Half Title

Introduction to Bayesian Econometrics:
A GUIded toolkit using R

Title Page

Introduction to Bayesian Econometrics:
A GUIded toolkit using R

by Andrés Ramírez-Hassan, PhD. Statistical Science.

LOC Page

To my fiancée, Estephania, and to my parents, Nancy and Orlando.

Introduction

FIGURE 1
Supposedly portrait of Thomas Bayes.

Since the late 90s, Bayesian inference has gained significant popularity among researchers due to the computational revolution and the availability of algorithms to solve complex integrals. However, many researchers, students, and practitioners still lack a deep understanding and practical application of this inferential approach. The primary reason for this is the requirement for strong programming skills.

Introduction to Bayesian Econometrics: A GUIded Toolkit using R mainly targets those who want to apply Bayesian inference with a good conceptual and formal understanding, but who may not necessarily have the time to develop programming skills. Thus, this book provides a graphical user interface (GUI) to carry out Bayesian regression in a very user-friendly environment. The book also offers the basic theory and its code implementation using **R** software [318], along with some applications to highlight the potential of Bayesian inference. Additionally, it includes theory and computational exercises for those interested in developing more complex models. In particular, this book presents the mathematical proofs of the basic models step-by-step in the first part, which serve as the foundation for obtaining the most relevant mathematical results in the more complex models covered in the second and third parts.

Our GUI is based on an interactive web application using `shiny` [73],

along with several packages in **R**. Users can estimate univariate, multivariate, time series, longitudinal/panel data, and Bayesian model averaging models using our GUI. In addition, it provides basic summaries, as well as formal and graphical diagnostics of the posterior chains. Our GUI can be run on any operating system and is freely available at **https://github.com/besmarter/BSTApp**.

Users can access simulated and real datasets in the folders **DataSim** and **DataApp**, respectively. The former folder also includes the files used to simulate different processes, so the population parameters are available. As a result, these files can be used as a pedagogical tool to demonstrate various statistical properties. The latter folder contains the datasets used in our applications. Users are encouraged to use these datasets as templates for structuring their own datasets.

This book is divided into three parts. The first part covers the theory (both conceptual and mathematical), programming, and simulation foundations (chapters 1 to 4). The second part focuses on the applications of regression analysis, with particular emphasis on the computational aspect of obtaining draws from the posterior distributions at three levels of programming skills: no skills at all using our GUI, intermediate skills using specialized **R** packages for Bayesian inference, and relatively advanced programming skills to obtain posterior draws from scratch (chapters 5 to 10). The third part provides an introductory treatment of *advanced methods* in Bayesian inference (chapters 11 to 14).

I present some of the mathematical deductions in detail in the first part of the book, whereas I do not show most of the proofs in the second and third parts. However, the same mathematical steps in the first part can be used to derive the results in parts two and three.

In the first part, Chapter 1 begins with an introduction to formal concepts in Bayesian inference, starting with Bayes' rule, its components, their formal definitions, and basic examples. It then presents the basics of Bayesian inference based on decision theory under uncertainty. Chapter 2 discusses the conceptual differences between Bayesian and Frequentist statistical approaches, as well as provides a historical and philosophical perspective on Bayesian statistics and econometrics, highlighting the contrasts with the Frequentist approach. In Chapter 3, I introduce conjugate families in basic statistical models, solving them both analytically and computationally. Simulation-based methods are presented in Chapter 4; these algorithms are crucial in modern Bayesian inference since most realistic models do not have standard forms or analytical solutions.

In the second part, Chapter 5 presents our graphical user interface, and univariate and multivariate regression models are covered in Chapters 6 and 7. Chapter 8 focuses on univariate and multivariate time series models, while Chapter 9 covers Bayesian longitudinal/panel data models. Chapter 10 introduces Bayesian model averaging. In the third part, Chapter 11 explores semi-parametric and non-parametric models, Chapter 12 covers Bayesian methods

in machine learning algorithms, Chapter 13 discusses causal inference, and Chapter 14 describes approximation methods.

About me

My name is Andrés Ramírez-Hassan, and I am an applied and theoretical econometrician working as a Distinguished Professor in the School of Finance, Economics, and Government at Universidad EAFIT (Medellín, Colombia). I hold a PhD in Statistical Science, a Master's degree in Finance, another in Economics, and a Bachelor's degree in Economics. I was a research fellow at the Department of Econometrics and Business Statistics at Monash University, and a visiting Professor in the Department of Economics at the University of Melbourne and the University of Glasgow.

