=1}^n \frac{\tilde{W}_j}{q_j} G\left(\frac{\tilde{W}_j}{W_j}\right) \\ \text{s.t.} \quad & \sum_{j=1}^n \tilde{W}_j x_j = \underline{t}. \end{aligned} \quad (2)$$

To find the minimal value of a function subject to equality constraint, Lagrangian multipliers, $\underline{\lambda} = (\lambda_1, \dots, \lambda_p)'$, are commonly used. We have,

$$\phi(\tilde{W}_1, \dots, \tilde{W}_n, \underline{\lambda}) = \sum_{j=1}^n \frac{\tilde{W}_j G\left(\frac{\tilde{W}_j}{W_j}\right)}{q_j} - \underline{\lambda}' \left(\sum_{j=1}^n \tilde{W}_j x_j - \underline{t} \right). \quad (3)$$

After we obtain $\hat{\underline{\lambda}}$ (Haziza and Beaumont, 2017; Nandram *et al.*, 2021), the calibrated weights are straightforward,

$$\tilde{W}_j = W_j g^{-1}\left(q_j \hat{\underline{\lambda}}' x_j\right), j = 1, \dots, n. \quad (4)$$

In our study, we consider the simple Euclidean distance, $G(u) = \frac{1}{2}(u - 1)^2$, to have the closed-form solutions. Let

$$A = \sum_{j=1}^n q_j W_j x_j x_j', \quad \underline{b} = \underline{t} - \sum_{j=1}^n W_j x_j. \quad (5)$$

Therefore, by solving $A\underline{\lambda} = \underline{b}$ for $\hat{\underline{\lambda}}$ and assuming A is invertible, we have $\hat{\underline{\lambda}} = A^{-1}\underline{b}$ and the calibrated weights are,

$$\tilde{W}_j = W_j \left(1 + q_j \hat{\underline{\lambda}}' x_j\right), j = 1, \dots, n. \quad (6)$$

It is worth noting that if $\hat{\underline{\lambda}} = \underline{0}$, there are no differences between original survey weights and calibrated ones. In addition, negative calibrated weights can occur when the sample selected for the survey is not representative of the population in some way. This can happen for a variety of reasons, such as non-response bias, sampling error, or the use of an inappropriate weighting adjustment procedure. When this happens, the calibrated weights may be negative for some observations in the sample.

To avoid the negative calibrated weights, we rescale the negative weights to be positive. This is done by setting the weights that less than 1 to 1, because the smallest representative size should be greater or equal to 1, then normalized them to sum to the original total weights so that the sample size is preserved.

Effective sample size is a measure of the amount of independent information contained in a sample of data. In statistics, it is used to adjust for the fact that the observed sample size may not reflect the actual amount of independent information in the data due to issues such as dependence between observations, clustering of data, or non-random sampling (Potthoff *et al.*, 1992). A larger effective sample size indicates that the sample contains more independent information, and therefore provides more accurate estimates and narrower confidence intervals. Conversely, a smaller effective sample size indicates that the sample contains less independent information, which can result in less precise estimates and wider confidence intervals. Let n_e denote the effective sample size, and w be adjusted original weights, w^* be adjusted trimmed weights, $\tilde{w}$ be adjusted calibrated weights. For $i = 1, \dots, n$, we construct three types of adjusted weights,

$$n_e = \frac{\left(\sum_{j=1}^n W_j\right)^2}{\sum_{j=1}^n W_j^2}, \quad w_i = \frac{n_e W_i}{\sum_{j=1}^n W_j}, \quad (7)$$

$$n_e^* = \frac{\left(\sum_{j=1}^n W_j^*\right)^2}{\sum_{j=1}^n W_j^{*2}}, \quad w_i^* = \frac{n_e^* W_i^*}{\sum_{j=1}^n W_j^*}, \quad (8)$$

$$\tilde{n}_e = \frac{\left(\sum_{j=1}^n \tilde{W}_j\right)^2}{\sum_{j=1}^n \tilde{W}_j^2}, \quad \tilde{w}_i = \frac{\tilde{n}_e \tilde{W}_i}{\sum_{j=1}^n \tilde{W}_j}. \quad (9)$$

So far, we described the details of these three types of adjusted weights; adjusted original weights w , adjusted trimmed weights w^* , and adjusted calibrated weights $\tilde{w}$. They are more accurate and reliable than unadjusted ones. In addition, adjusted trimmed weights are not able to be unduly influenced by a few extreme observations, and can help improve the robustness of the analysis to data that may not confirm to certain assumptions, such as the normality of the data or the homogeneity of variance (Lohr, 2009). Survey weights are designed to reflect the representativeness of sampled units about the broader population. Extreme weights may result from various factors, such as nonresponse, oversampling of certain groups, or errors in the weight calculation process. For example, if the extreme survey weights are caused by oversampling of certain groups, the adjusted trimmed weights downplay the contribution of these values, reducing their impact on the analysis. In this case, the impact could be the normality of the data or the homogeneity of variance. Designed to adjust for differences between the sample and the population on certain variables of interest, such as age, gender, or education level, calibrated weights can help correct for potential biases and improve the representativeness of survey data.

2.2. Weighted density

Suppose that a probability sample of size n is taken from a finite population,

$$y_1, \dots, y_n \mid \underline{\theta} \stackrel{iid}{\sim} f(y \mid \underline{\theta}).$$

With given survey weights of each unit in the sample, W , the Horvitz-Thompson estimator of the log-likelihood for the entire population, $L(\underline{\theta})$, is

$$\widehat{L}(\underline{\theta}) = \sum_{i=1}^n W_i \log \{f(y_i \mid \underline{\theta})\},$$

where we use W_i to illustrate our models, but replace the weights with adjusted survey weights w_i in application.

The density function is $g(y_i \mid \underline{\theta}) \propto f(y_i \mid \underline{\theta})^{W_i}, i = 1, \dots, n$.

However, $f(y_i \mid \underline{\theta})^{W_i}$ is an unnormalized ‘density’ function. We need to insert the normalization constant, $\int f(y_i \mid \underline{\theta})^{W_i} dy_i$, such that $g(y_i \mid \underline{\theta})$ integrates to 1. Therefore, we have,

$$g(y_i \mid \underline{\theta}) = \frac{(f(y_i \mid \underline{\theta}))^{W_i}}{\int (f(y_i \mid \underline{\theta}))^{W_i} dy_i}.$$

When the normalization constant is a function of $\underline{\theta}$, it is important to keep this part to follow the rules of Bayesian theory (see more in Nandram and Rao, 2024). The denominator can be a function of $\underline{\theta}$ and can not be ignored.

In this study, we focus on the binary response; see more examples in Yang *et al.* (2023) and details in Appendix B. We assume logistic regression with p covariates, including an intercept, for the population model,

$$y_i \mid \underline{\beta} \stackrel{ind}{\sim} \text{Bernoulli} \left\{ \frac{e^{\underline{x}'_i \underline{\beta}}}{1 + e^{\underline{x}'_i \underline{\beta}}} \right\}, i = 1, \dots, N, \quad (10)$$

where N , the population size, and the non-sampled $\underline{x}_i$ may be unknown, and these can come from an external source or from a calibrated bootstrap of the sampled covariates. Note that there are no survey weights in the population model.

For the sample, we have $(W_i, \underline{x}_i, y_i), i = 1, \dots, n$. Without the normalization,

$$f_1(y_i \mid \underline{\beta}) = \left(\frac{e^{\underline{x}'_i \underline{\beta}}}{1 + e^{\underline{x}'_i \underline{\beta}}} \right)^{W_i y_i} \left(\frac{1}{1 + e^{\underline{x}'_i \underline{\beta}}} \right)^{W_i (1 - y_i)}, i = 1, \dots, n, \quad (11)$$

independently, but note that this is not probability mass function in y_i , despite the fact that it is typically employed in model-based analysis of data derived from survey sampling. With normalization,

$$f_2(y_i|\theta) = \frac{(f_1(y_i|\theta))^{W_i}}{f_1(y_i=0|\theta)^{W_i} + f_1(y_i=1|\theta)^{W_i}} \quad (12)$$

$$= \frac{\left(\frac{e^{x'_i\beta}}{1+e^{x'_i\beta}}\right)^{y_i W_i} \left(\frac{1}{1+e^{x'_i\beta}}\right)^{(1-y_i)W_i}}{\left(\frac{e^{x'_i\beta}}{1+e^{x'_i\beta}}\right)^{W_i} + \left(\frac{1}{1+e^{x'_i\beta}}\right)^{W_i}} \quad (13)$$

$$= \left(\frac{e^{W_i x'_i \beta}}{1+e^{W_i x'_i \beta}}\right)^{y_i} \left(\frac{1}{1+e^{W_i x'_i \beta}}\right)^{(1-y_i)}. \quad (14)$$

Therefore,

$$y_i | \beta \stackrel{\text{ind}}{\sim} \text{Bernoulli} \left\{ \frac{e^{W_i x'_i \beta}}{1+e^{W_i x'_i \beta}} \right\}, i = 1, \dots, n. \quad (15)$$

Once more, the Bayesian paradigm favors the normalized form (*i.e.* the covariates and weights) to provide coherent.

Then, using a flat prior on β , $\pi(\beta) = 1$, the joint pseudo-posterior density and the joint posterior density are,

$$\pi_1(\beta | y) \propto \left\{ \frac{e^{\sum_{i=1}^n y_i W_i x'_i \beta}}{\prod_{i=1}^n (1+e^{x'_i \beta})^{W_i}} \right\}, \beta \in R^p, \quad (16)$$

$$\pi_2(\beta | y) \propto \left\{ \frac{e^{\sum_{i=1}^n y_i W_i x'_i \beta}}{\prod_{i=1}^n (1+e^{W_i x'_i \beta})} \right\}, \beta \in R^p. \quad (17)$$

For both posterior distributions, the prior is assumed to be a uniform distribution on β ; see Chen *et al.* (2008) for more details about Jeffreys' prior.

Note that the Pólya–Gamma data augmentation for logistic regression (Polson *et al.*, 2013) cannot be applied generally in our setting. In the expression for $\pi_2(\beta | y)$ in (17), we can rewrite x'_i as $\tilde{x}_i$, in a form that allows the Pólya–Gamma augmentation to be used entirely within (17). However, this reparameterization is not possible for the expression in (16), and therefore the Pólya–Gamma data augmentation cannot be applied there.

In this application, for the sake of generality, we instead use a more flexible and general Metropolis–Hastings sampler to obtain samples of β ; see Appendix A for more details. For both posteriors, large values of W_i will give unacceptably small variance, so it is necessary to replace the original weights with adjusted weights. After we replace the original survey weights with three adjusted weights in these two posteriors, we have six models, and they are illustrated in the following flowchart; see Figure 1.

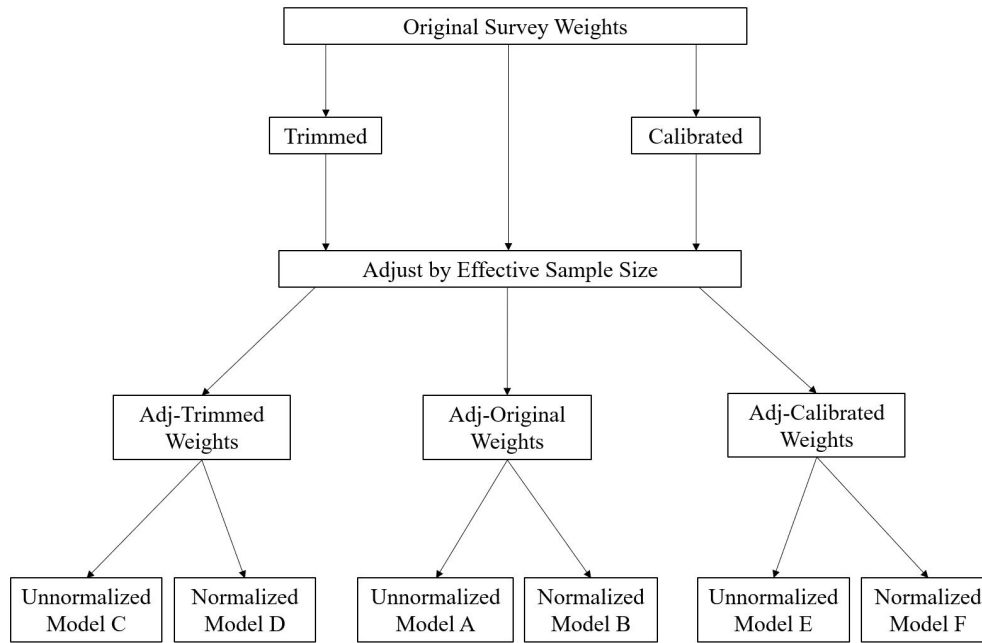


Figure 1: A flowchart of the six models

2.3. Bayesian predictive inference for logistic regression

To perform Bayesian predictive inference, we first specify a prior distribution for the model parameters $\beta_1, \beta_2, \dots, \beta_{p-1}$. This prior can be chosen based on our prior beliefs or knowledge about the relationship between the predictor variables and the outcome. We then update this prior using Bayes' theorem to obtain the posterior distribution over the parameters given the observed data,

$$\pi(\beta|y, x) \propto \pi(y|x, \beta)\pi(\beta),$$

where $\pi(y|x, \beta)$ is the likelihood function, which specifies the probability of the observed data (y) given the predictor variables (x) and the model parameters (β), and $\pi(\beta)$ is the prior distribution over the parameters. It can be shown that under mild conditions, $\pi(\beta|y, x)$ is proper.

Bayesian predictive inference for logistic regression is a method used to make predictions about the probability of a binary outcome (*e.g.*, success/failure) based on a set of predictor variables. In this approach, we start with a prior distribution over the model parameters and update this distribution based on the observed data using Bayes' theorem. The resulting posterior distribution over the parameters can then be used by the stratification method to make predictive inferences about the population.

Specifically, we use a logistic regression model of the form,

$$\log \frac{p(y = 1|x)}{1 - p(y = 1|x)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{p-1} x_{p-1},$$

where $p(y = 1|x)$ is the probability of the binary outcome ($y = 1$) given the predictor variables (x), β_0 is the intercept, $\beta_1, \beta_2, \dots, \beta_{p-1}$ are the coefficients for the predictor variables

$x_1, x_2, \dots, x_{p-1}$, and the logistic function maps the linear combination of predictor variables to a probability between 0 and 1. Once we have obtained the posterior distribution over the parameters, we can use it to make predictions about the probability of the binary outcome for new observations.

In this study, one of the primary challenges is the estimation of non-sampled variables. These are essential for predicting the proportion of the finite population but are typically unknown. To address this, we use a form of stratification to make predictive inferences about covariates of the population.

In this approach, the population is divided into K distinct strata or cells. Under the assumption that the sample covariates sufficiently represent the population, particularly when the sample size is large, unobserved variables can be structurally excluded, because the sampled covariates are likely to capture a broad spectrum of the population characteristics.

For the k -th stratum, the corresponding variables x_k are known and every unit has the same covariates. Once the successful stratification of the population, the reliance on the estimation of non-sampled variables, which is a hard and resource-intensive task, is efficiently eliminated. Then, we can use the Horvitz-Thompson estimator for each stratum by aggregating the original survey weights of each unit in the stratum, denoted by $\hat{N}_k = \sum_{i \in \kappa} W_i$, where κ is the set of indexes in stratum k .

Surrogate sampling (Nandram, 2007) in Bayesian predictive inference involves obtaining predictions for a target parameter or quantity of interest by drawing samples from the posterior distribution of the model parameters. Instead of directly simulating from the predictive distribution, surrogate sampling simplifies the process by breaking it into two steps. Firstly, obtaining samples from the posterior distribution of the model parameters given the observed data by using Bayesian inference techniques, such as Markov chain Monte Carlo (MCMC) or other sampling methods. Secondly, for each set of sampled parameters, draw predictions for the quantity of interest using the population model. The population model represents the assumed distribution of the data given the model parameters. The term “surrogate” reflects the fact that the samples are used as substitutes for direct samples from the predictive distribution, and the whole population is sampled.

Therefore, according to the population distribution (Equation 10), we have,

$$t_k \mid \beta \stackrel{ind}{\sim} \text{Binomial} \left\{ \hat{N}_k, \frac{e^{x'_k \beta}}{1 + e^{x'_k \beta}} \right\}, k = 1, \dots, K, \quad (18)$$

where t_k is the total of the binary outcome ($y = 1$) in this stratum, and K denotes the number of unique combinations of covariates. For example, in our study, we only focus on age (20-90), race (0 or 1), and sex (0 or 1), which means there are 284 combinations of these covariates in the population. The estimate of the finite population proportion is then given by,

$$\hat{T} = \frac{\sum_{k=1}^K t_k}{\sum_{k=1}^K \hat{N}_k}.$$

Moreover, this method, which is a form of stratified analysis, is particularly feasible when all covariates are discrete, each having only a few levels, or when a specific continuous

variable can be discretized to a limited number of levels. The discretization of continuous variables can simplify their handling, enabling the application of methods designed for discrete data. It also helps to reduce the complexity and computational load of the methods.

Furthermore, when auxiliary information is available in the form of known marginal counts, the raking approach can be used as an alternative and more robust procedure within strata. However, this auxiliary information is typically not readily available in most instances, and survey weights are normally used as we have done, so prediction using the already fitted logistic regression is preferred. For example, consider a single stratum with s successes and f failures, a sample size of $n = s + f$, with a stratum size of N . Then, raking allocates N units proportionally to get $(s/n)N$ and $(f/n)N$. But, of course, this proportionality assumption may not be true; see Deville *et al.* (1993) for general multi-way tables using iterative proportional fitting (IPF).

To summarize, in this approach, the population is divided into K distinct strata or cells. Under the assumption that the sample covariates sufficiently represent the population, particularly when the sample size is large, unobserved variables can be structurally excluded, because the sampled covariates are likely to capture a broad spectrum of the population characteristics. Then, we use the estimated sizes (sum of survey weights) for each stratum. By dividing sampled variables into distinct strata, we can efficiently make predictive inferences for a finite population. This is achieved using the covariates and the cumulative weights of each stratum, and the entire stratum has just one covariate vector. This stratification strategy enables us to effectively use sampled variables for estimation, thus eliminating the need for estimating non-sampled variables, simplifying the process, and bolstering the reliability of our results.

3. Simulation

In this section, we generate the finite population and the sample using the design-based method, where we also vary the strength of the association between the response and covariates, in accordance with Nandram and Rao (2021) and Chen *et al.* (2020). Age, race, and sex, the three covariates, determine the structure of the simulation data as follows:

$$z_i = 24.2449 + .0559x_{1i} + 1.2656x_{2i} + 1.2525x_{3i} + \epsilon_i, \quad (19)$$

$$y_i = \begin{cases} 1, & z_i \geq 30 \\ 0, & z_i < 30 \end{cases}, \quad i = 1 \dots N, \quad (20)$$

where,

$$\begin{aligned} x_{1i} &\overset{ind}{\sim} \text{DiscreteUniform}(20, 90), \\ x_{3i} &\overset{ind}{\sim} \text{Bernoulli}(0.5), \\ x_{2i} \mid x_{1i}, x_{3i} &\overset{ind}{\sim} \text{Bernoulli} \left\{ \frac{e^{a_i}}{1 + e^{a_i}} \right\}, \\ \epsilon_i &\overset{iid}{\sim} \text{Normal} \left(0, \sigma^2 \right), \end{aligned}$$

with $a_i = \{24.2449 + .0559x_{1i} + 1.2525x_{3i}\}^{\frac{1}{10}}$. In order to control the correlation coefficient ρ between z_i and the linear predictor $\tilde{x}_i' \tilde{\beta}$ at 0.2, 0.3, 0.5, and 0.8 for the simulation studies,

the values of σ are selected as 6.8638, 4.4551, 2.4267, and 1.0508, respectively. A Monte Carlo check with $N = 10^6$ reproduces the target correlations to within 0.0003.

Then, the probability sample with the target sample size $n = 200$ is taken from the population with population size, $N = 10,000$, by the randomized systematic probability-proportional-to-size (PPS) sampling method with the selection probabilities,

$$\pi_i = \frac{nb_i}{\sum_{i=1}^N b_i}, \quad i = 1 \dots N,$$

with $b_i = \theta + x_{1i} + 0.2x_{2i} + 0.1x_{3i}$, and θ is chosen by trial and error to control the variation of the survey weights such that $\max\{b_i\} / \min\{b_i\} \approx 50$. After we use the PPS sampling method with π to select samples, the survey weight of each unit in the sample is $W_i = 1/\pi_i, i = 1, \dots, n$.

Two scenarios are considered: scenario 1 using the generated samples and scenario 2 using the generated samples with manually added weights outliers. For scenario 2, we randomly select 10 samples where the response variable is 0 and multiply the corresponding weights by 5. These ten units are considered outliers and the outlier fraction is 5% when sample size is 200. Then, we can calculate these three adjusted weights based on $\tilde{W}$. The outliers are manually generated to create the bias that may be caused by nonresponse, oversampling of certain groups, or errors in the weight calculation process. We do not make any changes on the population. For the adjusted calibrated weights, assume the totals of the population are obtainable. Note that we include the survey weights in the logistic regression model because the key variables used in the model do not fully account for the complex survey design, such as the nonresponse issue. Nandram and Choi (2010) have shown that incorporating weights into the model can provide more accurate assessments of obesity and overweight in children and adolescents.