Since completing my PhD, much of my research has focused on Bayesian Econometrics, with applications in crime, finance, health, sports, and utilities. My work has been published (or is forthcoming) in highly regarded journals such as the *International Journal of Forecasting*, *Journal of Applied Econometrics*, *Econometric Reviews*, *Journal of Computational and Graphical Statistics*, *The R Journal*, *Economic Modelling*, *Spatial Economic Analysis*, *Economic Inquiry*, *World Development*, *Journal of Sport Economics*, *Empirical Economics*, *Australian and New Zealand Journal of Statistics*, *Brazilian Journal of Probability and Statistics*, and other prestigious international research outlets.

I founded **BEsmarter** – **B**ayesian **E**conometrics: **s**imulations, **m**odels, and **a**pplications to **r**esearch, **t**eaching, and **e**ncoding with **r**esponsibility. This is a research group whose **mission** is to *lead and excel in the generation and dissemination of Bayesian Econometric knowledge through research, teaching, and software*. We **envision** *worldwide econometric research, teaching, and applications based on the Bayesian framework that:*

- Inspires new econometric ideas,
- Creates a user-friendly environment for applications of Bayesian econometrics,
- Transforms classic econometric research, teaching, and applications,
- where one of the main concerns of science is to solve social problems.

Contact: aramir21@gmail.com / aramir21@eafit.edu.co
Website: http://www.besmarter-team.org

FIGURE 2
This book is licensed under the **Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License**.

Contents

Introduction **vii**

Foreword **xi**

Preface **xiii**

Symbols **xv**

I Foundations: Theory, simulation methods and programming **1**

1 Basic formal concepts **3**
1.1 The Bayes' rule 3
1.2 Bayesian framework: A brief summary of theory 10
1.2.1 Example: Health insurance 16
1.3 Bayesian reports: Decision theory under uncertainty 29
1.3.1 Example: Health insurance continues 33
1.4 Summary 35
1.5 Exercises 35

2 Conceptual differences between the Bayesian and Frequentist approaches **37**
2.1 The concept of probability 37
2.2 Subjectivity is not the key 38
2.3 Estimation, hypothesis testing and prediction 39
2.4 The likelihood principle 43
2.5 Why is not the Bayesian approach that popular? 45
2.6 A simple working example 47
2.6.1 Example: Math test 49
2.7 Summary 50
2.8 Exercises 51

3 Cornerstone models: Conjugate families **53**
3.1 Motivation of conjugate families 53
3.1.1 Examples of exponential family distributions 55
3.2 Conjugate prior to exponential family 58
3.2.1 Examples: Theorem 4.2.1 58

3.3 Linear regression: The conjugate normal-normal/inverse gamma model . . . 75
3.4 Multivariate linear regression: The conjugate normal-normal/inverse Wishart model . . . 89
3.5 Summary . . . 93
3.6 Exercises . . . 93

4 Simulation methods 97
4.1 Markov chain Monte Carlo methods . . . 97
4.1.1 Gibbs sampler . . . 98
4.1.2 Metropolis-Hastings . . . 101
4.1.3 Hamiltonian Monte Carlo . . . 105
4.2 Importance sampling . . . 111
4.3 Particle filtering . . . 116
4.4 Convergence diagnostics . . . 127
4.4.1 Numerical standard error . . . 127
4.4.2 Effective number of simulation draws . . . 128
4.4.3 Tests of convergence . . . 129
4.4.4 Checking for errors in the posterior simulator . . . 130
4.5 Summary . . . 135
4.6 Exercises . . . 135

II Regression models: A GUIded toolkit 137

5 Graphical user interface 139
5.1 Introduction . . . 139
5.2 Univariate models . . . 141
5.3 Multivariate models . . . 145
5.4 Time series models . . . 146
5.5 Longitudinal/panel models . . . 149
5.6 Bayesian model average . . . 150
5.7 Help . . . 152
5.8 Warning . . . 153

6 Univariate models 155
6.1 The Gaussian linear model . . . 155
6.2 The logit model . . . 160
6.3 The probit model . . . 165
6.4 The multinomial probit model . . . 167
6.5 The multinomial logit model . . . 172
6.6 Ordered probit model . . . 176
6.7 Negative binomial model . . . 181
6.8 Tobit model . . . 186
6.9 Quantile regression . . . 190
6.10 Bayesian bootstrap regression . . . 193
6.11 Summary . . . 196

6.12 Exercises 196

7 Multivariate models 199
7.1 Multivariate regression 199
7.2 Seemingly unrelated regression 205
7.3 Instrumental variable 210
7.4 Multivariate probit model 215
7.5 Summary 221
7.6 Exercises 221

8 Time series models 225
8.1 State-space representation 226
8.2 ARMA processes 237
8.3 Stochastic volatility models 249
8.4 Vector Autoregressive models 257
8.5 Summary 264
8.6 Exercises 267

9 Longitudinal/Panel data models 269
9.1 Normal model 270
9.2 Logit model 279
9.3 Poisson model 283
9.4 Summary 288
9.5 Exercises 288

10 Bayesian model averaging 291
10.1 Foundation 291
10.2 The Gaussian linear model 296
10.3 Generalized linear models 315
10.4 Calculating the marginal likelihood 323
10.4.1 Savage-Dickey density ratio 324
10.4.2 Chib's methods 325
10.4.3 Gelfand-Dey method 325
10.5 Summary 332
10.6 Exercises 333

III *Advanced* methods: A brief introduction 337

11 Semi-parametric and non-parametric models 339
11.1 Mixture models 339
11.1.1 Finite Gaussian mixtures 340
11.1.2 Direchlet processes 357
11.2 Splines 370
11.3 Summary 382
11.4 Exercises 383

12 Bayesian machine learning **387**
12.1 Cross-validation and Bayes factors 387
12.2 Regularization 390
12.2.1 Bayesian LASSO 392
12.2.2 Stochastic search variable selection 394
12.3 Bayesian additive regression trees 398
12.4 Gaussian process 406
12.5 Tall data problems 416
12.5.1 Divide-and-conquer methods 417
12.5.2 Subsampling-based algorithms 419
12.6 Summary 432
12.7 Exercises 432

13 Causal inference **435**
13.1 Identification setting 435
13.2 Randomized controlled trial (RCT) 437
13.3 Conditional independence assumption (CIA) 446
13.4 Instrumental variables (IV) 453
13.5 Difference-in-differences (DiD) 467
13.6 Sample selection 470
13.7 Regression discontinuity design 470
13.8 Regression kink design 470
13.9 Synthetic control 471
13.10 Event Analysis 471
13.11 Bayesian exponential tilted empirical likelihood 471
13.12 A general framework for updating belief distributions 471
13.13 Bayesian model averaging 471
13.14 Summary 472
13.15 Exercises 472

14 Approximate Bayesian methods **473**
14.1 Simulation-based approaches 473
14.1.1 Approximate Bayesian computation 474
14.1.2 Bayesian synthetic likelihood 480
14.2 Optimization approaches 488
14.2.1 Integrated nested Laplace approximations 488
14.2.2 Variational Bayes 501
14.3 Summary 512
14.4 Exercises 512

Bibliography **515**

Appendix **547**

Foreword

Preface

The main goal of this book is to make the Bayesian inferential framework more approachable to students, researchers, and practitioners who wish to understand and apply this statistical/econometric approach but do not have the time to develop programming skills. I have aimed to strike a balance between applicability and theory. This book provides a very user-friendly graphical user interface (GUI) to implement the most common regression models, while also covering the basic mathematical developments and their code implementation for those interested in advancing to more complex models.

To instructors and students

This book is divided into three parts: foundations (chapters 1 to 4), regression analysis (chapters 5 to 10), and *Advanced* methods (chapters 11 to 14). Our graphical user interface (GUI) is designed for the second part. The source code can be found at **https://github.com/besmarter/BSTApp**. Instructors and students can access all the code, along with simulated and real datasets. There are three ways to install our GUI:

1. Type **shiny::runGitHub("besmarter/BSTApp", launch.browser=T)** in the **R** console or any **R** code editor and execute it.
2. Visit **https://posit.cloud/content/4328505**, log in or sign up for **Posit Cloud**, navigate to the **BSTApp-master** folder in the **Files** tab of the right-bottom window, then click on the **app.R** file and select **Run App**.
3. Use a **Docker** image by typing in the **Command Prompt**
   (a) docker pull magralo95/besmartergui:latest
   (b) docker run –rm -p 3838:3838 magralo95/besmartergui

   Then users can access our GUI going to **http://localhost:3838/**. See Chapter 5 for details.