In this simulation, our parameter of interest is the finite population proportion. For a given estimator, its performance is evaluated through the absolute relative bias (ARB), the posterior standard deviation (PSD), the posterior root mean squared error (PRMSE), the coverage probability (CP), and the width of the highest posterior density interval (Wid) computed as

$$\begin{aligned} \text{ARB} &= \frac{1}{H} \sum_{h=1}^H \left| \frac{PM^{(h)} - T}{T} \right|, \\ \text{PSD} &= \frac{1}{H} \sum_{h=1}^H PSD^{(h)}, \\ \text{PRMSE} &= \frac{1}{H} \sum_{h=1}^H \sqrt{(PM^{(h)} - T)^2 + (PSD^{(h)})^2}, \\ \text{CP} &= \frac{1}{H} \sum_{h=1}^H I\left(C_{025}^{(h)} \leq T \leq C_{975}^{(h)}\right), \\ \text{Wid} &= C_{975}^{(h)} - C_{025}^{(h)}, \end{aligned}$$

where T is the true finite population proportion, $PM^{(h)}$ is the posterior mean, $PSD^{(h)}$ is the posterior standard deviation, $I(\cdot)$ is the indicator function, $C_{025}^{(h)}$ and $C_{975}^{(h)}$ denote the

endpoints, not necessarily percentiles. All were obtained from the h^{th} simulated sample with $H = 1,000$ as the total number of simulation runs.

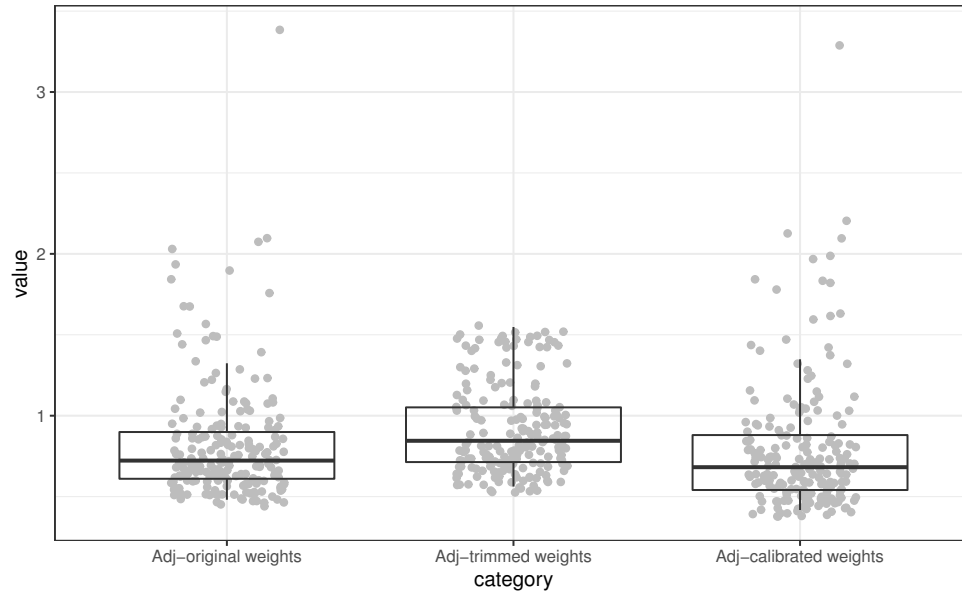


Figure 2: Comparison boxplots with three different weights with outliers using one simulation data when $n = 200$ and there is 5% weights outliers.

Figure 2 is the boxplot of one simulation run under scenario 2 with weights outliers, which indicates that the median, quartiles, and range of the adjusted calibrated weights are very similar to those of the adjusted original weights. Note that this calibration method could be useful in other cases when the weights need more adjustment. As we trimmed some weights outliers and reassigned weights to ensure $\sum_{i=1}^n W_i^* = \sum_{i=1}^n W_i = \hat{N}$, the boxplot of adjusted trimmed weights is a little higher and more consistent than others, without any outliers. Their corresponding effective sample sizes are $n_e = 162$, $n_e^* = 184$, $\tilde{n}_e = 147$ out of the real sample size $n = 200$ in this run.

Table 1 displays simulation comparisons in the absence of weights outliers. When there are no outliers of weights in the selected samples, models with unnormalized densities (model A, C, and E) outperform models with normalized densities (model B, D, and F). However, in surveys with complex designs, large-scale surveys covering diverse populations, or surveys experiencing high rates of nonresponse or measurement error, we must consider the occurrence of survey weights outliers (Nandram and Choi, 2010).

Table 2 presents results under the scenario with 5% weights outliers. In this scenario, it is reasonable that the assessment metrics of the model ignoring weights are better in general compared to when both weights with outliers are incorporated. As we generate samples using the PPS sampling method and specifically introduce weights outliers to the sample weights, we focus on examining the discrepancies arising from different types of weights and their associated densities. It shows that model D appears to be a better choice compared to other models (A, B, C, E, and F) because it consistently demonstrates lower bias, better precision, and higher overall performance in estimating the parameter of interest. Models using unnormalized densities with adjusted original weights (model A) and adjusted

Table 1: Comparison of six models using simulation data with different correlations and weights without outliers

	ρ	no wgt	A	B	C	D	E	F
ARB	0.2	0.076	0.087	0.126	0.077	0.080	0.086	0.132
	0.3	0.076	0.082	0.130	0.076	0.079	0.082	0.128
	0.5	0.106	0.110	0.171	0.106	0.131	0.109	0.172
	0.8	0.096	0.096	0.127	0.096	0.126	0.097	0.134
PSD	0.2	0.027	0.031	0.041	0.025	0.025	0.032	0.043
	0.3	0.026	0.033	0.044	0.025	0.025	0.032	0.042
	0.5	0.023	0.029	0.033	0.022	0.023	0.029	0.035
	0.8	0.016	0.022	0.020	0.017	0.017	0.023	0.020
PRMSE	0.2	0.046	0.054	0.073	0.046	0.046	0.054	0.076
	0.3	0.043	0.050	0.073	0.042	0.043	0.049	0.071
	0.5	0.047	0.051	0.073	0.046	0.055	0.051	0.074
	0.8	0.034	0.038	0.044	0.034	0.042	0.038	0.046
CP	0.2	0.946	0.951	0.933	0.945	0.941	0.952	0.939
	0.3	0.956	0.965	0.948	0.962	0.941	0.958	0.960
	0.5	0.905	0.960	0.842	0.918	0.796	0.921	0.863
	0.8	0.910	0.953	0.819	0.923	0.810	0.919	0.841
Wid	0.2	0.107	0.124	0.161	0.100	0.100	0.127	0.168
	0.3	0.105	0.129	0.174	0.098	0.098	0.125	0.167
	0.5	0.092	0.113	0.133	0.088	0.093	0.114	0.139
	0.8	0.064	0.089	0.078	0.066	0.068	0.089	0.080

calibrated weights (model E) have similar results across all ρ levels, which suggests that there are no big differences between adjusted original weights w and adjusted calibrated weights $\tilde{w}$, and the same situation for normalized densities with adjusted original weights (model B) and adjusted calibrated weights (model F). The models with adjusted trimmed weights (model C, and D), however, exhibit more substantial difference in these measurements, especially at lower ρ levels (0.2 and 0.3), with lower ARB, PRMSE, and higher CP. With the setting of weights outliers, the adjusted trimmed weights show their advantage in this simulation.

It is also notable that at lower ρ levels (0.2 and 0.3), normalized model with adjusted trimmed weights (model D) performs better than unnormalized model with adjusted trimmed weights (model C), where model D tends to have relatively low or close ARB values, and higher CP values.

In our context, outliers refer specifically to survey units with extreme weights, meaning weights that are much larger (or smaller) than those of the other sampled units. Such extreme weights are common in survey data and, in binary-outcome settings, can strongly influence predictive inference for the population.

To examine the impact of these weight outliers, we conduct a simulation in which we generate synthetic data under varying levels of weight contamination, with the fraction of outlier units ranging from 0 to 0.5. For each scenario, we randomly select a subset of sampled units according to the specified outlier fraction and inflate their original weights

Table 2: Comparison of six models using simulation data with different correlations and weights with 5% outliers

	ρ	no wgt	A	B	C	D	E	F
ARB	0.2	0.076	0.145	0.240	0.074	0.068	0.145	0.222
	0.3	0.094	0.149	0.253	0.092	0.088	0.151	0.237
	0.5	0.090	0.158	0.247	0.091	0.088	0.159	0.251
	0.8	0.116	0.176	0.195	0.124	0.114	0.176	0.210
PSD	0.2	0.028	0.036	0.045	0.026	0.026	0.035	0.044
	0.3	0.027	0.035	0.045	0.025	0.025	0.034	0.043
	0.5	0.023	0.034	0.037	0.023	0.024	0.032	0.036
	0.8	0.016	0.025	0.020	0.016	0.017	0.024	0.020
PRMSE	0.2	0.046	0.076	0.118	0.044	0.042	0.076	0.110
	0.3	0.049	0.073	0.117	0.048	0.046	0.074	0.110
	0.5	0.042	0.069	0.099	0.042	0.042	0.068	0.100
	0.8	0.039	0.058	0.061	0.041	0.039	0.058	0.065
CP	0.2	0.965	0.962	0.835	0.973	0.982	0.945	0.841
	0.3	0.962	0.941	0.840	0.966	0.968	0.881	0.808
	0.5	0.899	0.954	0.794	0.942	0.975	0.940	0.776
	0.8	0.820	0.910	0.648	0.809	0.872	0.853	0.641
Wid	0.2	0.110	0.143	0.177	0.103	0.102	0.138	0.173
	0.3	0.105	0.140	0.176	0.100	0.099	0.135	0.169
	0.5	0.092	0.134	0.147	0.089	0.093	0.125	0.141
	0.8	0.063	0.098	0.081	0.065	0.068	0.096	0.081

by a factor of five. We then apply our full procedure to obtain posterior means under six competing models and compare their performance using average relative bias (ARB).

The performance of six models at various outlier fraction levels is shown in Figure 3. When the outlier fraction is 0, the result shows that there are no significant differences between any of the six models. The models using adjusted trimmed weights (models C and D) perform better than the other models when the outlier fraction rises. This result was anticipated since weight trimming can reduce the influence of weights with outliers.

It is worth noting that when outlier fractions at a lower lever (0.1, 0.2, and 0.3), normalized density (model D) appears to have a lower ARB values than unnormalized density (model C). Nevertheless, the normalized densities (models B and F) have higher ARB values than the unnormalized densities (models A and E) across all the outlier fractions. There are no significant differences between unnormalized models using adjusted original weights (model A) and adjusted calibrated weights (model E), and the same applies to normalized models (models B and F).

4. Application on body mass index

The proportion of the population that has been identified as being obese is an important indicator to research when looking at the health of a finite population. In this section, we illustrate our six models by using sample BMI data of individuals from eight counties

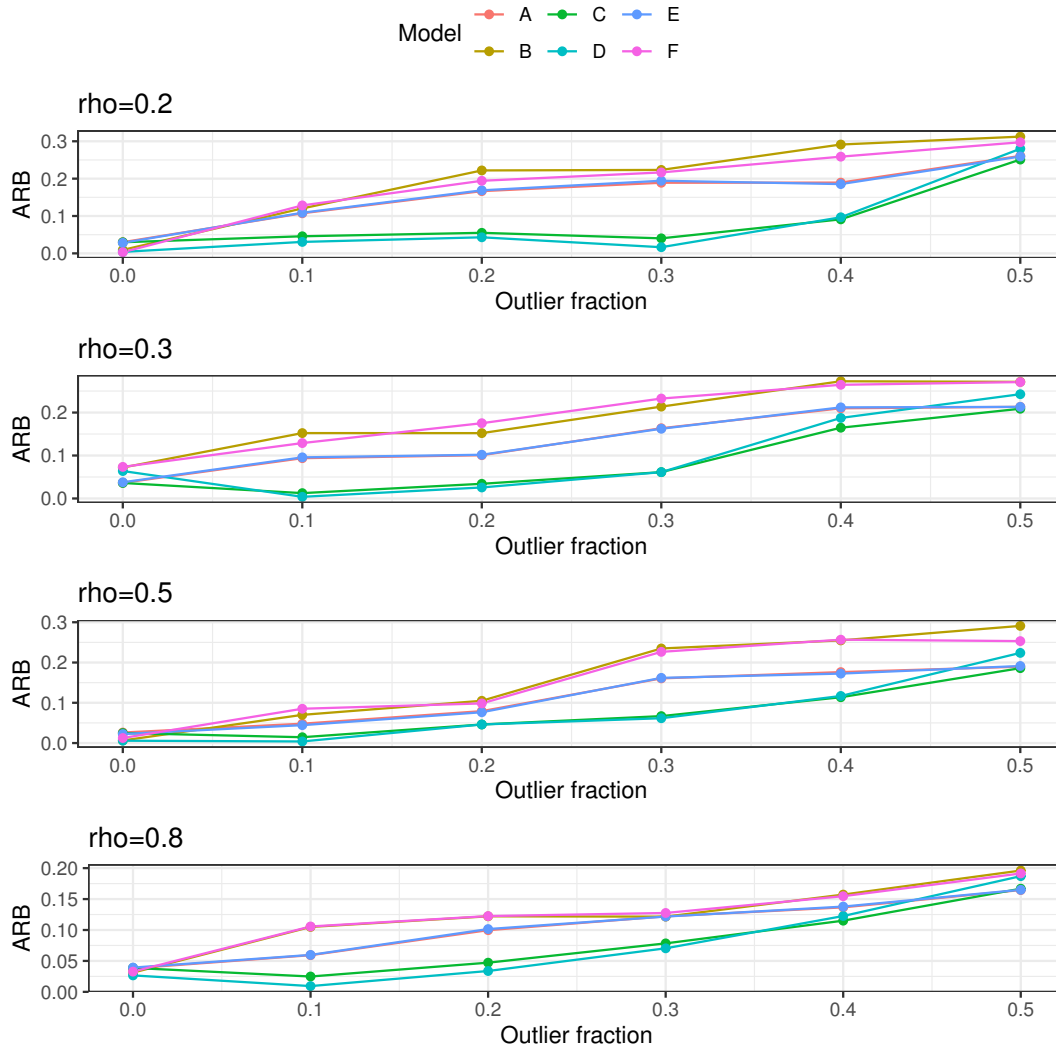


Figure 3: ARB Comparison of six models using simulation data with $\rho = 0.2, 0.3, 0.5, 0.8$ and different outlier fractions.

in California from the Third National Health and Nutrition Examination Survey (NHANES III). It contains 1,867 observations, which are large enough to create the contingency table to represent the finite population covariates if we only focus on age, race, and sex. The three covariates that are typically utilized with BMI data are discrete and include age, race, and sex. Age has about 71 levels (age runs from 20 to 90) and race and sex, each have two levels, so there are 284 distinct vectors $\tilde{x}$ of covariates.

For this example on eight counties in California, we use web scraping and the 1990 census report of California to obtain the totals of each covariate. From the Census, we can find the population size N , the average age, and race and sex proportions of the population, which are $N = 4,035,862$, average age = 36.7, race proportion = .719, sex proportion = .497. Then, we can make comparisons between the results of these various models.

First of all, the incorporation of jittered points in the boxplot may give the impression that some data points lie below the 0 line, however, it should be noted that all points are in

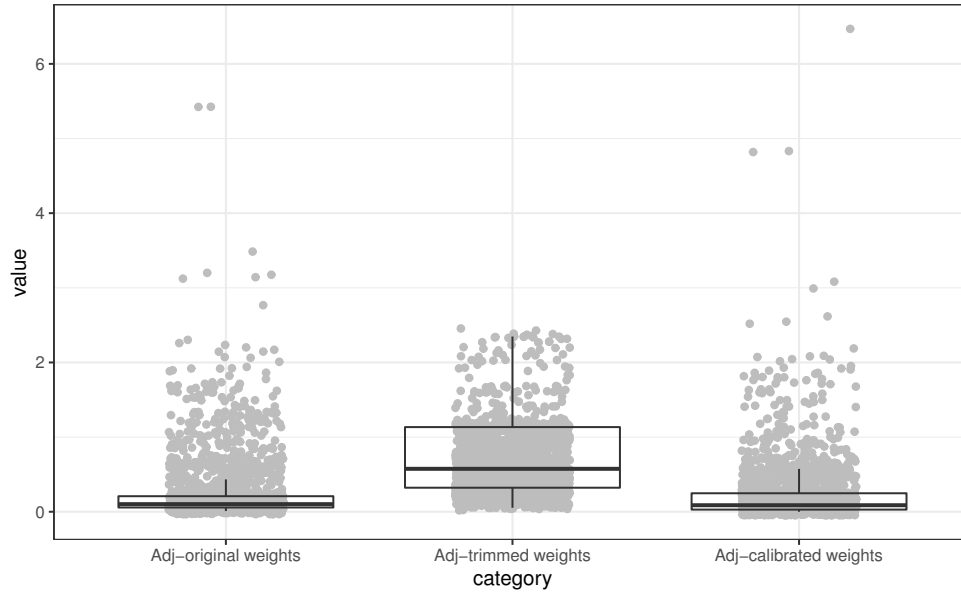


Figure 4: Comparison boxplot with jittered points of three different weights using BMI data

fact positive. In Figure 4, the boxplots of adjusted original weights and adjusted calibrated weights have a heavy tail, which means there may be some outliers of weights above the maximum boundary and can affect the estimates a lot. Their corresponding effective sample size are $n_e = 498$, $n_e^* = 1301$, $\tilde{n}_e = 498$.

As we know, outliers can have a significant impact on the effective sample size. For example, if a dataset with a large number of outliers is used to estimate the mean of a population, the outliers can pull the mean away from the true value, resulting in a larger variance and a smaller effective sample size. In this case, the effective sample size would be lower than the actual sample size, indicating that there is less useful information in the data for making inferences about the population. For the BMI dataset, the large amount of outliers decreases the two effective sample sizes derived by adjusted original weights and adjusted calibrated weights.

The distributions of PMs in Figure 5 show that these normalized models with adjusted original weights (model B) and adjusted calibrated weights (model F) are to be more to the right than other models. As a result, it is possible to challenge the accuracy of their estimates. To thoroughly investigate this issue, additional analysis of the table is required; see Appendix A for convergence tests.

In Table 3, we present posterior summaries of these six models of the finite population proportion of obesity. These six models are compared using four metrics, posterior mean (PM), posterior standard deviation (PSD), posterior coefficient of variation (PCV) and 95% credible interval. Based on the table, we can observe that for adjusted original weights and adjusted calibrated weights, the models with unnormalized densities (models A and E) offer more robustness against weights with outliers and provide more stable estimates of the population proportion in the presence of extreme weights than the models using normalized densities (model B and F). The proportion of obese individuals should be a little more around

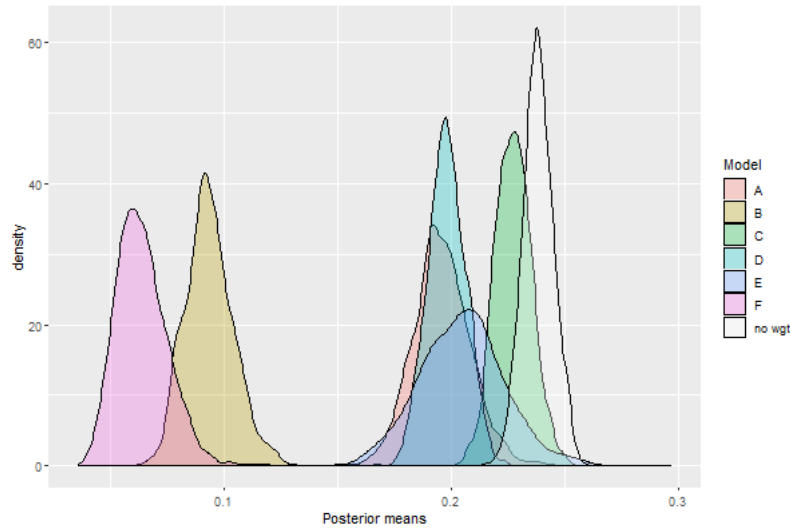


Figure 5: Comparison of densities of posterior means (PM) using BMI data

Table 3: Comparison of six models using BMI data

Model	PM	PSD	PCV	95% CI
no wgt	0.238	0.007	0.028	(0.226, 0.251)
A	0.196	0.017	0.089	(0.160, 0.232)
B	0.095	0.016	0.172	(0.065, 0.127)
C	0.226	0.012	0.055	(0.202, 0.251)
D	0.198	0.012	0.059	(0.175, 0.222)
E	0.195	0.018	0.092	(0.162, 0.233)
F	0.094	0.017	0.178	(0.063, 0.128)

0.2. If we exclude those weights, the posterior density of the population proportion shifts towards the right, resulting in a higher estimated population proportion.

An examination of responses associated with extreme weights reveals that most of them are 0's, which decreases the PMs of Models B and F to 0 in Table 3 and Figure 5. However, when we implement weight trimming, an approach designed to mitigate the influence of these weights with outliers, Models C and D have the higher PMs.

In addition, Models C and D exhibit the smallest PSD (0.012), implying the least amount of uncertainty in their estimates. This is a critical feature in statistical analysis, as reduced uncertainty typically translates to more reliable results. The low PCVs for Models C and D (0.055 and 0.059, respectively) further support their robustness, suggesting the lowest relative variability in their estimates. Contrastingly, when adjusted original weights and adjusted calibrated weights are used (as in Models A, B, E, and F), the resultant interval is wider than that of Models C and D. This interval being wider could potentially introduce a greater degree of uncertainty into the analysis; see Table 3 and Figure 5.

Figure 6 shows the posterior distributions of β . It is clear to state that when the sample size is small, the posterior densities of β have a large variance. But when we used the normalized model in the small sample size cases, the distributions shrink a little bit. For

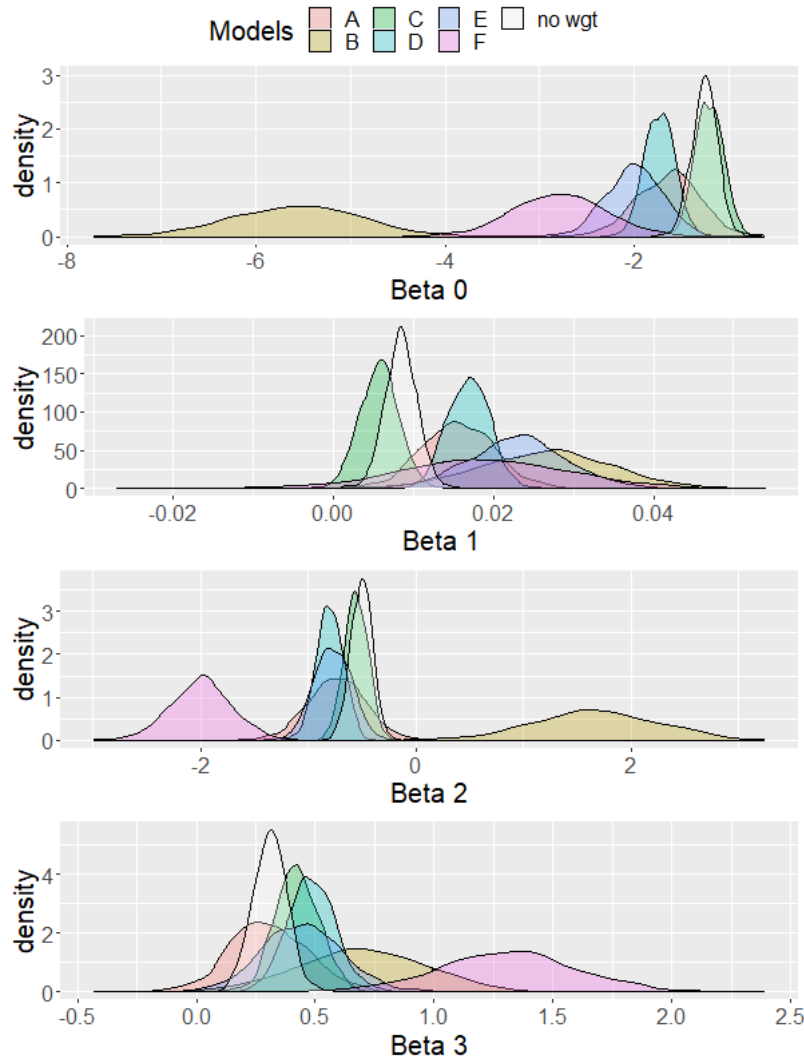


Figure 6: Comparison of posterior densities of β using BMI data

the large effective sample size, models with the adjusted trimmed survey weights, there is no significant difference between the unnormalized model and the normalized model.

5. Concluding remarks

Integrating the relevant auxiliary data, our study presents a framework to conduct Bayesian predictive inference on binary responses using probability survey samples. We introduce three distinct adjusted survey weights: the adjusted original survey weights, the adjusted trimmed survey weights, and the adjusted calibrated survey weights. These weights are integrated into both unnormalized and normalized densities within the Bayesian framework. Subsequently, the logistic regression model is implemented utilizing the Metropolis-Hastings sampler, and the predictive inference of finite population interests is made by surrogate sampling and the stratification approach.

The simulation and BMI dataset demonstrate the advantages of models incorporating adjusted trimmed weights into normalized density functions, particularly when the

correlation coefficient is small. These models not only excel in estimation accuracy but also adhere to the principles of the Bayesian paradigm, ensuring proper posterior distributions. In scenarios with exceedingly large population sizes, the normalized posterior that employs adjusted trimmed weights showcases increased robustness against weights with outliers, ultimately resulting in more precise estimations of the parameters of interest. Thus, such models should be carefully considered for their potential benefits when dealing with small correlation coefficients and large population sizes.

There will be many future works on the Bayesian predictive inference with covariates and survey weights. The proposed framework can be further expanded and applied to other types of response variables, such as categorical or continuous responses. It can also be adapted to handle more complex survey data, such as nonresponse and nonprobability sampling. One possible extension of this weighted logistic regression model is the Bayesian nonparametric priors based on the stick-breaking process, to increase model's robustness. The stick-breaking construction provides a rich framework for capturing complex patterns in data, making it a powerful tool in Bayesian nonparametric. Ren *et al.* (2011) introduced a logistic stick-breaking process as a non-parametric clustering method for general spatially- or temporally-dependent data, and the underlying assumption is that closely located data points are more likely to be grouped together. In survey sampling, the clusters often arises in complex survey sampling methods, especially those involving stratification and clustering.

Acknowledgments

I am indeed grateful to the Editors for their guidance and counsel. I am very grateful to the reviewer for valuable comments and suggestions of generously listing many useful references.

Conflict of interest

The authors do not have any financial or non-financial conflict of interest to declare for the research work included in this article.

APPENDIX A. Metropolis-Hastings algorithm for β

The Metropolis-Hastings algorithm is a Markov chain Monte Carlo (MCMC) method for generating samples from a target probability distribution that is difficult to sample from directly. The algorithm is widely used in Bayesian statistics and other applications where sampling from a complex distribution is necessary. The basic idea of the Metropolis-Hastings algorithm is to simulate a Markov chain whose stationary distribution is the target distribution of interest. The algorithm proceeds in a series of iterations, where at each iteration, a candidate state is proposed and accepted or rejected based on a acceptance probability. The acceptance probability is designed to ensure that the chain converges to the target distribution, even if the initial distribution is far from it.

In this paper, we use the Metropolis-Hastings algorithm to obtain a sample of β , with $\pi(\beta | y)$ in (16) as the target density. Then, we need to set a candidate generating (proposal) density, denoted by $q(\beta)$. The choice of proposal distribution can have a significant impact on the performance of the algorithm. Ideally, the proposal distribution should have high

probability density in regions where the target distribution has high probability density, to minimize the number of rejected proposals. However, it should also have some probability density in regions where the target distribution has low probability density, to allow the chain to explore the entire space.

It is reasonable to consider a normal approximation distribution when the target density is uni-modal. Specifically, by using the mode of our target density as the mean, $\hat{\beta}$, and the negative of the inverse-Hessian matrix at the mode as the covariance matrix, $\hat{\Sigma}$, we can obtain the normal distribution. Therefore,

$$\beta \mid y \stackrel{app}{\sim} \text{Normal} \left\{ \hat{\beta}, \sigma^2 \hat{\Sigma} \right\},$$

$$\frac{\gamma}{\sigma^2} \sim \chi_{\gamma}^2,$$

where γ is the degrees of freedom of the chi-square distribution. We also tune the degrees of freedom to ensure the jumping rate is between 25% – 75%.

The algorithm works as follows,

1. Start with an initial value, β_0 .
2. At each iteration t :
 - (a) Generate a new state, β_t , from $q(\beta)$.
 - (b) Calculate the acceptance probability, $\alpha(\beta_t, \beta_{t-1}) = \min\{1, \frac{\pi(\beta_{t-1})q(\beta_t)}{\pi(\beta_t)q(\beta_{t-1})}\}$, where β_{t-1} is the current state.
 - (c) Generate a uniform random variable u from $[0,1]$.
 - (d) If $u \leq \alpha(\beta_t, \beta_{t-1})$, accept the proposed state and set $\beta_{t+1} = \beta_t$. Otherwise, reject the proposed state and set $\beta_{t+1} = \beta_{t-1}$.
3. Repeat step 2 until convergence.

For the simulation and BMI example, the degrees of freedom are set equal to 8, and we get jumping rates, (0.468, 0.460, 0.461, 0.462, 0.469, 0.468), for all models in BMI. Additionally, we set the iteration times to 15,000 with burn-in at 5,000 and take every tenth value to have the posterior samples of parameters. Then, we use trace plots, auto-correlations, the Geweke test of stationary, and the effective sample sizes to check their convergence. In the model without survey weights, the acceptance rate is 0.466, and the Geweke test yields p-values of (0.482, 0.253, 0.289, 0.446) for each parameter, all with corresponding effective sample sizes of 1,000.

Table 4 presents the results of a Geweke test and the Effective Sample Size (ESS) for different parameters (β) under six different models in the BMI application. In the Geweke test, a higher p-value suggests that the chain has converged well. In our table, all p-values are greater than 0.05, apart from β_4 of Model E (unnormalized density with adjusted calibrated weights). Effective Sample Size (ESS) is another diagnostic measure used in MCMC analysis.

Table 4: P-values and effective sample size for models A-F in BMI

	Model A		Model B	
	P-value	ESS	P-value	ESS
Beta 1	0.623	1000	0.620	1000
Beta 2	0.383	1000	0.492	1000
Beta 3	0.435	1000	0.908	1000
Beta 4	0.592	1000	0.926	1000
	Model C		Model D	
	P-value	ESS	P-value	ESS
Beta 1	0.296	1079	0.124	1000
Beta 2	0.078	902	0.421	1164
Beta 3	0.543	762	0.662	1000
Beta 4	0.302	1099	0.606	1000
	Model E		Model F	
	P-value	ESS	P-value	ESS
Beta 1	0.454	1096	0.447	1000
Beta 2	0.706	1000	0.537	1000
Beta 3	0.848	1000	0.069	1000
Beta 4	0.028	1000	0.907	1000

In our case, the ESS for all parameters of each model is very close to or exceeds 1000 (the length of $\beta_i, i = 1, \dots, 4$), except β_3 of Model C (unnormalized density with adjusted trimmed weights), and a few more runs can do, but will virtually not change the predictive inference.

According to all these mentioned measurements, we can conclude that all models across all parameters have good convergence of the MCMC chains and the results obtained from these models can be considered reliable.

APPENDIX B. Illustration of impact of normalization constant

Consider the following models for y ,

$$f(y | \theta) = \theta e^{-\theta y}, \quad \theta, y > 0.$$

Let $W > 0$ be a fixed real number. Then,

$$(f(y | \theta))^W = \theta^W e^{-\theta W y}, \quad y > 0,$$

which is not a density function of y . In this way, the density function of y is,

$$\frac{(f(y|\theta))^W}{\int (f(y|\theta))^W dy} = \frac{\theta^W e^{-\theta W y}}{\int \theta^W e^{-\theta W y} dy}, \quad y > 0.$$

If we assume the prior of θ , $\pi(\theta) = 1, \theta > 0$, we have two forms of the posterior

distribution of θ ,

$$\begin{aligned}\pi(\theta | y) &\propto \theta^W e^{-\theta W y}, & i.e. \theta | y &\sim \text{Gamma}(W + 1, W y), \\ \pi(\theta | y) &\propto \theta W e^{-\theta W y}, & i.e. \theta | y &\sim \text{Gamma}(2, W y).\end{aligned}$$

Obviously,

$$\begin{aligned}E_1(\theta | y) &= \frac{W + 1}{W y} = (W + 1)a, & E_2(\theta | y) &= \frac{2}{W y} = 2a, \\ \text{Var}_1(\theta | y) &= \frac{W + 1}{(W y)^2} = (W + 1)a^2, & E_2(\theta | y) &= \frac{2}{(W y)^2} = 2a^2,\end{aligned}$$

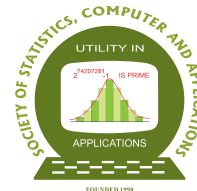
where $a = 1/(W + 1)$ is an irrelevant value.

If $W = 1$, there is no difference between the posterior distribution of unnormalized and normalized density. If $W < 1$, $E_1(\theta|y) < E_2(\theta|y)$ and $\text{Var}_1(\theta|y) < \text{Var}_2(\theta|y)$. If $W > 1$, the opposite situation holds. Now, it is clear when the normalization constant is related to the parameter, how it will affect our estimation.

References

- Archer, K. J. and Lemeshow, S. (2006). Goodness-of-fit test for a logistic regression model fitted using survey sample data. *The Stata Journal*, **6**, 97–105.
- Barasa, K. S. and Muchwanju, C. (2015). Incorporating survey weights into binary and multinomial logistic regression models. *Science Journal Applied Mathematics and Statistics*, **3**, 243–249.
- Basu, D. (2010). An essay on the logical foundations of survey sampling, part one. In DasGupta, A., editor, *Selected Works of Debabrata Basu*, pages 167–206. Springer, New York, NY.
- Chen, L. and Nandram, B. (2023). Bayesian logistic regression model for sub-areas. *Stats*, **6**, 209–231.
- Chen, M.-H., Ibrahim, J. G., and Kim, S. (2008). Properties and implementation of Jeffreys’s prior in binomial regression models. *Journal of the American Statistical Association*, **103**, 1659–1664.
- Chen, Q., Elliott, M. R., Haziza, D., Yang, Y., Ghosh, M., Little, R. J., Sedransk, J., and Thompson, M. (2017). Approaches to improving survey-weighted estimates. *Statistical Science*, **32**, 227–248.
- Chen, Y., Li, P., and Wu, C. (2020). Doubly robust inference with nonprobability survey samples. *Journal of the American Statistical Association*, **115**, 2011–2021.
- Deville, J.-C., Särndal, C.-E., and Sautory, O. (1993). Generalized raking procedures in survey sampling. *Journal of the American statistical Association*, **88**, 1013–1020.
- Elliott, M. R. and Valliant, R. (2017). Inference for nonprobability samples. *Statistical Science*, **32**, 249–264.
- Gelman, A. (2007). Struggles with survey weighting and regression modeling. *Statistical Science*, **22**, 153–164.
- Haziza, D. and Beaumont, J.-F. (2017). Construction of weights in surveys: A review. *Statistical Science*, **32**, 206–226.

- Lohr, S. (2007). Comment: Struggles with survey weighting and regression modeling. *Statistical Science*, **22**, 175–178.
- Lohr, S. (2009). *Sampling: Design and Analysis*. Nelson Education.
- Nandram, B. (2007). Bayesian predictive inference under informative sampling via surrogate samples. *Bayesian Statistics and its Applications*, **1**, 356–374.
- Nandram, B. and Choi, J. W. (2010). A bayesian analysis of body mass index data from small domains under nonignorable nonresponse and selection. *Journal of the American Statistical Association*, **105**, 120–135.
- Nandram, B., Choi, J. W., and Liu, Y. (2021). Integration of nonprobability and probability samples via survey weights. *International Journal of Statistics and Probability*, **10**, 1–5.
- Nandram, B. and Rao, J. (2021). A Bayesian approach for integrating a small probability sample with a non-probability sample. In *JSM Proceedings, Survey Research Methods Section, Alexandria, VA: American Statistical Association*, **1**, 1568–1603.
- Nandram, B. and Rao, J. (2024). Bayesian integration for small areas by supplementing a probability sample with a non-probability sample. *Statistics and Applications {ISSN 2454-7395 (online)}*, **22**, 343–374.
- Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, **108**, 1339–1349.
- Potter, F. J. (1990). A study of procedures to identify and trim extreme sampling weights. In *Proceedings of the American Statistical Association, Section on Survey Research Methods*, volume 225230. American Statistical Association Washington, DC.
- Potthoff, R. F., Woodbury, M. A., and Manton, K. G. (1992). “Equivalent sample size” and “equivalent degrees of freedom” refinements for inference using survey weights under superpopulation models. *Journal of the American Statistical Association*, **87**, 383–396.
- Rader, K. A., Lipsitz, S. R., Fitzmaurice, G. M., Harrington, D. P., Parzen, M., and Sinha, D. (2017). Bias-corrected estimates for logistic regression models for complex surveys with application to the united states’ nationwide inpatient sample. *Statistical Methods in Medical Research*, **26**, 2257–2269.
- Rao, J. (1966). Alternative estimators in pps sampling for multiple characteristics. *Sankhyā: The Indian Journal of Statistics, Series A*, **28**, 47–60.
- Ren, L., Du, L., Carin, L., and Dunson, D. B. (2011). Logistic stick-breaking process. *Journal of Machine Learning Research*, **12**.
- Robbins, M. W., Ghosh-Dastidar, B., and Ramchand, R. (2021). Blending probability and nonprobability samples with applications to a survey of military caregivers. *Journal of Survey Statistics and Methodology*, **9**, 1114–1145.
- Roberts, G., Rao, N., and Kumar, S. (1987). Logistic regression analysis of sample survey data. *Biometrika*, **74**, 1–12.
- Srivastava, S. K. (1967). An estimator using auxiliary information in sample serveys. *Calcutta Statistical Association Bulletin*, **16**, 121–132.
- Yang, L., Nandram, B., and Choi, J. W. (2023). Bayesian predictive inference under nine methods for incorporating survey weights. *International Journal of Statistics and Probability*, **12**, 33–53.



From Classical Probability to Quantum Amplitudes: A Pedagogical Introduction for Statisticians

Shyam Dhamapurkar¹ and Sourish Das²

¹*Computer Science, Chennai Mathematical Institute*

²*Mathematics, Chennai Mathematical Institute*

Received: 9 May 2026; Revised: 10 July 2026; Accepted: 13 August 2026

Abstract

Quantum computing offers provable speedups for fundamental statistical tasks: estimating means, solving linear systems, and sampling from posterior distributions. This article provides a self-contained, pedagogical bridge from classical probability and statistics to their quantum analogues. We begin with Kolmogorov's axioms and build up through conditional probability, Bayes' rule, random variables, expectation, sampling distributions, the central limit theorem, maximum likelihood estimation, and Bayesian inference. For each concept, we present a quantum counterpart using only linear algebra. A single running example, *estimating the bias of a coin*, illustrates both classical and quantum approaches throughout. We give detailed, step-by-step explanations of Grover's search algorithm Grover (1996), Quantum Amplitude Estimation Brassard *et al.* (2002), the HHL algorithm Harrow *et al.* (2009), and Quantum Markov Chain Monte Carlo Szegedy (2004). Throughout, we state the precise regime of validity of each complexity bound and distinguish query complexity from total gate complexity; edge cases and technical caveats are kept brief and clearly marked so they inform without interrupting the main narrative. The article is self-contained and requires no prior knowledge of quantum mechanics. For deeper algorithmic detail, readers may consult the comprehensive review by Lopatnikova *et al.* (2024) or the standard textbook by Nielsen and Chuang (2010).

Key words: Quantum statistics; Quantum states; Quantum coin toss; Exponential efficiency; Binomial distribution as quantum state; Quantum expectation and variance.

AMS Subject Classifications: 62K05, 05B05

1. Introduction

Classical statistics and classical computing grew up together, like twins who learned to walk side by side. Markov chain Monte Carlo Geyer (1992), the bootstrap, the EM algo-

rithm, and gradient descent were all designed for classical computers; machines that operate according to the rules of classical physics. These machines can copy data arbitrarily (try copying a file on your computer), reset bits to zero (delete a file), and execute deterministic logic (if this, then that). We have become so accustomed to these capabilities that we rarely think about them.

Why should a statistician care about quantum computing?

Quantum computers operate under a different set of physical laws. They leverage three properties that have no classical analogues:

- **Superposition:** A quantum system can exist in multiple states simultaneously, like a coin spinning in the air before it lands.
- **Entanglement:** Quantum systems can be correlated in ways that cannot be reproduced by any classical system, even with hidden variables.
- **Interference:** Quantum amplitudes (the “square roots” of probabilities) can add constructively or destructively, allowing us to amplify desired outcomes and suppress undesired ones.

These properties are not just mathematical curiosities. They enable quantum computers to solve certain problems fundamentally faster than any classical computer ever could. For statisticians, this is exciting news because many of our core tasks, estimating means, solving linear systems, sampling from posterior distributions, are exactly the kinds of problems where quantum computers offer provable speedups. But there is a catch. Quantum computers do not think like classical computers. We cannot simply take existing `R` or `Python` code and run it on a quantum computer. We must rethink how we represent data, how we compute expectations, and how we perform inference. This article is designed to help build that new mental model, one classical concept at a time.