Students should have a basic understanding of probability theory and statistics, as well as some background in econometrics and time series, particularly regression analysis. Familiarity with standard univariate and multivariate probability distributions is strongly recommended. See a nice summary of useful probability distributions in [173, p. 182-191]. Additionally, students

who wish to master the material in this book should have programming skills in **R** software.[1]

I have included both formal and computational exercises at the end of each chapter to help students gain a better understanding of the material presented. A solutions manual for these exercises accompanies this book.

Instructors can use this book as a textbook for a course on introductory Bayesian Econometrics/Statistics, with a strong emphasis on implementation and applications. This book is intended to be complementary, rather than a substitute, for excellent resources on the topic, such as [150], [71], [350], [173], [163], [240], and [234].

Acknowledgments

I began developing our graphical user interface (GUI) in 2016, after being diagnosed with cervical dystonia. I worked on this side project during weekends, which I called "nerd weekends," and it served as a form of release from my health condition. Once I began to recover, I invited Mateo Graciano, my former student, business partner, and friend, to join the project. He has been instrumental in developing our GUI, and I am enormously grateful to him.

I would also like to thank the members of the BEsmarter research group at Universidad EAFIT, as well as the NUMBATs members at Monash University, for their valuable feedback and recommendations to improve our GUI.

This book is an extension of the paper *A GUIded tour of Bayesian regression* [335], which serves as a brief user guide for our GUI. I decided to write this book to explain the underlying theory and code in our GUI, and to use it as a textbook in my course on Bayesian econometrics/statistics. I am grateful to my students in this course; their insights and thoughtful questions have deepened my understanding of the material.

I also thank Chris Parmeter for his suggestions on how to present our user guide, Professor Raul Pericchi and Juan Carlos Correa for introducing me to Bayesian statistics, and Liana Jacobi, Tomasz Wozniak, and Chun Fung Kwok (Jackson) from the University of Melbourne, as well as David Frazier from Monash University, for engaging talks and amazing collaborations in Bayesian econometrics/statistics. My sincere gratitude goes to Professor Peter Diggle for his unwavering support of my career, and especially to Professor Gael Martin, who gave me the opportunity to work with her, she is a constant source of intellectual inspiration.

Finally, I would like to express my thanks to my colleagues and staff at Universidad EAFIT for their continuous support.

To my parents, Orlando and Nancy, who have always been there for me with their unconditional support. They have taught me that the primary aspect of human spiritual evolution is humility, a lesson I am still learning every day. To my fiancée, Estephania, for her unwavering love and support.

[1] An excellent starting point for **R** programming is the *R Introduction Manual*: **https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf**.

Symbols

Symbol Description

$\neg$	Negation symbol	$\mathbf{0}_l$	l-dimensional null vector
$\propto$	Proportional symbol	$\max_r$	Maximum over r
$\perp$	Independence symbol	$\min_r$	Minimum over r
$\mathcal{R}$	The Real set	∇	Gradiente operator
$\mathcal{R}^K$	Euclidean space of dimension K	$\overset{iid}{\sim}$	Independently and identically distributed
$\emptyset$	Empty set	$\overset{ind}{\sim}$	Independently and Not identically distributed
$\mathbb{1}$	Indicator function		
P	Probability measure	$>$	Greater than
$:=$	Is defined as	$<$	Less than
argmax	Argument of the maximum	$\approx$	Approximately equal to
argmin	Argument of the minimum	$\gtrsim$	Greater than or approximately equal to
tr	Trace operator		
vec	Vectorization operator	Δ	Difference operator
lim	Limit	$\subseteq$	Subset
$\otimes$	Kronecker product	$\subset$	Proper subset
diag$\{\cdot\}$	Diagonal matrix	$\xrightarrow{d}$	Convergence in distribution (law)
$dim\{\cdot\}$	Dimension of an object		
$\boldsymbol{I}_l$	l-dimensional identity matrix	$\xrightarrow{p}$	Convergence in probability
		$\xrightarrow{a.s.}$	Almost surely convergence

Part I

Foundations: Theory, simulation methods and programming

1

Basic formal concepts

We introduce formal concepts in Bayesian inference, beginning with Bayes' rule and its components, along with their formal definitions and basic examples. In addition, we present key features of Bayesian inference, such as Bayesian updating and asymptotic sampling properties. We also cover the basics of Bayesian inference from a decision-theoretic perspective under uncertainty, introducing important concepts like loss functions, risk functions, and optimal decision rules.

1.1 The Bayes' rule

As expected, the starting point for performing Bayesian inference is Bayes' rule,[1] which provides the solution to the inverse problem of determining causes from observed effects. This rule combines prior beliefs with objective probabilities based on repeatable experiments, allowing us to move from observations to probable causes.

Formally, the conditional probability of A_i given B is equal to the conditional probability of B given A_i, multiplied by the marginal probability of A_i, divided by the marginal probability of B,

$$\begin{aligned} P(A_i \mid B) &= \frac{P(A_i, B)}{P(B)} \\ &= \frac{P(B \mid A_i) \times P(A_i)}{P(B)}, \end{aligned} \tag{1.1}$$

where, by the law of total probability, $P(B) = \sum_i P(B \mid A_i)P(A_i) \neq 0$, and $\{A_i, i = 1, 2, \dots\}$ is a finite or countably infinite partition of the sample space.

In the Bayesian framework, B represents sample information that updates a probabilistic statement about an unknown object A_i according to probability rules. This is done using Bayes' rule, which incorporates prior "beliefs"

[1] Note that I use the term "Bayes' rule" rather than "Bayes' theorem." It was Laplace [242] who generalized Bayes' theorem [30], and his generalization is referred to as Bayes' rule.

about A_i, i.e., $P(A_i)$, sample information relating B to the particular state of nature A_i through a probabilistic statement, $P(B \mid A_i)$, and the probability of observing that specific sample information, $P(B)$.

Let's consider a simple example, *the base rate fallacy*:

Assume that the sample information comes from a positive result from a test whose true positive rate (sensitivity) is 98%, i.e., $P(+ \mid \text{disease}) = 0.98$. In addition, the false positive rate is 2%, i.e., $P(+ \mid \neg\text{disease}) = 0.02$. On the other hand, the prior probability of being infected with the disease is given by the base incidence rate, $P(\text{disease}) = 0.002$. The question is: *What is the probability of actually being infected, given a positive test result?*

This is an example of *the base rate fallacy*, where a positive test result for a disease with a very low base incidence rate still gives a low probability of actually having the disease.

The key to answering this question lies in understanding the difference between the probability of having the disease given a positive test result, $P(\text{disease} \mid +)$, and the probability of a positive result given the disease, $P(+ \mid \text{disease})$. The former is the crucial result, and Bayes' rule helps us to compute it. Using Bayes' rule (equation 1.1):

$$\begin{aligned} P(\text{disease} \mid +) &= \frac{P(+ \mid \text{disease}) \times P(\text{disease})}{P(+)} \\ &= \frac{0.98 \times 0.002}{0.98 \times 0.002 + 0.02 \times (1 - 0.002)} \\ &= 0.09, \end{aligned}$$

where $P(+) = P(+ \mid \text{disease}) \times P(\text{disease}) + P(+ \mid \neg\text{disease}) \times P(\neg\text{disease})$.[2]

The following code shows how to perform this exercise in **R**.

R code. The base rate fallacy

```
PD <- 0.002 # Probability of disease
PPD <- 0.98 # True positive (Sensitivity)
PPF <- 0.02 # False positive
PDP <- PD * PPD / (PD * PPD + (1 - PD) * PPF)
paste("Probability of disease given a positive test is", sep
      = " ", round(PDP, 2))
"Probability of disease given a positive test is 0.09"
```

[2] $\neg$ is the negation symbol.

We observe that despite having a positive result, the probability of actually having the disease remains low. This is due to the base rate being so small.

Another interesting example—one that lies at the heart of the origin of Bayes' theorem [30]—concerns the existence of God [390]. In Section X of David Hume's *An Inquiry concerning Human Understanding* (1748), titled *Of Miracles*, Hume argues that when someone claims to have witnessed a miracle, the claim provides weak evidence that the event actually occurred, since it contradicts our everyday experience. In response, Richard Price, who edited and published *An Essay Towards Solving a Problem in the Doctrine of Chances* in 1763 (following Bayes' death in 1761), criticizes Hume's argument by distinguishing between *logical* and *physical* impossibility. He illustrates this with the example of a die with a million sides: while rolling a specific number might seem impossible, it is merely *improbable*, whereas rolling a number not present on the die is *physically impossible*. The former may occur with enough trials; the latter never will.

Note on the following example:
The next example is adapted from a modern blog post that illustrates the base rate fallacy using a resurrection scenario. While it is not taken from the original writings of Hume or Price, it reflects themes central to their philosophical debate on probability and miracles. It is included here to demonstrate how Bayes' rule behaves in cases involving extremely low prior probabilities. References to figures such as Jesus and Elvis are used purely for illustrative and pedagogical purposes. The goal is to explore the statistical reasoning, not to make any theological claims. Readers are encouraged to focus on the mathematical and conceptual content of the example.