What this article is, and what it is not

This article is a self-contained, pedagogical introduction to quantum computing for people who already know probability and statistics. It provides a conceptual bridge that maps classical statistical concepts, probability distributions, random variables, expectation, Bayes’ rule, to their quantum analogues, and presents a step-by-step guide to the most important quantum algorithms for statisticians: Grover’s search Grover (1996), Quantum Amplitude Estimation Brassard *et al.* (2002), the HHL algorithm Harrow *et al.* (2009), and Quantum Markov Chain Monte Carlo Szegedy (2004). It is, throughout, a practical discussion of when quantum advantage is plausible and what bottlenecks remain, we are as interested in where these algorithms fall short as in where they succeed.

Comprehensive surveys of quantum algorithms for statistics already exist, most notably the recent review by Lopatnikova *et al.* (2024), which catalogues a wide range of algorithms and provides detailed complexity analyses. However, such surveys typically assume a degree of comfort with quantum notation, Dirac bra-ket formalism, or computational

complexity theory that a classically trained statistician may not yet have. This article is designed as a *translation guide*: every quantum concept is introduced *only after* its classical statistical counterpart has been established, and the two are compared side by side. The goal is not to replace Lopatnikova *et al.* (2024) or Nielsen and Chuang (2010), but to equip the reader with the vocabulary and intuition needed to tackle those heavier texts with confidence.

This article is not a comprehensive survey of all quantum algorithms (Lopatnikova *et al.* (2024) serves that purpose). It is not a guide to programming quantum computers, though we provide references to Qiskit and other SDKs at the end. And it is not a physics textbook: we use only linear algebra, no physics beyond the basic postulates of measurement and unitary evolution.

The running example: a biased coin

The problem: We have a coin that may be unfair. It comes up Heads with probability θ and Tails with probability $1 - \theta$. We do not know θ . We flip the coin n times and observe k heads. What can we say about θ ?

Throughout this article, we will return again and again to this single, simple problem: estimating the bias of a coin. This problem is deceptively simple. It is the “Hello, World” of statistics (Casella and Berger, 2002). It appears in every introductory textbook. And yet, it illustrates every major concept we will need: probability axioms, conditional probability and Bayes’ rule, random variables, expectation and variance, sampling distributions and the CLT, MLE, and Bayesian inference. By the end of this article we will know the **quantum** analogue of each of these concepts. We will understand how a quantum computer can estimate θ with quadratically fewer oracle queries than a classical computer needs samples. We will see how Grover’s search (Grover, 1996) can find the maximum of a posterior distribution faster than classical search. And we will appreciate the practical challenges, data loading and readout, that currently limit how much of this advantage can be realised on actual hardware.

If you understand how the classical and quantum approaches differ for this one simple problem, you will understand how they differ for any problem. Now, let us build the classical foundation.

Prerequisites and notation

The only mathematical prerequisites for this article are threefold. First, **basic probability**: the reader should know what a probability space is, what conditional probability means, and what expectation does; Chapters 1–3 of Casella and Berger (2002) are more than sufficient background. Second, **linear algebra**: the reader should be comfortable with vectors, matrices, inner products, eigenvalues, and eigenvectors. We do not need advanced topics like singular value decomposition, though we will mention it in passing; for quantum computing specifically, the standard reference is Nielsen and Chuang (2010). Third, **basic calculus**: one should know how to take derivatives and integrals, which we will need for maximum likelihood estimation and for locating the posterior mode.

Equally important is what we do *not* need. We do not assume any physics knowledge, no Schrödinger equation, no Hamiltonian mechanics, no quantum field theory. We do not assume any prior exposure to quantum computing whatsoever. And we do not assume any programming experience, though we do mention programming platforms briefly near the end of the article for readers who want to experiment hands-on.

We use standard statistical notation throughout ($\mathbb{E}[X]$, $\text{Var}(X)$, $\text{Cov}(X, Y)$, $N(\mu, \sigma^2)$, $\text{Binomial}(n, \theta)$, $\text{Beta}(\alpha, \beta)$), alongside Dirac’s bra-ket notation, which we introduce here as a quick reference before we begin using it in earnest:

$ \psi\rangle$	A column vector (pronounced “ket psi”)
$\langle\psi $	The conjugate transpose of $ \psi\rangle$ (“bra psi”)
$\langle\phi \psi\rangle$	The inner product of $ \phi\rangle$ and $ \psi\rangle$
$\langle\psi O \psi\rangle$	The expectation value $\langle\psi O \psi\rangle$, <i>i.e.</i> , $\langle\psi O \psi\rangle$
$ \psi\rangle\langle\psi $	The outer product $ \psi\rangle\langle\psi $ (a projection operator)
$ 0\rangle, 1\rangle$	Standard basis vectors: $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

The key thing to remember is that $|\psi\rangle$ is just a vector; the notation is a convention, not new mathematics. Note that $\langle\phi|\psi\rangle$ always has exactly two slots (an inner product), while $\langle\psi|O|\psi\rangle$ is a different shape, an operator sandwiched between a bra and a ket; we use the two notations consistently throughout so they are never conflated.

A word of encouragement. Richard Feynman famously said, “If you think you understand quantum mechanics, you don’t understand quantum mechanics.” Do not be discouraged if some concepts feel counterintuitive at first; the mathematics is linear algebra you already know, and the intuition comes with practice.

How to read this article

This article is designed to be read sequentially: each section builds on the previous ones, and the running coin example is meant to be tracked continuously rather than treated as a series of disconnected illustrations. That said, readers already comfortable with classical probability and statistics may skim Sections 2–4, which establish the classical foundations most statisticians already know well, and focus their attention on the quantum analogues developed in Sections 5 through 9, which is where the genuinely new material lives. Readers who want the algorithmic payoff first and the careful justification afterward might also choose to read the four algorithm sections, Sections 6, 7, 8, and 9, somewhat out of order, since each is largely self-contained once Sections 2 through 5 have been absorbed; we have tried to make the cross-references explicit enough that this is a safe way to read the article.

2. Classical probability axioms and quantum states

Why do we need axioms?

Before we write down the axioms, let us ask a more fundamental question: why do we need axioms for probability at all? The answer is that probability is a mathematical theory,

and like all mathematical theories (Euclidean geometry, group theory, real analysis), it needs a set of foundational assumptions from which all other results can be derived. The axioms ensure that probability is self-consistent and that different people using the same rules will reach the same conclusions. The modern axiomatic foundation of probability was laid by the Russian mathematician Andrey Kolmogorov in 1933. Before Kolmogorov, probability was a collection of useful but loosely connected techniques. After Kolmogorov, probability became a rigorous branch of measure theory, on par with real analysis or topology (Billingsley, 1995).

We begin with the definition of a probability space. This is, as we will see shortly, the classical analogue of what will later become a Hilbert space in the quantum setting, so it is worth paying close attention to the structure.

Definition 1 (Probability space): A probability space is a triple $(\Omega, \mathcal{F}, P)$ where Ω is the sample space (all possible outcomes), $\mathcal{F}$ is a σ -algebra of measurable events, and $P : \mathcal{F} \rightarrow [0, 1]$ is a probability measure.

The sample space Ω

The sample space is simply the set of all possible outcomes of an experiment. It can be finite, countably infinite, or uncountably infinite, and the three cases behave quite differently.

Example 1 (Finite sample space): For a single coin flip, $\Omega = \{H, T\}$. There are exactly two possible outcomes.

Example 2 (Countably infinite sample space): For flipping a coin until the first head appears, $\Omega = \{H, TH, TTH, TTTH, \dots\}$. There are infinitely many outcomes, but we can list them in a sequence.

Example 3 (Uncountable sample space): For measuring the exact waiting time until a radioactive decay, $\Omega = [0, \infty)$. There are uncountably many possible outcomes.

The σ -algebra $\mathcal{F}$

Not every subset of Ω can be assigned a probability. This is a technical subtlety that arises specifically for uncountable sample spaces; it is related to the Banach–Tarski paradox and the existence of non-measurable sets. The σ -algebra $\mathcal{F}$ is the collection of subsets to which we *can* assign probabilities, and it must satisfy three properties: (1) $\Omega \in \mathcal{F}$ (the whole space is an event); (2) if $A \in \mathcal{F}$ then $A^c \in \mathcal{F}$ (complements are events); (3) if $A_1, A_2, \dots \in \mathcal{F}$ then $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$ (countable unions are events). For discrete sample spaces we usually take $\mathcal{F}$ to be the power set of Ω (all subsets); for continuous sample spaces we take the Borel σ -algebra, generated by open intervals.

Definition 2 (Kolmogorov’s axioms): The probability measure P satisfies: **(1) Non-negativity:** $P(A) \geq 0$. **(2) Normalization:** $P(\Omega) = 1$. **(3) Countable additivity:** for disjoint $A_1, A_2, \dots \in \mathcal{F}$, $P(\bigcup_i A_i) = \sum_i P(A_i)$.

From these three axioms alone we can derive finite additivity, the complement rule

$P(A^c) = 1 - P(A)$, monotonicity, and inclusion-exclusion.

Example 4 (Coin flip): $\Omega = \{H, T\}$, $\mathcal{F}$ = all subsets, $P(\{H\}) = \theta$, $P(\{T\}) = 1 - \theta$. Non-negativity holds since $\theta, 1 - \theta \geq 0$; normalization holds since $\theta + (1 - \theta) = 1$; countable additivity reduces to finite additivity here.

From classical probability to quantum mechanics: a parallel structure

Now that we have the classical foundation, let us observe a striking parallel. In classical probability we have: Sample space $\Omega \xrightarrow{\text{events}}$ Probability measure $P \xrightarrow{\text{outcomes}}$ Random variable X . In quantum mechanics, the structure is similar but with complex numbers replacing real numbers and vectors replacing sets: Hilbert space $\mathcal{H} \xrightarrow{\text{subspaces}}$ State $|\psi\rangle \xrightarrow{\text{measurement}}$ Observable O .

This parallel is not accidental. Quantum mechanics is, in a deep sense, a non-commutative generalisation of probability theory. Where classical probabilities are real numbers between 0 and 1 that add, quantum “probabilities” are squared magnitudes of complex amplitudes that, for orthogonal events, also add; the distinctive new feature is that for *non-orthogonal* decompositions the amplitudes can cancel before squaring. For a thorough introduction, see Nielsen and Chuang (2010).

The quantum analogue of a probability space is a Hilbert space; the definition below lists three properties an inner product must satisfy, each of which is worth pausing on since they recur constantly in everything that follows.

Definition 3 (Hilbert space): A Hilbert space $\mathcal{H}$ is a vector space over the complex numbers $\mathbb{C}$ equipped with an inner product $\langle \cdot, \cdot \rangle$ that is: (1) *linear in the second argument*, $\langle \phi, a\psi_1 + b\psi_2 \rangle = a\langle \phi, \psi_1 \rangle + b\langle \phi, \psi_2 \rangle$; (2) *conjugate symmetric*, $\langle \phi, \psi \rangle = \overline{\langle \psi, \phi \rangle}$; and (3) *positive definite*, $\langle \psi, \psi \rangle \geq 0$ with equality only if $\psi = 0$. The space is complete with respect to the induced norm $\|\psi\| = \sqrt{\langle \psi, \psi \rangle}$. For our purposes, finite-dimensional Hilbert spaces $\mathcal{H} \cong \mathbb{C}^d$ are entirely sufficient, we never need the infinite-dimensional machinery that a full treatment of quantum mechanics eventually requires.

Remark 1 (Why complex numbers?): This is a natural question to ask. Why do quantum states use complex amplitudes rather than real numbers? The answer is that complex numbers are necessary to capture interference phenomena. With only real amplitudes, the only possible interference is constructive (same sign) or destructive (opposite sign), but complex phases allow for far more subtle interference patterns than a real-valued theory could produce. This is not merely a mathematical convenience adopted for elegance; experiments confirm that quantum mechanics genuinely requires complex amplitudes to match observed reality (Nielsen and Chuang, 2010).

The fundamental object in quantum mechanics is the quantum state itself.

Definition 4 (Pure quantum state): A pure quantum state is a unit vector $|\psi\rangle$ in a Hilbert space $\mathcal{H}$. In an orthonormal basis $\{|i\rangle\}_{i=0}^{N-1}$, we can write $|\psi\rangle = \sum_i \psi_i |i\rangle$ with $\sum_i |\psi_i|^2 = 1$; the complex numbers ψ_i are called **amplitudes**.

The notation $|\psi\rangle$ (pronounced “ket psi”) is, as a column vector, $|\psi\rangle = (\psi_0, \dots, \psi_{N-1})^\top$; the corresponding row vector is $\langle\psi| = (\psi_0^*, \dots, \psi_{N-1}^*)$, where $*$ denotes complex conjugation, and the inner product of two states is $\langle\phi|\psi\rangle = \sum_i \phi_i^* \psi_i$.

The key difference: amplitudes vs. probabilities

Classical probability distributions are vectors of non-negative real numbers that sum to 1: $p = (p_0, \dots, p_{N-1})$, $p_i \geq 0$, $\sum_i p_i = 1$. Quantum states are vectors of complex numbers whose squared magnitudes sum to 1: $\psi = (\psi_0, \dots, \psi_{N-1})$, $\psi_i \in \mathbb{C}$, $\sum_i |\psi_i|^2 = 1$. The difference looks small on the page but is profound in its consequences. Because amplitudes can be negative or complex, they can **interfere**: two paths to the same outcome can have amplitudes with opposite signs and cancel, reducing, or even eliminating, the probability of that outcome. This is flatly impossible in classical probability, where probabilities are non-negative numbers that can only add constructively; there is no classical mechanism by which two positive probabilities can combine to give something smaller than either one.

Example 5 (Interference): Consider the quantum state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The probability of measuring 0 is $|\frac{1}{\sqrt{2}}|^2 = 1/2$. Now consider a transformation that sends $|1\rangle$ to $-|1\rangle$; the new state is $|\psi'\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. If we measure immediately in the original basis, $P(0)$ is still $1/2$, the individual probabilities have not changed at all, since the sign flip on $|1\rangle$ is invisible to a measurement that only looks at $|0\rangle$ versus $|1\rangle$. But if we first apply this transformation and *then* measure in a different basis, the interference becomes visible, and the resulting probabilities can differ substantially from what either branch alone would predict. This is the principle behind Grover’s search algorithm (Section 6) and, in one form or another, behind every quantum speedup discussed in this article.

The Born rule: from amplitudes to probabilities

The Born rule connects the quantum state (amplitudes) to observable outcomes (probabilities). It is named after Max Born, who proposed it in 1926 and later received the Nobel Prize for this work.

Definition 5 (Born rule): A measurement of $|\psi\rangle$ in basis $\{|i\rangle\}$ yields outcome i with probability $P(i) = |\langle i|\psi\rangle|^2 = |\psi_i|^2$; the state then **collapses** to $|i\rangle$.

The collapse is a distinctive feature of quantum mechanics. Before measurement, the system is in a superposition of many possibilities. At the moment of measurement, it “chooses” one outcome, and the state becomes that outcome. This is not an approximation or a lack of knowledge, it is a fundamental feature of the theory. (The mechanism of collapse is still debated philosophically, but the mathematical rule itself is not.) In classical probability, if you have a random variable X with distribution $p(x)$, and you observe $X = x_0$, you update your beliefs to “ X is definitely x_0 ”, this is similar to collapse. However, there is no classical analogue of superposition: a classical system cannot be simultaneously in two distinct states with coherent amplitudes the way a quantum system can.

It is important to be precise about what the Born rule does and does not preserve

from Kolmogorov's axioms. For a *fixed* basis $\{|i\rangle\}$, $P(i) = |\psi_i|^2$ satisfies countable additivity exactly as required: for disjoint $A, B \subseteq \{i\}$,

$$P(A \cup B) = \sum_{i \in A \cup B} |\psi_i|^2 = \sum_{i \in A} |\psi_i|^2 + \sum_{i \in B} |\psi_i|^2 = P(A) + P(B),$$

with *no* interference term, because A and B correspond to orthogonal projectors in the same basis. Interference enters only when amplitudes from a *non-orthogonal* decomposition are summed before squaring: if $|\phi\rangle = \alpha|u\rangle + \beta|v\rangle$ for non-orthogonal $|u\rangle, |v\rangle$, the cross terms $\alpha^*\beta\langle u|v\rangle + \alpha\beta^*\langle v|u\rangle$ generally do not vanish. This is the genuine source of interference and of every speedup in this article, a statement about non-orthogonal decompositions, not a blanket failure of additivity for the Born rule itself.

Mixed states and density matrices

Not all quantum states we will need to work with are pure. Sometimes we have classical uncertainty about which pure state a system is actually in, for example, we might prepare a qubit in state $|0\rangle$ with probability $1/2$ and in state $|1\rangle$ with probability $1/2$, perhaps because of an imperfect preparation procedure. This kind of situation, where there is a genuine classical probability distribution over which pure state we have, is called a **mixed state**, and it requires a more general mathematical object than a single ket to describe.

Definition 6 (Mixed state): A mixed state is represented by a density matrix $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$, where $p_i \geq 0$ and $\sum_i p_i = 1$. The density matrix satisfies three properties: it is Hermitian ($\rho = \rho^\dagger$), positive semidefinite (all eigenvalues non-negative), and has unit trace ($\text{tr}(\rho) = 1$). For a pure state $|\psi\rangle$, the density matrix is simply $\rho = |\psi\rangle\langle\psi|$, the special case where the mixture collapses onto a single term.

The Born rule generalizes naturally to mixed states, but doing so requires introducing a slightly more general notion of “measurement” than the simple basis measurements we used above. Measurements are described most generally by a collection of **Kraus operators** $\{M_m\}$ satisfying $\sum_m M_m^\dagger M_m = I$; the associated **Positive Operator-Valued Measure (POVM)** is the set of operators $E_m := M_m^\dagger M_m$, and the probability of outcome m is $P(m) = \text{tr}(E_m \rho) = \text{tr}(M_m^\dagger M_m \rho)$. We will use the Kraus operators M_m themselves when we need to track the post-measurement state, as in Section 3, and the POVM elements E_m when we only care about outcome probabilities; the two notations describe exactly the same physical measurement and are related by $E_m = M_m^\dagger M_m$. For the special case of **projective measurements**, where $M_m = |m\rangle\langle m|$ for some orthonormal basis $\{|m\rangle\}$ (so that $E_m = M_m$ as well), this entire apparatus reduces to the familiar $P(m) = \langle m|\rho|m\rangle$ we would expect from the Born rule.

Example 6 (The quantum coin): A biased coin can be represented two ways. As a *mixed* state,

$$\rho_{\text{mix}} = \theta|0\rangle\langle 0| + (1 - \theta)|1\rangle\langle 1| = \begin{pmatrix} \theta & 0 \\ 0 & 1 - \theta \end{pmatrix},$$

or as a *pure* state $|\psi_\theta\rangle = \sqrt{\theta}|0\rangle + \sqrt{1-\theta}|1\rangle$, with density matrix

$$\rho_{\text{pure}} = |\psi_\theta\rangle\langle\psi_\theta| = \begin{pmatrix} \theta & \sqrt{\theta(1-\theta)} \\ \sqrt{\theta(1-\theta)} & 1-\theta \end{pmatrix}.$$

Both are valid density matrices (Hermitian, positive semidefinite, unit trace); what distinguishes them is the off-diagonal *coherence* term $\sqrt{\theta(1-\theta)}$, present only in ρ_{pure} . The mixed state represents *epistemic* uncertainty, classical ignorance about which of $|0\rangle, |1\rangle$ we are in, exactly like a classical probability mixture, producing no interference. The pure state represents *ontological* uncertainty: a genuine coherent superposition, not secretly one definite state, and it is this coherence that gives rise to interference effects.¹

We use $|\psi_\theta\rangle$ as our standard quantum coin throughout the rest of this article. Applying the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$:

$$H|\psi_\theta\rangle = \frac{\sqrt{\theta} + \sqrt{1-\theta}}{\sqrt{2}}|0\rangle + \frac{\sqrt{\theta} - \sqrt{1-\theta}}{\sqrt{2}}|1\rangle,$$

so $P(0) = \frac{1}{2}(\theta + (1-\theta) + 2\sqrt{\theta(1-\theta)}) = \frac{1}{2}(1 + 2\sqrt{\theta(1-\theta)})$ and $P(1) = \frac{1}{2}(1 - 2\sqrt{\theta(1-\theta)})$. These still sum to 1, but each now contains the cross-term $2\sqrt{\theta(1-\theta)}$: a genuine quantum interference effect, arising because H mixes the amplitudes of $|0\rangle$ and $|1\rangle$ *before* the Born rule squares them, with no analogue if we could only manipulate the classical probabilities $\theta, 1-\theta$ directly. The effect is visible only because we measure *after* applying H ; measured immediately in the original basis, no such cross-term appears.

Summary: classical vs. quantum probability

Before moving on to conditional probability and Bayes' rule, it is worth pausing to consolidate everything we have built in this section into a single reference table, since we will return to and extend this dictionary repeatedly throughout the rest of the article.