To illustrate the statistical implications of this discussion, let us consider the following scenario.[3]

The scenario involves two reported cases of resurrection: Jesus Christ and Elvis Presley. The base rate is therefore two out of the total number of people who have ever lived, estimated at approximately 117 billion.[4] That is,

$$P(\text{Res}) = \frac{2}{117 \times 10^9}.$$

Suppose now that the sample information comes from a highly reliable witness, with a true positive rate of 0.9999999. Unlike the original blog post, we assume a more conservative false positive rate of 50%.

We ask: *What is the probability that a resurrection actually occurred, given that a witness claims it did?*

Using Bayes' rule, and letting *Res* denote the event of resurrection, and *Witness* the event of a witness declaring a resurrection:

[3] **https://blog.ephorie.de/base-rate-fallacy-or-why-no-one-is-justified-to-believe-that-jesus-rose**

[4] **https://www.prb.org/articles/how-many-people-have-ever-lived-on-earth/?utm_source=chatgpt.com**

$$
\begin{aligned}
P(\text{Res} \mid \text{Witness}) &= \frac{P(\text{Witness} \mid \text{Res}) \times P(\text{Res})}{P(\text{Witness})} \\
&= \frac{0.9999999 \times \frac{2}{117 \times 10^9}}{0.9999999 \times \frac{2}{117 \times 10^9} + 0.5 \times \left(1 - \frac{2}{117 \times 10^9}\right)} \\
&\approx 3.42 \times 10^{-11}
\end{aligned}
$$

Here, the denominator is the marginal probability of a witness reporting a resurrection:

$$P(\text{Witness}) = P(\text{Witness} \mid \text{Res}) \times P(\text{Res}) + P(\text{Witness} \mid \neg\text{Res}) \times (1 - P(\text{Res}))$$

Thus, the probability that a resurrection actually occurred, even when reported by an extremely reliable witness, is approximately

$$3.42 \times 10^{-11}.$$

This value is exceedingly small, but importantly, it is not zero. A value of zero would imply *impossibility*, whereas this result reflects *extremely low probability*.

The following code shows how to perform this exercise in **R**.

R code. Of Miracles

```
PR <- 2/(117 * 10^9) # Probability of resurrection
PWR <- 0.9999999 # True positive rate
PWNR <- 0.5 # False positive
PRW <- PR * PWR / (PR * PWR + (1 - PR) * PWNR) # Probability
     of resurrection given witness
paste("Probability of resurrection given witness is", sep =
    " ", PRW)
"Probability of resurrection given witness is
    3.41880307686464e-11"
```

Observe that we can condition on multiple events in Bayes' rule. Let's consider two conditioning events, B and C. Then, equation 1.1 becomes

$$
\begin{aligned}
P(A_i \mid B, C) &= \frac{P(A_i, B, C)}{P(B, C)} \\
&= \frac{P(B \mid A_i, C) \times P(A_i \mid C) \times P(C)}{P(B \mid C) P(C)}.
\end{aligned} \tag{1.2}
$$

Let's use this rule in one of the most intriguing statistical puzzles, *the*

FIGURE 1.1
The Monty Hall show.

Monty Hall problem, to illustrate how to use equation 1.2 [370, 102]. This was the situation faced by a contestant in the American television game show *Let's Make a Deal*. In this game, the contestant was asked to choose a door, behind one of which there is a car, and behind the others, goats.

Let's say the contestant picks door No. 1, and the host (Monty Hall), who knows what is behind each door, opens door No. 3, revealing a goat (see Figure 1.1). Then, the host asks the tricky question: *Do you want to pick door No. 2?*

Let's define the following events:

- P_i: the event **contestant picks door No.** i, which stays closed,
- H_i: the event **host picks door No.** i, which is open and contains a goat,
- C_i: the event **car is behind door No.** i.

In this particular setting, the contestant is interested in the probability of the event $P(C_2 \mid H_3, P_1)$. A naive answer would be that it is irrelevant, as initially, $P(C_i) = \frac{1}{3},\ i = 1, 2, 3$, and now $P(C_i \mid H_3) = \frac{1}{2},\ i = 1, 2$, since the host opened door No. 3. So, why bother changing the initial guess if the odds are the same (1:1)?