The key takeaway: quantum mechanics is a **non-commutative probability theory**. Within any fixed measurement basis, the Born rule is additive exactly as Kolmogorov's axioms require; the new ingredient is that amplitudes belonging to a coherent superposition can be combined, via a unitary transformation, *before* the Born rule is applied, and the resulting cross-terms are the engine of every quantum speedup discussed below.

3. Conditional probability and Bayes' rule

Conditional probability is the mathematical machinery of updating beliefs in light of new information, and Bayes' rule is its most important consequence, arguably the single

¹Classical mixture models (*e.g.* Gaussian mixtures) are the statistical analogue of the *mixed* state; the *pure* state has no classical statistical analogue. Measuring either ρ_{mix} or ρ_{pure} directly in the $\{|0\rangle, |1\rangle\}$ basis gives $P(0) = \theta, P(1) = 1-\theta$ with no interference, since $\{0\}$ and $\{1\}$ are orthogonal events; the coherence only becomes visible after a unitary mixes $|0\rangle$ and $|1\rangle$ before measurement, as below.

Table 1: Classical probability and its quantum analogue

Concept	Classical	Quantum
Space	Sample space Ω	Hilbert space $\mathcal{H}$
Basic object	Distribution $p(\omega)$	State vector $ \psi\rangle$
Normalization	$\sum_{\omega} p(\omega) = 1$	$\ \psi\rangle \ ^2 = 1$
Combination rule	$p(\omega) \geq 0$, add	Add within a basis; may cancel across non-orthogonal states
Measurement outcome	$\omega \sim p(\omega)$	$i \sim \psi_i ^2$
Update after measurement	Condition on observed ω	Collapse to $ i\rangle$
Uncertainty about state	Mixture of distributions	Density matrix ρ

most useful formula in all of applied statistics. We state both here in their familiar classical form before turning to their quantum counterparts.

Definition 7 (Conditional probability): For $P(B) > 0$, $P(A | B) = P(A \cap B)/P(B)$ (Casella and Berger, 2002).

Theorem 1 (Bayes' rule):

$$P(A | B) = \frac{P(B | A)P(A)}{\sum_i P(B | A_i)P(A_i)}.$$

Theorem 2 (Law of total probability): For a partition $\{A_i\}$, $P(B) = \sum_i P(B | A_i)P(A_i)$.

Example 7 (Coin inference): Suppose our prior places equal weight on two candidate biases: $P(\theta = 0.5) = 0.5$ and $P(\theta = 0.7) = 0.5$. After observing $k = 7$ heads in $n = 10$ flips, Bayes' rule gives the posterior weight on $\theta = 0.7$ as proportional to the likelihood times the prior: $P(K=7 | \theta=0.7) \cdot 0.5 = \binom{10}{7}(0.7)^7(0.3)^3 \cdot 0.5 \approx 0.2668 \times 0.5 = 0.1334$, while the competing hypothesis contributes $P(K=7 | \theta=0.5) \cdot 0.5 = \binom{10}{7}(0.5)^{10} \cdot 0.5 \approx 0.1172 \times 0.5 = 0.0586$. Normalising, the posterior probability that $\theta = 0.7$ is $0.1334/(0.1334 + 0.0586) \approx 0.695$ (Gelman *et al.*, 2013), a substantial update from the prior 50% toward the hypothesis better supported by the data, exactly as intuition would suggest.

Quantum conditional probability

In quantum mechanics, conditioning on a measurement outcome updates the state in a manner directly analogous to classical conditioning, using the Kraus-operator notation M_m we introduced in Section 2.

Definition 8 (Quantum conditional state): For Kraus operators $\{M_m\}$ with $\sum_m M_m^\dagger M_m = I$, if outcome m occurs with probability $p_m = \text{tr}(M_m^\dagger M_m \rho)$, the post-measurement state is $\rho_m = M_m \rho M_m^\dagger / p_m$. For projective measurements ($M_m = |m\rangle\langle m|$), this is $\rho_m = |m\rangle\langle m|$, collapse onto the measured basis vector.

A natural question is whether there is a quantum analogue of Bayes' rule updating a prior *state* into a posterior *state* given quantum “data.” Unlike the classical case, there is no single canonical quantum Bayes' rule; several inequivalent formalizations exist depending on what plays the role of the likelihood (Wiebe *et al.*, 2012). We present one widely used construction, flagged explicitly as one choice among several.

Definition 9 (A quantum Bayes' update): Given prior ρ and likelihood (Kraus) operators $\{L_x\}$ with $\sum_x L_x^\dagger L_x = I$, one natural posterior, obtained by applying the corresponding quantum instrument and renormalising, is

$$\rho_{\text{post}} = \frac{\sum_x L_x \rho L_x^\dagger}{\text{tr}(\sum_x L_x \rho L_x^\dagger)}.$$

This aggregates over all possible outcomes x rather than conditioning on one observed x ; conditioning on a single realised x_0 instead uses the single-Kraus-operator rule $\rho_{x_0} = L_{x_0} \rho L_{x_0}^\dagger / \text{tr}(L_{x_0} \rho L_{x_0}^\dagger)$, a special case of the quantum conditional state above. In the classical-like (diagonal, commuting) case, the amplitude version of this update reduces, heuristically, to $\sqrt{p(\theta | y)} \propto \sqrt{p(y | \theta)} \sqrt{p(\theta)}$, a useful mnemonic, not a theorem, since making it precise requires specifying exactly how θ and y are encoded as quantum registers. For a detailed treatment of the several competing notions, see Wiebe *et al.* (2012).

Theorem 3 (Quantum law of total probability): For a POVM $\{E_m\}$ and $\rho = \sum_i p_i \rho_i$, $P(m) = \text{tr}(E_m \rho) = \sum_i p_i \text{tr}(E_m \rho_i)$.

4. Random variables and probability distributions

Now that we have established the foundations of classical probability and their quantum analogues, we turn to one of the most important concepts in statistics: the random variable. A random variable is a way of turning the abstract outcomes of an experiment into numbers that we can analyse, instead of talking about “Heads” and “Tails,” we assign the number 1 to Heads and 0 to Tails, which lets us compute averages, variances, and other summaries. In quantum mechanics, the analogue is an **observable**: just as a random variable assigns a number to each outcome in the sample space, an observable assigns a number, an eigenvalue, to each possible measurement outcome.

Classical random variables

Intuitively, a random variable is simply a numerical summary of an experiment. Before we flip a coin, we do not know whether it will land Heads or Tails, but we know in advance that if it is Heads we will record 1, and if it is Tails we will record 0; the random variable is precisely this rule for converting outcomes into numbers.

Definition 10 (Random variable): A random variable $X : \Omega \rightarrow \mathbb{R}$ is a measurable function from the sample space Ω to the real numbers. The cumulative distribution function (CDF) of X is $F_X(x) = P(X \leq x) = P(\{\omega \in \Omega : X(\omega) \leq x\})$.

The term “measurable” here is a technical condition ensuring that the set $\{\omega : X(\omega) \leq$

$x\}$ is itself an event in $\mathcal{F}$, so that we are actually allowed to assign it a probability. For all practical purposes in statistics, every function one would ever write down is measurable; the condition only becomes relevant for the same pathological, non-measurable sets that motivated the σ -algebra machinery in Section 2, and essentially never arises in applied work.

Example 8 (Coin flip indicator): Define $X(\text{Heads}) = 1$ and $X(\text{Tails}) = 0$. Then $F_X(x) = 0$ for $x < 0$; $F_X(x) = 1 - \theta$ for $0 \leq x < 1$; and $F_X(x) = 1$ for $x \geq 1$.

Definition 11 (PMF and PDF): For discrete X , the PMF is $p_X(x) = P(X = x) \geq 0$, $\sum_x p_X(x) = 1$. For continuous X , the PDF f_X satisfies $P(a \leq X \leq b) = \int_a^b f_X(x) dx$, $f_X(x) \geq 0$, $\int_{-\infty}^{\infty} f_X(x) dx = 1$ (Casella and Berger, 2002).

Definition 12 (Binomial distribution): For independent Bernoulli(θ) trials $X_1, \dots, X_n$, $K = \sum_i X_i$ has $P(K = k) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}$, $k = 0, \dots, n$. This is the natural distribution governing exactly our running example: the total number of heads in n independent flips of a coin with bias θ .

Example 9 (Coin flips): For $n = 10, \theta = 0.7$: $P(K = 7) = \binom{10}{7} (0.7)^7 (0.3)^3 = 120 \times 0.0823543 \times 0.027 \approx 0.267$, the same likelihood we used numerically in the Bayes' rule example of Section 3.

Quantum observables

Now we come to the quantum analogue. Just as a classical random variable assigns a real number to each outcome in the sample space, a quantum observable assigns a real number, an eigenvalue, to each possible measurement outcome. Recall from linear algebra that a Hermitian matrix $A = A^\dagger$, where $\dagger$ denotes conjugate transpose, has three important properties: all of its eigenvalues are real; eigenvectors corresponding to distinct eigenvalues are automatically orthogonal; and the matrix can always be diagonalised by a unitary transformation. The spectral theorem generalises these familiar linear-algebra facts to operators on general Hilbert spaces (Nielsen and Chuang, 2010).

Definition 13 (Observable): A Hermitian operator $O = O^\dagger$ on a Hilbert space $\mathcal{H}$ is called an **observable**. Its spectral decomposition is $O = \sum_o o |o\rangle \langle o|$, where $\{o\}$ are the distinct eigenvalues, the possible measurement outcomes, and $\{|o\rangle\}$ form an orthonormal basis of eigenvectors. This is the precise quantum analogue of writing a classical random variable as $X(\omega) = \sum_x x \cdot \mathbf{1}_{\{X=x\}}(\omega)$, where $\mathbf{1}_A$ is the indicator function of event A : both expressions decompose a numerical summary into a weighted sum over its possible values, weighted by an indicator (classically) or a projector (quantumly) onto the event that summary takes that value.

When we measure the observable O on a quantum state ρ , the Born rule tells us the probability of obtaining eigenvalue o is $P(O = o) = \text{tr}(|o\rangle \langle o| \rho) = \langle o | \rho | o \rangle$; for a pure state $|\psi\rangle$, this simplifies to the familiar $P(O = o) = |\langle o | \psi \rangle|^2$.

Table 2: Random variables and their quantum analogue, observables

Classical	Quantum
Sample space Ω	Hilbert space $\mathcal{H}$
Outcome $\omega \in \Omega$	Basis state $ i\rangle \in \mathcal{H}$
Random variable $X : \Omega \rightarrow \mathbb{R}$	Observable $O = \sum_o o o\rangle\langle o $
$P(X = x) = \sum_{\omega: X(\omega)=x} P(\omega)$	$P(O = o) = \text{tr}(o\rangle\langle o \rho)$
$\mathbb{E}[X] = \sum_x xP(X = x)$	$\langle O \rangle = \sum_o oP(O = o) = \text{tr}(O\rho)$

Quantum sample states

One of the most powerful ideas in quantum computing for statistics is the **quantum sample state**: a single quantum state that encodes an entire probability distribution in its amplitudes, rather than in a classical array of numbers.

Definition 14 (Quantum sample state): For a distribution $p(x)$ on a finite set $\mathcal{X} = \{x_1, \dots, x_N\}$, the quantum sample state is $|P\rangle = \sum_{x \in \mathcal{X}} \sqrt{p(x)}|x\rangle$, where $\{|x\rangle\}$ is an orthonormal basis labeled by the elements of $\mathcal{X}$.

Why is this called a “sample state”? Because measuring $|P\rangle$ in the $\{|x\rangle\}$ basis gives outcome x with probability $|\langle x|P\rangle|^2 = |\sqrt{p(x)}|^2 = p(x)$, exactly the original distribution. The state therefore functions as a “factory” for producing i.i.d. samples from $p(x)$: each measurement draws one sample, and repeated measurements (on fresh copies of the state, since measurement is destructive) draw i.i.d. samples exactly as classical Monte Carlo would.

Exponential efficiency in storage. To store a distribution over $N = 2^n$ values classically requires N numbers, about 8 MB for $N = 10^6$ double-precision floats. As a quantum sample state, only $n = \log_2 N$ qubits are needed: 20 qubits for $N = 1,000,000$, an exponential reduction. But this is a storage statement, not a preparation-time statement: writing an arbitrary N -dimensional distribution into amplitudes generically requires $O(N)$ gates, so the storage saving does not automatically yield a preparation-time saving unless the distribution has special structure. Grover and Rudolph (2002) showed distributions with an efficiently integrable CDF (*e.g.* the normal, or our binomial example below) can be prepared in $O(\text{polylog } N)$ operations; but Herbert (2021) showed this construction is essentially limited to distributions simple enough to not need Monte Carlo in the first place, raising real doubt about whether efficient preparation is available for the complex empirical distributions statisticians actually meet. Reading out an already-prepared state (tomography) is a separate cost, discussed in Section 10.

Definition 15 (Amplitude encoding): For $x = (x_0, \dots, x_{N-1})^\top \in \mathbb{C}^N$ with $\|x\| = 1$, amplitude encoding represents it as $|x\rangle = \sum_i x_i|i\rangle$. Unlike a quantum sample state, the amplitudes here are arbitrary normalized complex numbers, not square roots of a target distribution; amplitude encoding underlies linear-algebra algorithms such as HHL (Harrow *et al.*, 2009), and loading an arbitrary vector this way costs $O(N)$ operations in general, the same generic cost as above.

Example 10 (Binomial quantum state): $|\text{Bin}_{n,\theta}\rangle = \sum_{k=0}^n \sqrt{\binom{n}{k} \theta^k (1-\theta)^{n-k}} |k\rangle$ needs only $\lceil \log_2(n+1) \rceil$ qubits to *store*, 7 qubits for $n = 100$, versus 101 classical probabilities. (Because the binomial amplitudes follow an efficiently integrable CDF, Grover–Rudolph can in fact prepare this particular state efficiently; a generic distribution over the same 101 outcomes would not enjoy this guarantee.) Given many copies of $|\text{Bin}_{n,\theta}\rangle$ for unknown θ , estimating θ is the quantum version of our coin problem, the setting for Quantum Amplitude Estimation, Section 6 (Brassard *et al.*, 2002; Montanaro, 2015).

5. Expectation and variance

Now that we understand random variables (classically) and observables (quantum mechanically), we turn to two of the most important summaries of any distribution: the expectation, which tells us the average value we should expect, and the variance, which tells us how spread out the distribution is around that average.

Definition 16 (Expectation): For a discrete random variable X with probability mass function $p(x)$, $\mathbb{E}[X] = \sum_x x p(x)$. For a continuous random variable X with probability density function $f(x)$, $\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x) dx$ (Casella and Berger, 2002). The expectation is linear: for any constants a, b and random variables X, Y , $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$.

Definition 17 (Variance): The variance of a random variable X is $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$; the standard deviation is $\sigma = \sqrt{\text{Var}(X)}$.

Example 11 (Coin flip): For a single coin flip, define $X = 1$ for Heads and $X = 0$ for Tails. Then $\mathbb{E}[X] = 1 \cdot \theta + 0 \cdot (1 - \theta) = \theta$ and $\mathbb{E}[X^2] = 1^2 \cdot \theta + 0^2 \cdot (1 - \theta) = \theta$, so $\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \theta - \theta^2 = \theta(1 - \theta)$. For a fair coin ($\theta = 0.5$), the variance is 0.25; for a completely biased coin ($\theta = 0$ or $\theta = 1$), the variance is 0, exactly as we would expect since there is then no uncertainty left in the outcome at all.

The quantum expectation generalizes the classical expectation to observables in a direct and natural way.

Definition 18 (Quantum expectation and variance): For a quantum state ρ (density matrix) and an observable $O = \sum_o o |o\rangle\langle o|$, the expectation value is $\langle O \rangle = \text{tr}(O\rho) = \sum_o o P(O=o)$, where $P(O=o) = \text{tr}(|o\rangle\langle o|\rho)$ is the probability of obtaining outcome o . For a pure state $|\psi\rangle$, we have $\rho = |\psi\rangle\langle\psi|$ and $\text{tr}(O|\psi\rangle\langle\psi|) = \langle\psi|O|\psi\rangle = \langle\psi|O|\psi\rangle$, so the two notations for the expectation value used throughout this article, trace form and bra-ket form, coincide exactly whenever ρ is pure, and we use whichever is more convenient in context. The quantum expectation satisfies the same linearity property as the classical expectation: $\langle aO_1 + bO_2 \rangle = a\langle O_1 \rangle + b\langle O_2 \rangle$. The variance of an observable O in state ρ is $\text{Var}(O) = \langle O^2 \rangle - \langle O \rangle^2 = \text{tr}(O^2\rho) - (\text{tr}(O\rho))^2$.

Example 12 (Quantum coin expectation): For the quantum coin state $|\psi_\theta\rangle = \sqrt{\theta}|0\rangle + \sqrt{1-\theta}|1\rangle$ and the observable $Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ (the Pauli Z matrix), we have $\langle Z \rangle = \langle\psi_\theta|Z|\psi_\theta\rangle = |\sqrt{\theta}|^2 \cdot 1 + |\sqrt{1-\theta}|^2 \cdot (-1) = \theta - (1-\theta) = 2\theta - 1$, a linear transform of θ . More directly, if we define the observable $O = |0\rangle\langle 0|$, which measures simply whether the outcome is Heads, then $\langle O \rangle = \langle\psi_\theta|O|\psi_\theta\rangle = \theta$, matching the classical expectation exactly, as

it should: the quantum and classical machinery must agree when restricted to this simple case.

Estimating expectations is perhaps the single most fundamental computational task in statistics. Almost everything we do as practitioners, computing sample means, running regression, performing hypothesis tests, evaluating risk in finance, calculating treatment effects in medicine, reduces, at bottom, to estimating an expectation. Classically, the workhorse method for this task is **Monte Carlo** (Metropolis and Ulam, 1949): draw N i.i.d. samples from a distribution and average the function of interest. The standard error of this estimate scales as $O(1/\sqrt{N})$, which means halving the error requires four times as many samples, a frustratingly slow rate of improvement that statisticians have lived with for decades. **Quantum Amplitude Estimation** (Brassard *et al.*, 2002; Montanaro, 2015) achieves a quadratic speedup *in the number of oracle queries* over this classical rate, under the assumption of coherent access to a state-preparation oracle (we make the precise caveats around this assumption explicit in Section 10). To understand how QAE achieves this, we first need to understand Grover’s search algorithm and its generalisation, Quantum Amplitude Amplification.

6. Quantum search and amplitude estimation

Before we can understand the quadratic speedup for estimating expectations promised at the end of the last section, we need to understand Grover’s search algorithm, since it is the conceptual foundation that both Quantum Amplitude Amplification and Quantum Amplitude Estimation are built on top of. We will work through Grover’s algorithm in full geometric detail, then show how it generalises, and finally apply the generalised version back to our running coin-bias problem.

The problem: unstructured search

Imagine you have a database with N items. One of these items is marked, it satisfies some condition, and you want to find it. You have a “black box” oracle that can tell you whether a given item is marked, but the oracle gives you no other information; this is called **unstructured search**. Classically, the best you can do is check items one by one. In the worst case you might check all N items before finding the marked one; on average you will check about $N/2$. So the classical query complexity is $O(N)$. Grover’s algorithm (Grover, 1996) solves this problem using only $O(\sqrt{N})$ queries, a quadratic speedup, and one that has been proven optimal: no quantum algorithm can do better for fully unstructured search.

Grover’s algorithm, step by step

- **Step 1: Represent the database as a quantum state.** For $N = 2^n$, label each item by an n -bit string $x \in \{0, 1\}^n$. We represent the entire database as a quantum superposition $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$, the **uniform superposition**, in which every basis state $|x\rangle$ carries amplitude $1/\sqrt{N}$. Intuitively, this single quantum state “contains” every item in the database simultaneously, with no item yet favoured over any other.