The important point here is that the host knows what is behind each door and always picks a door where there is a goat, given the contestant's choice. In this particular setting:

$$P(H_3 \mid C_3, P_1) = 0, \quad P(H_3 \mid C_2, P_1) = 1, \quad P(H_3 \mid C_1, P_1) = \frac{1}{2}.$$

Then, using equation 1.2, we can calculate the posterior probability.

$$\begin{aligned} P(C_2 \mid H_3, P_1) &= \frac{P(C_2, H_3, P_1)}{P(H_3, P_1)} \\ &= \frac{P(H_3 \mid C_2, P_1)P(C_2 \mid P_1)P(P_1)}{P(H_3 \mid P_1) \times P(P_1)} \\ &= \frac{P(H_3 \mid C_2, P_1)P(C_2)}{P(H_3 \mid P_1)} \\ &= \frac{1 \times 1/3}{1/2}, \end{aligned}$$

where the third equation uses the fact that C_i and P_i are independent events, and $P(H_3 \mid P_1) = \frac{1}{2}$ because this depends only on P_1 (not on C_2).

Therefore, changing the initial decision increases the probability of getting the car from $\frac{1}{3}$ to $\frac{2}{3}$! Thus, it is always a good idea to change the door.

Let's see a simulation exercise in **R** to check this answer:

R code. The Monty Hall problem

```
set.seed(0101) # Set simulation seed
S <- 100000 # Simulations
Game <- function(switch = 0){
  # switch = 0 is not change
  # switch = 1 is to change
  opts <- 1:3
  car <- sample(opts, 1) # car location
  guess1 <- sample(opts, 1) # Initial guess

  if(car != guess1) {
   host <- opts[-c(car, guess1)]
  } else {
   host <- sample(opts[-c(car, guess1)], 1)
  }
  win1 <- guess1 == car # Win no change
  guess2 <- opts[-c(host, guess1)]
  win2 <- guess2 == car # Win change
  if(switch == 0){
    win <- win1
  } else {
    win <- win2
  }
  return(win)
}

#Win probabilities not changing
Prob <- mean(replicate(S, Game(switch = 0)))
Prob
0.3334

#Win probabilities changing
Prob <- mean(replicate(S, Game(switch = 1)))
Prob
0.6654
```

1.2 Bayesian framework: A brief summary of theory

Given an unknown parameter set $\boldsymbol{\theta}$, and a particular realization of the data $\mathbf{y}$, Bayes' rule may be applied analogously,[5]

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})}{p(\mathbf{y})}, \tag{1.3}$$

where $\pi(\boldsymbol{\theta} \mid \mathbf{y})$ is the posterior density function, $\pi(\boldsymbol{\theta})$ is the prior density, $p(\mathbf{y} \mid \boldsymbol{\theta})$ is the likelihood (statistical model), and

$$p(\mathbf{y}) = \int_{\boldsymbol{\Theta}} p(\mathbf{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} = \mathbb{E}\left[p(\mathbf{y} \mid \boldsymbol{\theta})\right] \tag{1.4}$$

is the marginal likelihood or prior predictive. Observe that for this expected value to be meaningful, the prior should be a proper density, that is, it must integrate to one; otherwise, it does not make sense.

Observe that $p(\mathbf{y} \mid \boldsymbol{\theta})$ is not a density in $\boldsymbol{\theta}$. In addition, $\pi(\boldsymbol{\theta})$ does not have to integrate to 1, that is, $\pi(\boldsymbol{\theta})$ can be an improper density function, $\int_{\boldsymbol{\Theta}} \pi(\boldsymbol{\theta})d\boldsymbol{\theta} = \infty$. However, $\pi(\boldsymbol{\theta} \mid \mathbf{y})$ is a proper density function, that is, $\int_{\boldsymbol{\Theta}} \pi(\boldsymbol{\theta} \mid \mathbf{y})d\boldsymbol{\theta} = 1$.

For instance, set $\pi(\boldsymbol{\theta}) = c$, where c is a constant, then $\int_{\boldsymbol{\Theta}} c d\boldsymbol{\theta} = \infty$. However,

$$\int_{\boldsymbol{\Theta}} \pi(\boldsymbol{\theta} \mid \mathbf{y})d\boldsymbol{\theta} = \int_{\boldsymbol{\Theta}} \frac{p(\mathbf{y} \mid \boldsymbol{\theta}) \times c}{\int_{\boldsymbol{\Theta}} p(\mathbf{y} \mid \boldsymbol{\theta}) \times c\, d\boldsymbol{\theta}} d\boldsymbol{\theta} = 1$$

where c cancels out.