- **Step 2: The oracle.** We are given access to an oracle U_f that recognises the marked item x_0 and acts as $U_f|x\rangle = -|x\rangle$ if $x = x_0$, and $U_f|x\rangle = |x\rangle$ otherwise; equivalently $U_f = I - 2|x_0\rangle\langle x_0|$. This operator flips the sign of the amplitude on the marked state and leaves every other amplitude untouched, a purely local action that, on its own, does nothing observable, since the Born rule only sees squared magnitudes.
- **Step 3: Reflection about the mean.** Grover's algorithm pairs the oracle with a second operator, the **diffusion operator** $U_s = 2|s\rangle\langle s| - I$ for $|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$. To see what this does, consider any state $|\phi\rangle = \sum_x \alpha_x |x\rangle$ with mean amplitude $\bar{\alpha} = \frac{1}{N} \sum_x \alpha_x$; applying U_s sends $|\phi\rangle$ to $\sum_x (2\bar{\alpha} - \alpha_x) |x\rangle$, so amplitudes above the mean shrink and amplitudes below the mean grow, literally a reflection of each amplitude about the average amplitude.
- **Step 4: The Grover iteration.** Combining these two operators, $G = U_s U_f$, turns out to act as a rotation in the two-dimensional plane spanned by the marked state $|x_0\rangle$ and the uniform superposition of all the unmarked states, $|s'\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle$. This geometric picture, a single rotation in a two-dimensional subspace, repeated many times, is the key to understanding why the algorithm works at all.
- **Step 5: Geometric interpretation.** Writing $|s\rangle = \sin \theta_G |x_0\rangle + \cos \theta_G |s'\rangle$ with $\sin \theta_G = 1/\sqrt{N}$ (we use θ_G , not θ , to avoid clashing with the coin bias used throughout this article), each application of G rotates by a further angle $2\theta_G$: $G|s\rangle = \sin(3\theta_G)|x_0\rangle + \cos(3\theta_G)|s'\rangle$, and in general after k iterations, $G^k|s\rangle = \sin((2k+1)\theta_G)|x_0\rangle + \cos((2k+1)\theta_G)|s'\rangle$. Each iteration moves the state a little closer to $|x_0\rangle$, but, crucially, if we keep iterating past the optimal point, the rotation overshoots and the state begins moving *away* from $|x_0\rangle$ again.
- **Step 6: Measurement.** Measuring after k iterations succeeds with probability $P(\text{success}) = \sin^2((2k+1)\theta_G)$, which is maximised exactly when $(2k+1)\theta_G$ is as close as possible to $\pi/2$. The exact optimal iteration count is therefore $k^* = \lfloor \pi/(4\theta_G) \rfloor = \lfloor \pi/(4 \arcsin(1/\sqrt{N})) \rfloor$. Since $\arcsin(1/\sqrt{N}) = 1/\sqrt{N} + O(N^{-3/2})$ for large N , this gives the familiar asymptotic estimate $k^* \approx (\pi/4)\sqrt{N}$, an approximation, not an exact equality, since $\theta_G \neq 1/\sqrt{N}$ exactly for any finite N . This distinction matters in practice: stopping a few iterations early or late, especially for small N , can noticeably hurt the success probability.

Example 13 (Searching coin biases): Suppose we discretise the coin bias into $N = 1,000,000$ candidate values and an oracle can recognise the “correct” one. Classical search checks up to 10^6 possibilities in the worst case. Grover's algorithm needs only $\sqrt{10^6} = 1000$ queries asymptotically; more precisely, the exact optimal count is $k^* = \lfloor \pi/(4 \arcsin(10^{-3})) \rfloor \approx 785$, already matching the asymptotic order $(\pi/4)\sqrt{N} \approx 785$ closely even at this only moderately large N .

Quantum Amplitude Amplification (QAA)

Grover's algorithm is the special case of QAA (Brassard *et al.*, 2002) where the initial state is the uniform superposition; in general we start from $|\psi\rangle = \mathcal{A}|0\rangle = a_1|\psi_1\rangle + a_0|\psi_0\rangle$ prepared by a unitary $\mathcal{A}$, with $|\psi_1\rangle$ the “good” subspace, $a_1 = \langle \psi_1 | \psi \rangle$ its amplitude, and $|a_1|^2$ the probability of a good outcome.

Definition 19 (QAA): Since $\mathcal{A}$ is unitary, $\mathcal{A}^{-1} = \mathcal{A}^\dagger$ (both notations appear in the literature; we use $\mathcal{A}^{-1}$ to emphasise that it “undoes” state preparation). Define $Q = -\mathcal{A}U_0\mathcal{A}^{-1}U_f$, where U_f flips the sign of the good subspace and U_0 flips the sign of $|0\rangle$. After $t = O(1/|a_1|)$ applications of Q , the good-subspace amplitude becomes $O(1)$ (Brassard *et al.*, 2002).

The structure of QAA mirrors Grover step for step, which is worth making explicit since it is what lets the same geometric intuition carry over: $\mathcal{A}$ prepares the initial state, replacing the Hadamard layer $H^{\otimes n}$; U_f marks good states, replacing the oracle; and $\mathcal{A}U_0\mathcal{A}^{-1} = \mathcal{A}U_0\mathcal{A}^\dagger$ reflects about the initial state, replacing reflection about the mean. Everything we learned about Grover’s rotation in a two-dimensional subspace applies again here, just with the uniform superposition $|s\rangle$ replaced by the more general prepared state $|\psi\rangle$.

Definition 20 (Quantum Amplitude Estimation): Q has eigenvalues $e^{\pm 2i\phi}$ on the 2D subspace spanned by $|\psi_1\rangle, |\psi_0\rangle$, where $\sin^2 \phi = |a_1|^2$. Quantum Phase Estimation (Section 8) on Q estimates ϕ to precision $O(\epsilon)$ using $O(1/\epsilon)$ queries; plugging $\hat{\phi}$ into $|\widehat{a_1}|^2 = \sin^2 \hat{\phi}$ then estimates $|a_1|^2$ to precision $O(\epsilon)$ as well, *provided* $|a_1|^2$ is bounded away from 0 and 1.²

We now return to the general problem of estimating an expectation $\mu = \mathbb{E}[f(X)] = \sum_x p(x)f(x)$ for some function f with $0 \leq f(x) \leq 1$, and state the classical and quantum bounds precisely before working through the coin example step by step.

Theorem 4 (Classical Monte Carlo error bound): Let $\hat{\mu} = \frac{1}{N} \sum_{i=1}^N f(X_i)$ be the sample mean from N i.i.d. draws of $X \sim p(x)$. Then $\mathbb{E}[\hat{\mu}] = \mu$ and $\text{Var}(\hat{\mu}) = \sigma^2/N$ where $\sigma^2 = \text{Var}(f(X))$, so $N = O(\sigma^2/\epsilon^2)$ draws suffice to achieve error ϵ with constant probability (Metropolis and Ulam, 1949).

Theorem 5 (Quantum Monte Carlo via QAE (Brassard *et al.*, 2002; Montanaro, 2015)): QAE estimates μ to precision ϵ with high probability using $O(\sigma/\epsilon)$ queries to f and the state-preparation oracle, away from the boundary regime $\mu \approx 0$ or $\mu \approx 1$, a **quadratic speedup** over classical Monte Carlo in query complexity.

Step-by-step QAE for the coin example

To estimate θ itself, rather than the sample proportion of some fixed batch of flips, the oracle below must encode the Bernoulli probability θ directly, rather than n individually realised 0/1 outcomes. This distinction is easy to gloss over but is actually the crux of why QAE behaves so differently from classical Monte Carlo, so we spell out the construction carefully.

- **Step 1: State preparation.** We build the state $\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_n |0\rangle_m |0\rangle_1$. The first register, of n qubits, is an *index* register, it is not recording flip outcomes the way a

²Near these endpoints, $\sin^2$ is locally flat (derivative vanishes), so the same angular precision ϵ translates into a smaller error in $|a_1|^2$; near $\phi = \pi/4$ the map is best-conditioned. The headline “ $O(1/\epsilon)$ queries for precision ϵ ” should be read as holding uniformly away from the endpoints; see Brassard *et al.* (2002) for the precise non-asymptotic bound and Montanaro (2015) for the resulting Monte Carlo speedup.

classical batch of n flips would, but is instead used to drive N independent coherent “virtual flips” of the same underlying coin. The second register will encode θ ; the third is a single auxiliary qubit that will eventually carry the information we want to measure.

- **Step 2: Function encoding.** An oracle $\mathcal{A}_\theta$ prepares, completely independently of the index i , the same encoding of the bias θ in the second register: $\mathcal{A}_\theta|i\rangle_n|0\rangle_m = |i\rangle_n|\theta\rangle_m$. Every branch of the superposition therefore shares the *same* parameter θ , rather than each branch carrying its own independently realised 0/1 outcome as a classical batch of flips would. This is the key conceptual difference from classical sampling, and it is exactly what lets QAE target the parameter θ itself rather than merely a noisy sample proportion k/n .
- **Step 3: Controlled rotation.** A controlled rotation $C\text{-}R_y(2\arcsin\sqrt{\theta})$ acts on the auxiliary qubit: $|\theta\rangle_m|0\rangle_1 \mapsto |\theta\rangle_m(\sqrt{1-\theta}|0\rangle + \sqrt{\theta}|1\rangle)$, which exactly reproduces the pure quantum coin state $|\psi_\theta\rangle$ from Section 2 inside the auxiliary qubit.
- **Step 4: Uncomputation.** We apply $\mathcal{A}_\theta^{-1}$ to clear the second register back to $|0\rangle_m$, so that it no longer carries any entanglement with the rest of the system.
- **Step 5: Resulting state.** The state is now $|\psi\rangle = \left(\frac{1}{\sqrt{N}}\sum_i|i\rangle_n\right) \otimes (\sqrt{1-\theta}|0\rangle + \sqrt{\theta}|1\rangle)$, a product state, with the index register and the auxiliary qubit completely unentangled. By the Born rule applied to this unentangled auxiliary factor, $P(\text{aux}=1) = \theta$ exactly, with no dependence whatsoever on N or on any single realised classical sample. The role of N here is entirely different from its classical role: it only controls how many times the same rotation is coherently repeated across the index register within a single QAE run, which is precisely the resource that QAE’s underlying amplitude amplification machinery uses to sharpen the eventual estimate.
- **Step 6: Amplitude estimation.** Applying QAE to this construction estimates $|a_1|^2 = \theta$ to additive precision ϵ using $O(1/\epsilon)$ queries, subject to the boundary caveat discussed above when θ is close to 0 or 1.

Example 14 (Numerical comparison): For $\epsilon = 0.001$, away from the boundary: classical Monte Carlo needs $\approx 1/\epsilon^2 = 10^6$ flips; QAE needs $\approx 1/\epsilon = 1000$ oracle queries, requiring coherent execution, but provably a quadratic speedup in query complexity. The gap widens dramatically as the target precision tightens, which is worth seeing concretely rather than only asymptotically:

Table 3: The query-complexity gap between classical Monte Carlo and QAE grows linearly in $1/\epsilon$; for high-precision estimation the practical difference becomes enormous, though, as emphasised throughout, this gap is in oracle queries, not necessarily wall-clock time on real hardware

Target precision ϵ	Classical samples $O(1/\epsilon^2)$	QAE queries $O(1/\epsilon)$	Ratio
10^{-1}	100	10	$10\times$
10^{-2}	10,000	100	$100\times$
10^{-3}	1,000,000	1,000	$1,000\times$
10^{-4}	100,000,000	10,000	$10,000\times$

7. Central limit theorem, Markov chains, and quantum walks

Definition 21 (Sample mean): For i.i.d. $X_1, \dots, X_n$ with mean μ , variance σ^2 : $\bar{X}_n = \frac{1}{n} \sum_i X_i$, $\mathbb{E}[\bar{X}_n] = \mu$, $\text{Var}(\bar{X}_n) = \sigma^2/n$ (Casella and Berger, 2002).

As n grows, $\text{Var}(\bar{X}_n) \rightarrow 0$ and the distribution becomes approximately normal regardless of the underlying distribution, the Central Limit Theorem.

Theorem 6 (CLT): $\sqrt{n}(\bar{X}_n - \mu)/\sigma \xrightarrow{d} N(0, 1)$, equivalently $\sqrt{n}(\bar{X}_n - \mu) \xrightarrow{d} N(0, \sigma^2)$.

Example 15 (Coin flips and CLT): $\mu = \theta, \sigma^2 = \theta(1 - \theta)$: $\sqrt{n}(\frac{k}{n} - \theta) \approx N(0, \theta(1 - \theta))$, giving the 95% CI $\hat{\theta} \pm 1.96\sqrt{\hat{\theta}(1 - \hat{\theta})/n}$, where $\hat{\theta} = k/n$ and 1.96 is the 97.5% standard-normal quantile.

Now that we understand expectation and variance, we turn to a deeper question: if we take a sample and compute a statistic, like the sample mean, how does that statistic behave across different samples? This is the study of **sampling distributions**, and it is where the Central Limit Theorem enters.

When we measure identical quantum sample states $|P\rangle = \sum_x \sqrt{p(x)}|x\rangle$ (Section 4) repeatedly, we obtain i.i.d. samples from $p(x)$, and the classical CLT applies directly to these measurement outcomes: quantum sampling is statistically indistinguishable from classical sampling *once the state is prepared*. This is worth dwelling on for a moment, since it is easy to assume quantum mechanics changes everything, it does not change the statistics of measurement outcomes once you have them. The quantum advantage instead lies entirely in *how efficiently we can prepare* $|P\rangle$ in the first place, particularly when p is the stationary distribution of a Markov chain. This is the role of quantum walks, to which we now turn. We first review the classical theory of Markov chains, since the quantum walk construction is built directly on top of it.

Classical Markov chains

A Markov chain is a random process where the next state depends only on the current state, the so-called “Markov property”, and not on the full history of how the process arrived there. This memorylessness is what makes Markov chains tractable to analyse and is the foundation on which MCMC methods are built.

Definition 22 (Markov chain): A (discrete-time, finite-state) Markov chain on a state space $\mathcal{X}$ of size N is defined by a transition matrix $P \in \mathbb{R}^{N \times N}$ where $P(x, y) = P(X_{t+1}=y \mid X_t=x)$, satisfying $P(x, y) \geq 0$ for all x, y and $\sum_y P(x, y) = 1$ for each x (each row sums to 1, since the chain must go *somewhere*).

A Markov chain is called **reversible** with respect to a distribution π if it satisfies the *detailed balance* condition $\pi(x)P(x, y) = \pi(y)P(y, x)$ for all x, y ; in this case π is automatically a **stationary distribution**, satisfying $\pi^\top P = \pi^\top$, meaning that once the chain reaches π it stays there in distribution forever after. The **mixing time** of a Markov chain is the

number of steps required for the distribution of X_t to get close to π , regardless of the chain's starting point; this mixing time is governed by the **spectral gap** $\delta = 1 - \lambda_2$, where λ_2 is the second-largest eigenvalue of P in absolute value (the largest eigenvalue is always exactly 1, corresponding to the stationary distribution itself). The classical mixing time scales as $O(1/\delta)$: chains with a large spectral gap mix quickly, while chains with a small spectral gap, meaning the second eigenvalue is close to 1, can take a very long time to forget their starting point.

Example 16 (Two-state chain): Consider a two-state chain on $\{0, 1\}$ with transition matrix $P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$. Direct computation shows the eigenvalues of P are 1 and $1 - \alpha - \beta$, so the spectral gap is $\delta = 1 - (1 - \alpha - \beta) = \alpha + \beta$. Solving $\pi^\top P = \pi^\top$ gives the stationary distribution $\pi = (\frac{\beta}{\alpha+\beta}, \frac{\alpha}{\alpha+\beta})$, and the mixing time is $O(1/(\alpha + \beta))$: the chain mixes quickly when either transition probability is large, and slowly when both α and β are small, since the chain then has little tendency to leave whichever state it starts in.

Szegedy quantum walks

In 2004, Mario Szegedy showed how to quantise any reversible Markov chain (Szegedy, 2004). On the doubled register $\mathcal{H} = \mathbb{C}^{\mathcal{X}} \otimes \mathbb{C}^{\mathcal{Y}}$, define $U_P|x\rangle|0\rangle = |x\rangle \sum_y \sqrt{P(x, y)}|y\rangle$ (extended to a full unitary in the standard way). Let $\Pi = \sum_x |x\rangle\langle x| \otimes U_P|0\rangle\langle 0|U_P^\dagger$ be the projector onto the span of $\{U_P|x\rangle|0\rangle\}_x$, S the swap operator, and $\Pi' = S\Pi S$. The reflections are $\mathcal{R}_1 = 2\Pi - I$, $\mathcal{R}_2 = 2\Pi' - I$, and the walk operator is $W(P) = \mathcal{R}_2\mathcal{R}_1$.³

Theorem 7 (Quantum walk speedup (Szegedy, 2004)): The state $|\pi\rangle = \sum_x \sqrt{\pi(x)}|x\rangle \otimes (\sum_y \sqrt{P(x, y)}|y\rangle)$ is a $+1$ eigenstate of $W(P)$; measuring its first register recovers exactly the quantum sample state $\sum_x \sqrt{\pi(x)}|x\rangle$ of Section 4. If the classical chain has spectral gap δ , the eigenvalues of $W(P)$ are $e^{\pm i\Theta}$ with *phase gap* $\Theta = \Omega(\sqrt{\delta})$, quadratically larger than δ . Quantum phase estimation on $W(P)$ can prepare a state ϵ -close to $|\pi\rangle$ using $O(1/\sqrt{\delta})$ applications of $W(P)$, versus $O(1/\delta)$ classical steps.

Why does the quantum walk achieve a quadratic speedup at all? The classical Markov chain moves by random, memoryless steps, and each step reduces the distance to stationarity by a factor governed by δ ; convergence is a diffusive, gradual process. The quantum walk instead uses interference to explore the state space coherently. The walk operator rotates the state in a two-dimensional invariant subspace, and the relevant rotation angle scales as $\Theta = \Omega(\sqrt{\delta})$ rather than δ itself, so the number of applications of $W(P)$ needed to complete an order-one rotation, and hence to localise on $|\pi\rangle$ via phase estimation, is $O(1/\sqrt{\delta})$ rather than $O(1/\delta)$. This is exactly the same square-root relationship between rotation angle and iteration count that drove the Grover speedup in Section 6; the Szegedy walk is, in this sense, Grover's algorithm generalised from searching a fixed marked item to sampling from an entire stationary distribution.

³We use the projector notation Π, Π' here, rather than $U_P U_P^\dagger$ directly, because $U_P U_P^\dagger = I$ for a full unitary completion and would trivialise the reflection; what matters is the projector onto the specific subspace prepared by U_P conditioned on the first register.

Example 17 (Cycle): A classical random walk on an N -cycle has *hitting time* $O(N^2)$ (diffusive); a continuous-time quantum walk reaches the antipodal node in $O(N)$ steps, *ballistic* propagation (Childs *et al.*, 2003; Kempe, 2003).⁴

Quantum Markov chain Monte Carlo

Classical MCMC is among the most widely used computational tools in all of Bayesian statistics: we construct a Markov chain whose stationary distribution is our target distribution π , typically a posterior distribution that we cannot sample from directly, and then simulate the chain for many steps to obtain approximate samples from π . Quantum MCMC (QMCMC) aims to skip the simulation entirely and prepare the quantum sample state $|\pi\rangle = \sum_x \sqrt{\pi(x)}|x\rangle$ directly, using the Szegedy walk together with quantum phase estimation to project onto (a state close to) the eigenstate encoding $|\pi\rangle$. The number of applications of $W(P)$ needed to resolve the eigenvalue-1 eigenspace from its neighbours scales as $O(1/\sqrt{\delta})$ (Szegedy, 2004), mirroring the spectral-gap speedup from the theorem above.

Remark 2 (The $\pi_{\min}$ caveat): The full runtime is more subtle than the spectral-gap factor alone: the success probability of projecting onto $|\pi\rangle$, starting from a generic initial state such as the uniform superposition, depends on the squared overlap of that state with $|\pi\rangle$, which can be as small as $\Omega(\pi_{\min})$ for $\pi_{\min} = \min_x \pi(x)$. In high-dimensional Bayesian posteriors, $\pi_{\min}$ can be exponentially small in the parameter dimension, so the direct (non-annealed) algorithm can require $1/\sqrt{\pi_{\min}}$ extra resources and underperform classical MCMC. **Annealed QMCMC** mitigates this with a slowly varying sequence $P_0, \dots, P_r$ with π_0 easy to prepare and $\langle \pi_i | \pi_{i+1} \rangle \geq p > 0$, achieving total runtime $O(r\sqrt{\tau} \log^2(r/\epsilon)(1/p) \log(1/p))$ for relaxation time τ (Lopatnikova *et al.*, 2024). Practitioners should treat the direct algorithm with caution until this overlap is well controlled.

Table 4: Quadratic speedup of quantum walks over classical Markov chains, measured in number of walk steps.

Chain type	Classical	Quantum	Why it differs
Two-state chain	$O(1/(\alpha+\beta))$	$O(1/\sqrt{\alpha+\beta})$	Coherent rotation toward $ \pi\rangle$ vs. random diffusion.
Random walk on cycle (hitting time)	$O(N^2)$	$O(N)$	Ballistic vs. diffusive propagation (Childs <i>et al.</i> , 2003; Kempe, 2003).
General reversible chain (mixing, Szegedy walk)	$O(1/\delta)$	$O(1/\sqrt{\delta})$	Phase gap $\Omega(\sqrt{\delta})$ (Szegedy, 2004); possible $\pi_{\min}$ overhead, see remark above.

⁴This is a hitting-time statement, not mixing-time: a discrete-time quantum walk on a cycle does not converge to a stationary distribution as a classical ergodic chain does, since evolution under a fixed unitary is reversible. The Szegedy-walk *mixing* speedup above uses phase estimation, not repeated unitary evolution, to localise on $|\pi\rangle$.

8. Maximum likelihood estimation

Maximum likelihood estimation is one of the most widely used methods for parameter estimation in all of statistics. Given data, we find the parameter value that makes the observed data most probable under the assumed model, a deceptively simple idea that underlies an enormous fraction of applied statistical practice, from simple coin-flip inference to the most complex regression models.

Definition 23 (Likelihood and MLE): For i.i.d. $X_1, \dots, X_n$ with density $f(x; \theta)$, the likelihood is $L(\theta) = \prod_i f(x_i; \theta)$, log-likelihood $\ell(\theta) = \sum_i \log f(x_i; \theta)$, and $\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} \ell(\theta)$.

Under standard regularity conditions, the MLE enjoys three important asymptotic properties that explain why it is the default choice across so much of statistics. First, **consistency**: $\hat{\theta}_{\text{MLE}} \xrightarrow{p} \theta$ as $n \rightarrow \infty$, so with enough data the estimator converges to the true parameter value. Second, **asymptotic normality**: $\sqrt{n}(\hat{\theta}_{\text{MLE}} - \theta) \xrightarrow{d} N(0, I(\theta)^{-1})$, where $I(\theta)$ is the Fisher information, so the estimator is approximately normally distributed around the truth for large n , which is what licenses the usual confidence-interval constructions. Third, **asymptotic efficiency**: the MLE achieves the Cramér-Rao lower bound asymptotically, meaning no other consistent, asymptotically unbiased estimator can do better in terms of asymptotic variance.

Example 18 (MLE of coin bias): $\ell(\theta) = k \log \theta + (n-k) \log(1-\theta)$; $d\ell/d\theta = k/\theta - (n-k)/(1-\theta)$; $d\ell/d\theta = 0 \Rightarrow \hat{\theta}_{\text{MLE}} = k/n$, confirmed a maximum since $d^2\ell/d\theta^2 = -k/\theta^2 - (n-k)/(1-\theta)^2 < 0$.

A particularly important special case of MLE, and one that will recur in Section 8's discussion of HHL, is linear regression. The model is $y = X\beta + \varepsilon$, where $y \in \mathbb{R}^n$ is the vector of responses, $X \in \mathbb{R}^{n \times d}$ is the design matrix, $\beta \in \mathbb{R}^d$ is the vector of regression coefficients we wish to estimate, and $\varepsilon \sim N(0, \sigma^2 I_n)$ is Gaussian noise. The log-likelihood under this model is $\ell(\beta) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|y - X\beta\|^2$; since the first term does not depend on β , maximizing $\ell(\beta)$ is exactly equivalent to minimizing the residual sum of squares $\|y - X\beta\|^2$, which is the familiar ordinary least squares criterion. Differentiating and setting the gradient to zero gives the normal equations $X^\top X \hat{\beta} = X^\top y$, so whenever $X^\top X$ is invertible, $\hat{\beta} = (X^\top X)^{-1} X^\top y$, the closed-form solution every statistician encounters in their first regression course, here derived as a special case of the same maximum-likelihood machinery we used for the coin.

Quantum Phase Estimation (QPE)

Before we can understand the HHL algorithm, we need to understand Quantum Phase Estimation, a fundamental quantum subroutine that recurs throughout quantum algorithms wherever we need to extract information encoded in the phase of a unitary's eigenvalue. QPE estimates the phase θ_U in an eigenvalue $e^{2\pi i \theta_U}$ of a unitary U (Kitaev, 1995; Cleve *et al.*, 1998) (we write θ_U to distinguish from the coin bias θ). Given a unitary U and an eigenstate $|u\rangle$ such that $U|u\rangle = e^{2\pi i \theta_U}|u\rangle$, QPE produces an n -bit approximation of θ_U in an auxiliary register, using n ancilla qubits initialised to $|0\rangle$ and proceeding in four stages:

- **Step 1: Superposition.** Apply a Hadamard gate to each of the n ancilla qubits to create a uniform superposition: $\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle \otimes |u\rangle$. At this point the ancilla register

carries no information about θ_U at all; it is simply an equal-weight superposition over every possible n -bit string.

- **Step 2: Controlled unitaries.** Apply controlled- U^{2^j} , with the j -th ancilla qubit as control, for each j . Because $|u\rangle$ is an eigenstate of U , each controlled application multiplies the corresponding branch of the superposition by the appropriate power of the eigenvalue phase, leaving the state $\frac{1}{\sqrt{2^n}} \sum_k e^{2\pi i \theta_U k} |k\rangle \otimes |u\rangle$: the phase θ_U is now encoded in the relative phases between the $|k\rangle$ branches of the ancilla register, though it is not yet directly observable.
- **Step 3: Inverse QFT.** Applying the inverse Quantum Fourier Transform to the ancilla register converts these relative phases into an ordinary binary amplitude pattern, concentrating the amplitude on (or near) the n -bit binary expansion of θ_U .
- **Step 4: Measurement.** Measuring the ancilla register in the computational basis now yields the best n -bit estimate of θ_U with probability $\geq 4/\pi^2 \approx 0.4$ (exactly, if $2^n \theta_U$ happens to be an integer); this success probability can be boosted arbitrarily close to 1 using standard repetition or median tricks (Cleve *et al.*, 1998).

The HHL algorithm

The HHL algorithm, named after its inventors Harrow, Hassidim, and Lloyd (Harrow *et al.*, 2009), solves linear systems of equations exponentially faster than classical algorithms under certain conditions. This makes it a powerful tool for quantum maximum likelihood estimation, particularly for the linear regression problem we just worked through classically. For an improved version of the algorithm with better dependence on the target precision, see Childs *et al.* (2017); for applications of HHL-style techniques to quantum machine learning more broadly, see Rebentrost *et al.* (2014); Schuld *et al.* (2016); Kerenidis and Prakash (2017).

Problem statement. We are given an $N \times N$ Hermitian matrix A (non-Hermitian matrices can always be embedded into a larger Hermitian matrix via $\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix}$, so this restriction costs no generality), assumed to be s -sparse, meaning at most s non-zero entries per row or column, and well-conditioned, with condition number $\kappa = \lambda_{\max}/\lambda_{\min}$ reasonably small. We are also given a quantum state $|b\rangle = \frac{1}{\|b\|} \sum_i b_i |i\rangle$ encoding the right-hand side of the linear system. The goal is to produce the quantum state $|x\rangle = A^{-1}|b\rangle/\|A^{-1}|b\rangle\|$, encoding the (normalised) solution to $Ax = b$.

Theorem 8 (HHL runtime (Harrow *et al.*, 2009)): The original HHL algorithm solves the quantum linear systems problem in time $O(s^2 \kappa^2 \log N/\epsilon)$, where ϵ is the allowed error. For sparse matrices ($s = O(\text{polylog } N)$) and well-conditioned matrices ($\kappa = O(1)$), this is exponentially faster in N than the best classical algorithms, which require $O(Ns\kappa)$ time using sparse conjugate-gradient-type methods, or $O(N^\omega)$ with $\omega \approx 2.37$ for dense matrices via fast matrix multiplication. Subsequent work has substantially improved the dependence on ϵ and κ : in particular, Childs *et al.* (2017) achieve $O(s\kappa \text{polylog}(s\kappa/\epsilon))$ query complexity, exponentially better in $1/\epsilon$ than the original bound. We state the original $O(s^2 \kappa^2 \log N/\epsilon)$ bound here because it corresponds directly to the elementary, easy-to-follow construction we walk through step by step below; readers implementing HHL in practice should use the

improved algorithm of Childs *et al.* (2017) instead.

Step-by-step HHL

- **Step 1: Quantum Phase Estimation.** The first step decomposes the input state $|b\rangle$ into the eigenbasis of A . With eigenvectors $|u_j\rangle$ and eigenvalues λ_j of A , write $|b\rangle = \sum_j \beta_j |u_j\rangle$; applying QPE to the unitary e^{iA} (constructed via Hamiltonian simulation (Lloyd, 1996; Berry *et al.*, 2007), since e^{iA} is not given to us directly) yields $\text{QPE}(|b\rangle|0\rangle) = \sum_j \beta_j |u_j\rangle |\tilde{\lambda}_j\rangle$, entangling each eigenvector branch with an estimate of its own eigenvalue.
- **Step 2: Controlled rotation.** Conditional on the estimated eigenvalue $\tilde{\lambda}_j$ sitting in an auxiliary register, apply a small rotation to a fresh ancilla qubit: $|u_j\rangle |\tilde{\lambda}_j\rangle |0\rangle \mapsto |u_j\rangle |\tilde{\lambda}_j\rangle \left(\sqrt{1 - (C/\tilde{\lambda}_j)^2} |0\rangle + (C/\tilde{\lambda}_j) |1\rangle \right)$, where the constant $C = O(1/\kappa)$ is chosen so that $C/|\tilde{\lambda}_j| \leq 1$ for every j , this is always possible because every eigenvalue satisfies $\lambda_{\min} \leq |\lambda_j| \leq \lambda_{\max}$, so $C/|\lambda_j| \leq C\kappa/\lambda_{\max} \leq 1$ once $C = O(\lambda_{\max}/\kappa)$ after rescaling so that $\lambda_{\max} = O(1)$. The point of this rotation is that it secretly encodes the factor $1/\tilde{\lambda}_j$, exactly the eigenvalue-inversion we need to compute A^{-1} , into the amplitude of the auxiliary qubit being $|1\rangle$.
- **Step 3: Postselection.** We measure the auxiliary qubit and keep only the runs where it comes out $|1\rangle$; the probability of this happening is $p_{\text{succ}} = \sum_j |\beta_j C/\tilde{\lambda}_j|^2$, which can be uncomfortably small when κ is large, this is precisely what drives the κ^2 dependence (or, in the improved algorithm, the milder κ dependence) in the runtime bound above, since naively we would need $O(1/p_{\text{succ}})$ repetitions to see a successful run. In practice this naive overhead is avoided by applying amplitude amplification to the postselection step itself, boosting the expected number of repetitions down to $O(1)$. Conditional on success, the (unnormalised) state is $\sum_j \beta_j (C/\tilde{\lambda}_j) |u_j\rangle |\tilde{\lambda}_j\rangle |1\rangle$, which we then renormalise by $1/\sqrt{p_{\text{succ}}}$.
- **Step 4: Uncomputation.** Finally, we apply the inverse of QPE to clear the eigenvalue register back to $|0\rangle$, leaving $|x\rangle = \frac{1}{\|A^{-1}|b\rangle\|} \sum_j \frac{\beta_j}{\lambda_j} |u_j\rangle = A^{-1}|b\rangle/\|A^{-1}|b\rangle\|$, exactly the normalised solution vector we wanted, encoded entirely in the amplitudes of a quantum state.

Example 19 (2×2 system): Consider the simple 2×2 system $A = \text{diag}(2, 1)$, $b = (1, 1)$. Classically, the solution is immediate: $A^{-1}b = (1/2, 1)$. Let us verify that the quantum construction above reproduces this answer. In the quantum setting, the eigenvalues of A are $\lambda_1 = 2, \lambda_2 = 1$, with corresponding eigenvectors $|u_1\rangle = |0\rangle$, $|u_2\rangle = |1\rangle$; the normalised right-hand side is $|b\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, so $\beta_1 = \beta_2 = 1/\sqrt{2}$. QPE records the eigenvalue estimates $\tilde{\lambda}_1 \approx 2, \tilde{\lambda}_2 \approx 1$. The controlled rotation in Step 2 then multiplies the amplitude on $|0\rangle$ by $C/2$ and the amplitude on $|1\rangle$ by $C/1 = C$, giving (before normalisation) amplitudes $(1/\sqrt{2})(C/2) = C/(2\sqrt{2})$ on $|0\rangle$ and $(1/\sqrt{2})(C) = C/\sqrt{2}$ on $|1\rangle$. After postselection and uncomputation, renormalising while preserving this ratio gives:

$$|x\rangle = \frac{C/(2\sqrt{2})}{\sqrt{(C/(2\sqrt{2}))^2 + (C/\sqrt{2})^2}} |0\rangle + \frac{C/\sqrt{2}}{\sqrt{(C/(2\sqrt{2}))^2 + (C/\sqrt{2})^2}} |1\rangle = \frac{1}{\sqrt{5}} |0\rangle + \frac{2}{\sqrt{5}} |1\rangle,$$

which is exactly proportional to $A^{-1}b = (1/2, 1)$, matching the classical answer exactly up to the overall normalisation that any quantum state must carry. This small example, worked through in full, is meant to make the four abstract steps above concrete enough to trust before we discuss the algorithm’s broader implications.

Why is HHL exponentially faster than classical algorithms, at least in N and under the sparsity and conditioning assumptions stated above? The key conceptual shift is that HHL works with quantum states that encode the solution, rather than with classical vectors that must be written down entry by entry. The matrix A is never fully stored in memory; instead, it is accessed only through a sparse oracle that can be queried in superposition. Quantum phase estimation then allows HHL to “invert” all eigenvalues simultaneously, exploiting quantum parallelism in a way that has no classical counterpart, while Hamiltonian simulation (Lloyd, 1996; Berry *et al.*, 2007) provides the time evolution e^{-iAt} that QPE needs as its input unitary.

Remark 3 (Limitations): Two important limitations temper this exponential promise. First, the speedup is exponential in N only for sparse, well-conditioned matrices; for dense or ill-conditioned matrices the theoretical advantage disappears entirely, since both s and κ then scale with N themselves. Second, and perhaps more importantly for practitioners, the output of HHL is a *quantum state* $|x\rangle$, not a classical vector. Extracting the full classical solution vector requires quantum tomography, which as we will see in Section 10 takes $O(N/\epsilon^2)$ measurements, potentially negating the speedup entirely if the full vector is what is actually needed (O’Donnell and Wright, 2016). Data loading, meaning the preparation of $|b\rangle$ itself in the first place, can also be a serious bottleneck (Giovannetti *et al.*, 2008; Kerenidis and Prakash, 2017).

Example 20 (Quantum linear regression): Suppose we have n data points in d dimensions, with $n \ll d$, a high-dimensional regression setting that is increasingly common in modern applications. Classical ridge regression, solving $(X^\top X + \lambda I)\hat{\beta} = X^\top y$, costs $O(d^3)$ time in general, or $O(nd^2)$ using the Woodbury identity when $n \ll d$ lets us avoid inverting the full $d \times d$ matrix directly. If the design matrix X is low-rank and we have quantum access to the data via QRAM (Giovannetti *et al.*, 2008; Kerenidis and Prakash, 2017), the HHL algorithm can estimate $\hat{\beta}$ in $O(\text{polylog}(d))$ time (Rebentrost *et al.*, 2014; Schuld *et al.*, 2016), an exponential speedup in the dimension d , though subject to the same sparsity, conditioning, and readout caveats discussed throughout this section.

Remark 4 (When HHL helps, and when it doesn’t): HHL’s exponential-in- N speedup needs both sparsity and good conditioning. Statistical design matrices $X^\top X$ are rarely sparse, and ill-conditioning is common in high-dimensional or collinear regression, exactly the regime where most challenging statistical problems live. In such settings classical ridge regression or conjugate gradient may outperform the quantum approach; sparsity and condition number should be verified explicitly before invoking HHL.

9. Bayesian statistics and posterior mode

Bayesian statistics provides a coherent framework for updating beliefs in light of data, and the posterior distribution it produces encodes all of our uncertainty about the

parameters after observing that data. The basic machinery, Bayes' rule and the law of total probability, was already introduced in Section 3; here we specialise that machinery to the Bayesian modelling workflow and then turn, as in every previous section, to its quantum analogue.

Definition 24 (Bayesian model): A Bayesian model consists of a prior distribution $p(\theta)$ over parameters $\theta \in \Theta$, expressing our beliefs about θ before seeing any data, together with a likelihood $p(y | \theta)$ for the observed data y given the parameters. Bayes' theorem, exactly as stated in Section 3, then gives the posterior distribution $p(\theta | y) = p(y | \theta)p(\theta)/p(y) \propto p(y | \theta)p(\theta)$, where $p(y) = \int p(y | \theta)p(\theta) d\theta$ is the marginal likelihood, or evidence (Gelman *et al.*, 2013).

The posterior distribution is, in principle, the complete answer to any inference question we might ask about θ . In practice, however, it is often high-dimensional and difficult to summarise directly, so two single-number summaries are reported most often. The **posterior mean** is $\mathbb{E}[\theta | y] = \int \theta p(\theta | y) d\theta$; the **posterior mode**, also called the maximum a posteriori (MAP) estimate, is $\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(\theta | y) = \arg \max_{\theta} [\log p(y | \theta) + \log p(\theta)]$. These two summaries coincide only when the posterior happens to be symmetric; in general they can differ substantially, particularly for skewed posteriors. When the prior is uniform, the MAP estimate coincides exactly with the MLE from Section 8, since the prior term then contributes nothing to the maximisation.

Example 21 (Beta prior): Suppose we place a $\text{Beta}(\alpha, \beta)$ prior on the coin bias, $p(\theta) = \theta^{\alpha-1}(1-\theta)^{\beta-1}/B(\alpha, \beta)$. The Beta distribution is conjugate to the binomial likelihood, meaning the posterior is again a Beta distribution: after observing k heads in n flips, $p(\theta | k) \propto p(k | \theta)p(\theta) \propto \theta^k(1-\theta)^{n-k} \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1} = \theta^{k+\alpha-1}(1-\theta)^{n-k+\beta-1}$, which we recognise as proportional to a $\text{Beta}(k+\alpha, n-k+\beta)$ density. For $k+\alpha > 1$ and $n-k+\beta > 1$, the mode of a $\text{Beta}(a, b)$ distribution is $(a-1)/(a+b-2)$, giving $\hat{\theta}_{\text{MAP}} = (k+\alpha-1)/(n+\alpha+\beta-2)$. When $\alpha = \beta = 1$ (the uniform prior), this reduces exactly to the MLE k/n , confirming the general fact noted above (Casella and Berger, 2002).

Quantum Bayesian inference via QMCMC

The quantum posterior state $|\pi\rangle = \sum_{\theta} \sqrt{\pi(\theta)}|\theta\rangle$ for $\pi(\theta) = p(\theta | y)$ is exactly the quantum sample state of Section 4 applied to the posterior. The Szegedy walk construction of Section 7 applies verbatim: given a classical chain on the parameter space with stationary distribution π and spectral gap δ , $W(P)$ has $|\pi\rangle$ as an eigenvalue-1 eigenstate, with $O(1/\sqrt{\delta})$ walk-step dependence versus $O(1/\delta)$ classically, subject to the same $\pi_{\min}$ caveat from Section 7, especially relevant here since high-dimensional posteriors are exactly where $\pi_{\min}$ tends to be exponentially small. For a detailed treatment of quantum Bayesian inference, see Wiebe *et al.* (2012).

Posterior mode via quantum search

Finding the posterior mode is fundamentally an optimization problem: we are searching over the parameter space for the point of highest posterior density. Grover's search al-

gorithm (Grover, 1996) provides a quadratic speedup for unstructured search, as we saw in Section 6, and its generalization, due to Dürr and Høyer, adapts naturally to the related task of finding the maximum of a function rather than a single marked item.

Definition 25 (Quantum maximum finding algorithm (Durr and Hoyer, 1996)): The algorithm proceeds iteratively, using Grover’s algorithm as a subroutine at each step. First, randomly select an initial index y and compute $f(y)$, establishing a baseline threshold. Then repeat the following until convergence: use the current threshold $f(y)$ to mark all items x such that $f(x) > f(y)$ (these are the items that would improve on our current best guess); apply Grover’s algorithm to amplify the marked items, exactly as in Section 6; measure to obtain a new index y' with $f(y') > f(y)$; and update $y = y'$ before repeating. Each round of this loop costs $O(\sqrt{N})$ Grover queries against the current (shrinking) set of marked items, and a careful accounting of the expected number of rounds needed shows that the algorithm finds the global maximum using expected $O(\sqrt{N})$ oracle calls in total (Durr and Hoyer, 1996), the same square-root scaling we have now seen repeatedly throughout this article.

Theorem 9 (Quantum MAP speedup): Finding $\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} \pi(\theta)$ over N discrete values needs expected $O(\sqrt{N})$ queries to a posterior-comparison oracle, versus $O(N)$ classically.

Example 22 (Quantum MAP): Suppose we discretise the space of possible coin biases into $\Theta = \{0.001, 0.002, \dots, 0.999\}$, a set of size $N = 999$. After observing some data, we want to find the bias with the highest posterior probability. Classical exhaustive search must check all 999 values, requiring 999 posterior evaluations. Quantum maximum finding requires only approximately $\sqrt{999} \approx 32$ queries to the posterior evaluation oracle, a striking reduction even at this modest scale. The advantage becomes dramatic as the parameter space grows: for $N = 1$ million candidate values, classical search still requires the full 1 million evaluations, while quantum maximum finding requires only about 1000.

Table 5: Quadratic speedups for Bayesian computation, in respective query/step complexity models (coherent oracle access assumed; see Section 10 for caveats governing wall-clock advantage)

Task	Classical	Quantum	Why it differs
Posterior sampling (MCMC)	$O(1/\delta)$	$O(1/\sqrt{\delta})$	Szegedy walk (Szegedy, 2004); coherent rotation vs. random diffusion.
Posterior mode (MAP)	$O(N)$	$O(\sqrt{N})$ expected	Grover/max-finding (Durr and Hoyer, 1996).
Posterior mean	$O(1/\epsilon^2)$	$O(1/\epsilon)$	QAE (Brassard <i>et al.</i> , 2002; Montanaro, 2015); away from boundary regime.

10. Practical considerations and open problems

The exponential-in- N speedups of quantum algorithms like HHL and quantum principal component analysis (Lloyd *et al.*, 2014) all assume that data can be loaded into quantum states efficiently. This assumption deserves scrutiny: it is not always true, and the two bottlenecks below, data loading and state readout, together govern whether a provable

query-complexity speedup ever shows up as a real, wall-clock advantage on actual hardware.

Definition 26 (QRAM (Giovannetti *et al.*, 2008)): QRAM is a data structure that allows quantum superposition access to classical data, the quantum analogue of an ordinary computer’s random access memory, but built to accept a query in superposition and return the corresponding superposition of answers. It performs $\sum_i a_i |i\rangle |0\rangle \mapsto \sum_i a_i |i\rangle |x_i\rangle$ with query *depth* $O(\log N)$, but the bucket-brigade architecture achieving this requires $O(N)$ physical components and very high-fidelity operations on all of them simultaneously, challenging with current hardware, and a major reason QRAM remains more a theoretical assumption than an engineering reality at present. An alternative architecture is given by Kerenidis and Prakash (2017).

Closely related to QRAM is the separate question of *amplitude encoding* cost. Loading an arbitrary vector $x \in \mathbb{C}^N$ into amplitudes $|x\rangle = \sum_i x_i |i\rangle$ requires $O(N)$ operations in general, since each amplitude must in principle be individually programmed via a sequence of controlled rotations (Prakash, 2014); there is no shortcut around this for a truly generic vector, since the circuit must, in effect, specify N independent real numbers. Structured data can sometimes avoid this cost entirely: Grover and Rudolph (Grover and Rudolph, 2002) showed that distributions with an efficiently integrable cumulative distribution function, the normal distribution being the canonical example, can be prepared using only $O(\text{polylog } N)$ operations, an exponential improvement over the generic bound. But Herbert (2021) showed that this method is fundamentally limited to distributions simple enough to not need Monte Carlo in the first place, precisely the distributions for which a quantum speedup is least needed in practice. This raises real doubt about whether efficient quantum state preparation is available for the complex, empirical distributions that statisticians actually encounter in real applications, as opposed to the clean textbook distributions used to illustrate the algorithms.

Definition 27 (Quantum and shadow tomography): Full state tomography of an N -dimensional pure state requires $\Theta(N/\epsilon^2)$ copies to achieve error ϵ in trace distance (O’Donnell and Wright, 2016), impossible for large N (e.g. $N = 2^{100}$). This means that for many quantum algorithms that produce an N -dimensional solution vector, we genuinely cannot read out the entire vector, no matter how cheap state preparation was. **Shadow tomography** (Huang *et al.*, 2020) instead estimates M chosen observables to precision ϵ using only $O(\log M)$ copies (with additional polynomial dependence on $1/\epsilon$), via random unitary rotations before measurement, sufficient for the low-dimensional summaries (regression coefficients, moments, threshold tests) statisticians usually want, even when full tomography is hopeless.

To make the contrast concrete: suppose HHL has just solved a regression problem with $d = 20$ qubits worth of coefficients, so $N = 2^{20} \approx 10^6$ coefficients in total. Reading out the entire solution vector to modest precision $\epsilon = 0.01$ via full tomography would require on the order of $N/\epsilon^2 \approx 10^{10}$ copies of the state, wildly impractical even on a perfect quantum computer. But if a data analyst only actually needs, say, the 20 largest coefficients or a single linear combination of them, shadow tomography can extract that summary using only on the order of $\log(20) \approx 4$ -ish copies, up to the polynomial overhead in $1/\epsilon$: a difference of many orders of magnitude that determines, in practice, whether HHL’s exponential speedup in state *preparation* survives contact with the need to actually read an answer back out.

Putting the data-loading and readout bottlenecks together, we can now summarise, in one place, when quantum advantage is plausible for a given statistical problem and when it is not. Table 6 below is meant to function as a quick diagnostic checklist: a problem with several entries from the left column is a genuinely promising candidate for a quantum approach, while a problem with several entries from the right column is likely to see its theoretical speedup erased in practice by one bottleneck or another, however elegant the underlying algorithm.

Table 6: Conditions for quantum advantage. NISQ (Noisy Intermediate-Scale Quantum) was coined by Preskill (2018)

Favorable conditions	Unfavorable conditions
Sparse or low-rank matrices	Dense, full-rank matrices
Well-conditioned (κ small)	Ill-conditioned (κ large)
Quantum access to data (QRAM available)	Classical data requiring amplitude encoding
Low-dimensional summary needed	Full solution vector required
Quantum queries dominate runtime	Data loading or readout dominates
Quadratic speedup is sufficient	Exponential speedup needed but unrealistic
Current NISQ devices (Preskill, 2018)	Fault-tolerant quantum computers required

Open problems for statisticians

The intersection of quantum computing and statistics has many open problems that are ripe for research. We close this section with six of them.

1. **Efficient quantum sample preparation:** Preparing quantum sample states for arbitrary distributions of practical interest remains an open problem. Current methods either require unrealistic assumptions (Grover–Rudolph (Grover and Rudolph, 2002)) or scale poorly (general amplitude encoding (Prakash, 2014)). Statisticians could help by designing preparation methods tailored to structured distributions that arise often in practice, exponential families, mixture models, Bayesian priors, rather than aiming for full generality.
2. **Quantum sequential Monte Carlo:** Quantizing particle filters for state-space models is an open research question. Sequential Monte Carlo is an important alternative to MCMC for high-dimensional, time-varying problems, and a quantum version could in principle offer similar speedups to those we saw for Szegedy walks.
3. **Quantum variational Bayes:** While hybrid quantum-classical variational algorithms already exist (Schuld *et al.*, 2020), implementing variational inference entirely on a quantum computer, rather than splitting the work between quantum and classical processors, is still an open problem.
4. **Quantum normalizing flows:** Using parameterized quantum circuits to transform quantum sample states may enable more efficient readout than generic shadow tomography, inspired by classical normalizing flows that transform simple distributions into complex ones.

5. **Quantum-inspired classical algorithms:** Some quantum algorithms have, perhaps surprisingly, led to improved *classical* algorithms. Quantum-inspired linear algebra (Gilyén *et al.*, 2022) achieves polynomial speedups on ordinary classical computers for certain low-rank problems; adapting these ideas to statistical computation is an active area that requires no quantum hardware at all.
6. **Better readout methods:** Quantum tomography is too expensive for large states, and shadow tomography (Huang *et al.*, 2020) is a promising start, but more work is needed to develop methods that extract precisely the summaries statisticians care about, posterior means, credible intervals, Bayes factors, rather than generic observable expectations.

What unites all six of these open problems is that they require genuine statistical expertise to make progress on, not just quantum computing expertise: knowing which distributions actually arise in practice, which summaries practitioners actually need, and which approximations are acceptable for a given application are statistical judgments, not algorithmic ones. This is precisely why the field needs more statisticians working in it, and why this article exists at all.

11. Conclusion

We have travelled a long road together. We began with Kolmogorov’s axioms and built up, step by step, to the quantum algorithms that promise to transform statistical computation, returning at every stage to the same single biased coin to ground the abstraction in something concrete. Sections 2–4 established the dictionary between classical probability and quantum states: Hilbert spaces stand in for sample spaces, state vectors for distributions, observables for random variables, and quantum sample states $|P\rangle = \sum_x \sqrt{p(x)}|x\rangle$ encode entire distributions in $O(\log N)$ qubits of storage, an idea we first met as an abstraction in Section 4 and then watched concretely turn our 101-outcome binomial distribution into a 7-qubit object. Section 5 carried expectation and variance across the same dictionary, showing that $\langle O \rangle = \text{tr}(O\rho)$ is exactly the quantum mirror of $\mathbb{E}[X] = \sum_x xp(x)$, with the coin’s Pauli-Z expectation $2\theta - 1$ as the worked check. Sections 6–9 then built the four central algorithms on top of this dictionary, each illustrated on the coin or on a discrete approximation to it: Grover’s search and Quantum Amplitude Estimation (Section 6) give a quadratic speedup for mean estimation, taking our coin-bias estimation problem from 10^6 classical flips down to roughly 1000 quantum oracle queries; Szegedy quantum walks (Section 7) give a quadratic speedup for MCMC mixing; the HHL algorithm (Section 8) gives an exponential speedup for sparse, well-conditioned linear systems, worked through explicitly on a 2×2 example; and quantum maximum-finding (Section 9) gives a quadratic speedup for locating the posterior mode of the coin’s bias. Section 10 then stepped back and detailed the two practical bottlenecks, data loading and state readout, that govern whether any of these provable speedups translates into real-world, wall-clock advantage on actual hardware.

Probability $p(x)$ $\xrightarrow{\sqrt{\cdot}}$ Amplitude $\sqrt{p(x)}$ $\xrightarrow{ \cdot\rangle}$ State $\sum_x \sqrt{p(x)} x\rangle$

This is the quantum dialect of probability: not a replacement for classical probability but an extension of it. Classical probability is the special case where every relevant decomposition is orthogonal, so amplitudes effectively add like ordinary probabilities and nothing distinctively quantum ever happens. Quantum probability allows coherent superpositions of non-orthogonal alternatives, whose amplitudes can interfere constructively or destructively *before* the Born rule converts them into probabilities, and this interference, which has simply no classical counterpart whatsoever, is the engine of every speedup in this article, from Grover’s search through Szegedy walks to HHL. If there is one idea a statistician should carry away from this entire article, it is this: quantum advantage is not magic, and it is not merely “more computing power.” It is a specific, well-understood mathematical mechanism, constructive and destructive interference between coherently maintained amplitudes, that has no analogue anywhere in classical probability theory, and understanding where that mechanism can and cannot be exploited is the key to understanding when quantum computing will actually help with a given statistical problem.

Table 7: Classical probability and statistics vs. quantum analogues (concepts)

Classical	Quantum	Why it differs
Sample space Ω	Hilbert space $\mathcal{H}$	States are vectors over $\mathbb{C}$, not sets.
Distribution $p(\omega)$	State vector $ \psi\rangle$	Amplitudes $\psi_i \in \mathbb{C}$; $P(i) = \psi_i ^2$.
$p_i \geq 0$, add	$\psi_i \in \mathbb{C}$; add within a basis	Interference for non-orthogonal superpositions.
$X : \Omega \rightarrow \mathbb{R}$	Observable	Eigenvalues replace function values.
$\mathbb{E}[X] = \sum_x xp(x)$	$O = \sum_o o o\rangle\langle o $ $\langle O \rangle = \text{tr}(O\rho) = \langle \psi O \psi \rangle$	Same structure; p from Born rule.
Condition on ω	Collapse to $ i\rangle$; $\rho_m = M_m \rho M_m^\dagger / p_m$	Measurement changes the physical state.
Bayes’ rule (unique)	A quantum Bayes’ update (Wiebe <i>et al.</i> , 2012) (one of several)	No single canonical analogue exists.
Mixture (epistemic)	Mixed state $\rho = \sum_i p_i \psi_i\rangle\langle \psi_i $	No interference (classical ignorance).
No classical analogue	Pure superposition $ \psi\rangle$	Ontological uncertainty, not ignorance.

What quantum computing can do

Quadratic speedups in query complexity are real and achievable for two of the central tasks of statistical computation. Quantum Amplitude Estimation (Brassard *et al.*, 2002; Montanaro, 2015) and Szegedy quantum walks (Szegedy, 2004) offer provable quadratic speedups for mean estimation and MCMC sampling respectively, subject to the boundary-regime and $\pi_{\min}$ caveats discussed throughout this article. These algorithms require coherent quantum computation, but they are an active target for near-term NISQ devices (Preskill, 2018) under reasonable assumptions about hardware quality. Exponential speedups also exist, most notably HHL, but they require sparsity and good conditioning, and the precise polynomial dependence on these two parameters matters a great deal in practice; this dependence was substantially improved by Childs *et al.* (2017) relative to the original 2009

Table 8: Classical probability and statistics vs. quantum analogues (algorithmic speedups; query/step complexity under coherent oracle access, see Section 10 for data-loading and readout caveats)

Classical	Quantum	Why it differs
Monte Carlo: $O(1/\epsilon^2)$ (Metropolis and Ulam, 1949)	QAE: $O(1/\epsilon)$ (Brassard <i>et al.</i> , 2002; Montanaro, 2015)	Phase estimation replaces repeated sampling; away from boundary regime.
MCMC mixing: $O(1/\delta)$ (Geyer, 1992)	Quantum walk: $O(1/\sqrt{\delta})$ (Szegedy, 2004)	Coherent rotation; possible $\pi_{\min}$ overhead.
Linear systems: $O(Ns\kappa)$	HHL: $O(s^2\kappa^2 \log N/\epsilon)$ (Harrow <i>et al.</i> , 2009); improved $O(s\kappa \text{ polylog})$ (Childs <i>et al.</i> , 2017)	Exponential saving in N only for sparse, well-conditioned matrices.
Maximum finding: $O(N)$	Grover/Dürr–Høyer: expected $O(\sqrt{N})$ (Grover, 1996; Dürr and Høyer, 1996)	Interference amplifies marked-state amplitude.
Posterior mode: $O(N)$	Quantum max-finding: expected $O(\sqrt{N})$ (Dürr and Høyer, 1996)	Same mechanism applied to MAP.

algorithm. Finally, quantum sample states are exponentially efficient to *store*: an N -outcome distribution that would need N classical numbers fits into $\log_2 N$ qubits. Both preparation (for unstructured distributions) and readout, however, remain genuinely hard in general, as Section 10 detailed.

What it cannot (yet) do

Quantum computers are not known to solve NP-complete problems in polynomial time. The class BQP (bounded-error quantum polynomial time) is believed to be incomparable with NP rather than a strict superset of it, though this remains an open question in complexity theory and should not be overstated either way. More practically: speedups are problem-specific, not universal. For many real statistical problems, the data-loading bottleneck (Giovannetti *et al.*, 2008; Grover and Rudolph, 2002) or the readout bottleneck (O’Donnell and Wright, 2016) silently erases the asymptotic advantage that looked so attractive on paper. Quantum methods, at least for the foreseeable future, will complement classical statistics rather than replace it; the most promising near-term path forward is hybrid quantum-classical algorithms that split work between the two kinds of processor (Schuld *et al.*, 2020).

A practical checklist

When considering whether a quantum approach is worth pursuing for a particular statistical problem, four questions are worth asking in order. **Can the data be loaded ef-**

efficiently? If it has known structure (*e.g.* comes from an efficiently integrable distribution), Grover–Rudolph-style preparation (Grover and Rudolph, 2002) may work; otherwise QRAM (Giovannetti *et al.*, 2008; Kerenidis and Prakash, 2017) or another loading scheme will be needed, at real hardware cost. **Is the full solution vector actually needed, or just a summary?** If only a few coefficients, a credible interval, or a hypothesis-test result is required, shadow tomography (Huang *et al.*, 2020) can extract it efficiently; if the entire vector must be read out classically, full tomography (O’Donnell and Wright, 2016) may erase the speedup entirely. **Is the relevant matrix sparse and well-conditioned?** HHL’s exponential advantage depends on both properties holding, and most statistical design matrices satisfy neither. **Is a quadratic speedup actually enough for the application, or is an unrealistic exponential one being implicitly assumed?** Quadratic speedups for mean estimation and MCMC are proven and practical targets; exponential speedups require much more careful justification.

Where to go next

This article has provided the conceptual foundation; to go deeper, several resources are worth knowing about.

For the full mathematical machinery of quantum computation: Nielsen and Chuang’s (Nielsen and Chuang, 2010) *Quantum Computation and Quantum Information* is the standard graduate textbook, covering all the algorithms we discussed and a great deal more; Kitaev, Shen, and Vyalii’s *Classical and Quantum Computation* offers a more theoretical, complexity-focused treatment. For a comprehensive statistician-oriented survey with detailed complexity analyses that goes well beyond what we covered here: Lopatnikova, Tran, and Sisson (Lopatnikova *et al.*, 2024), the review that directly inspired this article. For quantum machine learning specifically: Schuld and Petruccione’s (Schuld *et al.*, 2016) *Supervised Learning with Quantum Computers*.

For hands-on practice rather than further reading: Qiskit (IBM) is a Python-based quantum computing SDK with free access to real IBM quantum hardware and extensive tutorials; Cirq (Google) is a comparable Python framework oriented toward NISQ-era algorithms; and Amazon Braket offers cloud-based access to multiple quantum computing platforms from different hardware vendors in one place. The Qiskit Global Summer School runs annually and is free to attend, and is a good way to go from reading about these algorithms to actually running them.

The field is young, the problems are rich, and the opportunities for statisticians are vast. The next great statistical method may be quantum; the next great quantum algorithm may come from a statistician. The door is open.

Acknowledgements

Both authors were supported by a generous grant to CMI by the Infosys Foundation. We thank the anonymous reviewers for their valuable time, thorough evaluation, and constructive feedback, which significantly enhanced the quality of this manuscript.

Conflict of interest

The authors do not have any financial or non-financial conflict of interest to declare for the research work included in this article.

References

- Berry, D. W., Ahokas, G., Cleve, R., and Sanders, B. C. (2007). Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, **270**, 359–371.
- Billingsley, P. (1995). *Probability and Measure*. Wiley.
- Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, **305**, 53–74.
- Casella, G. and Berger, R. L. (2002). *Statistical Inference*. Duxbury.
- Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., and Spielman, D. A. (2003). Exponential algorithmic speedup by a quantum walk. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 59–68.
- Childs, A. M., Kothari, R., and Somma, R. D. (2017). Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, **46**, 1920–1950.
- Cleve, R., Ekert, A., Macchiavello, C., and Mosca, M. (1998). Quantum algorithms revisited. *Proceedings of the Royal Society A*, **454**, 339–354.
- Durr, C. and Hoyer, P. (1996). A quantum algorithm for finding the minimum. Technical report.
- Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. CRC Press.
- Geyer, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science*, **7**, 473–483.
- Gilyén, A., Song, Z., and Tang, E. (2022). An improved quantum-inspired algorithm for linear regression. *Quantum*, **6**, 754.
- Giovannetti, V., Lloyd, S., and Maccone, L. (2008). Quantum random access memory. *Physical Review Letters*, **100**, 160501.
- Grover, L. and Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions.
- Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA. Association for Computing Machinery.
- Harrow, A. W., Hassidim, A., and Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, **103**, 150502.
- Herbert, S. (2021). No quantum speedup with Grover-Rudolph state preparation for quantum monte carlo integration. *Physical Review E*, **103**, 063302.
- Huang, H.-Y., Kueng, R., and Preskill, J. (2020). Predicting many properties of a quantum system from very few measurements. *Nature Physics*, **16**, 1050–1057.
- Kempe, J. (2003). Quantum random walks: An introductory overview. *Contemporary Physics*, **44**, 307–327.

- Kerenidis, I. and Prakash, A. (2017). Quantum recommendation systems. In Papadimitriou, C. H., editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- Kitaev, A. Y. (1995). Quantum measurements and the Abelian Stabilizer Problem. *Electron. Colloquium Comput. Complex.*, **TR96**.
- Lloyd, S. (1996). Universal quantum simulators. *Science*, **273**, 1073–1078.
- Lloyd, S., Mohseni, M., and Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, **10**, 631–633.
- Lopatnikova, A., Tran, M.-N., and Sisson, S. A. (2024). An introduction to quantum computing for statisticians and data scientists. *Foundations of Data Science*, **6**, 278–307.
- Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, **44**, 335–341.
- Montanaro, A. (2015). Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A*, **471**, 20150301.
- Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.
- O’Donnell, R. and Wright, J. (2016). Efficient quantum tomography. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 899–912.
- Prakash, A. (2014). *Quantum Algorithms for Linear Algebra and Machine Learning*. PhD thesis, University of California, Berkeley, Berkeley, CA.
- Preskill, J. (2018). Quantum computing in the NISQ era and beyond. *Quantum*, **2**, 79.
- Rebentrost, P., Mohseni, M., and Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical Review Letters*, **113**, 130503.
- Schuld, M., Bocharov, A., Svore, K. M., and Wiebe, N. (2020). Circuit-centric quantum classifiers. *Physical Review A*, **101**, 032308.
- Schuld, M., Sinayskiy, I., and Petruccione, F. (2016). Prediction by linear regression on a quantum computer. *Physical Review A*, **94**, 022342.
- Szegedy, M. (2004). Quantum speed-up of markov chain based algorithms. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41.
- Wiebe, N., Braun, D., and Lloyd, S. (2012). Quantum algorithm for data fitting. *Physical Review Letters*, **109**, 050505.

Publisher
Society of Statistics, Computer and Applications
Mailing Address: B-133, Ground Floor, C.R. Park, New Delhi-110019, INDIA
Tele: 011-40517662
<https://ssca.org.in/>
statapp1999@gmail.com
2026

Printed by : Galaxy Studio & Graphics
Mob: +91 9818 35 2203, +91 9582 94 1203