$\pi(\boldsymbol{\theta} \mid \mathbf{y})$ is a sample updated "probabilistic belief" version of $\pi(\boldsymbol{\theta})$, where $\pi(\boldsymbol{\theta})$ is a prior probabilistic belief which can be constructed from previous empirical work, theoretical foundations, expert knowledge, and/or mathematical convenience. This prior usually depends on parameters, which are named *hyperparameters*. In addition, the Bayesian approach implies using a probabilistic model about $\mathbf{Y}$ given $\boldsymbol{\theta}$, that is, $p(\mathbf{y} \mid \boldsymbol{\theta})$, where its integral over $\boldsymbol{\Theta}$, $p(\mathbf{y})$, is named *the model evidence* due to being a measure of model fit to the data.

Observe that the Bayesian inferential approach is conditional, that is, what can we learn about an unknown object $\boldsymbol{\theta}$ given that we already observed $\boldsymbol{Y} = \mathbf{y}$? The answer is also conditional on the probabilistic model, that is, $p(\mathbf{y} \mid \boldsymbol{\theta})$. So, what if we want to compare different models, say $\mathcal{M}_m$, $m = \{1, 2, \ldots, M\}$? Then, we should make explicit this in the Bayes' rule formulation:

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}, \mathcal{M}_m) = \frac{p(\mathbf{y} \mid \boldsymbol{\theta}, \mathcal{M}_m) \times \pi(\boldsymbol{\theta} \mid \mathcal{M}_m)}{p(\mathbf{y} \mid \mathcal{M}_m)}. \tag{1.5}$$

[5]From a Bayesian perspective, $\boldsymbol{\theta}$ is fixed but unknown. Then, it is treated as a random object despite the lack of variability (see Chapter 2).

The posterior model probability is

$$\pi(\mathcal{M}_m \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathcal{M}_m) \times \pi(\mathcal{M}_m)}{p(\mathbf{y})}, \tag{1.6}$$

where $p(\mathbf{y} \mid \mathcal{M}_m) = \int_{\boldsymbol{\Theta}} p(\mathbf{y} \mid \boldsymbol{\theta}, \mathcal{M}_m) \times \pi(\boldsymbol{\theta} \mid \mathcal{M}_m) d\boldsymbol{\theta}$ due to equation 1.5, and $\pi(\mathcal{M}_m)$ is the prior model probability.

Calculating $p(\mathbf{y})$ in equations 1.3 and 1.6 is very demanding in most of the realistic cases. Fortunately, it is not required when performing inference about $\boldsymbol{\theta}$ as this is integrated out from it. Then, all you need to know about the shape of $\boldsymbol{\theta}$ is in $p(\mathbf{y} \mid \boldsymbol{\theta}, \mathcal{M}_m) \times \pi(\boldsymbol{\theta} \mid \mathcal{M}_m)$, or without explicitly conditioning on $\mathcal{M}_m$,

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta}). \tag{1.7}$$

Equation 1.7 is a very good shortcut to perform Bayesian inference about $\boldsymbol{\theta}$.

We can also avoid calculating $p(\mathbf{y})$ when performing model selection (hypothesis testing) using the posterior odds ratio, that is, comparing models $\mathcal{M}_1$ and $\mathcal{M}_2$,

$$\begin{aligned} PO_{12} &= \frac{\pi(\mathcal{M}_1 \mid \mathbf{y})}{\pi(\mathcal{M}_2 \mid \mathbf{y})} \\ &= \frac{p(\mathbf{y} \mid \mathcal{M}_1)}{p(\mathbf{y} \mid \mathcal{M}_2)} \times \frac{\pi(\mathcal{M}_1)}{\pi(\mathcal{M}_2)}, \end{aligned} \tag{1.8}$$

where the first term in equation 1.8 is named the *Bayes factor*, and the second term is the *prior odds*. Observe that the Bayes factor is a ratio of ordinates for $\mathbf{y}$ under different models. Then, the Bayes factor is a measure of relative sample evidence in favor of model 1 compared to model 2.

However, we still need to calculate $p(\mathbf{y} \mid \mathcal{M}_m) = \int_{\boldsymbol{\Theta}} p(\mathbf{y} \mid \boldsymbol{\theta}, \mathcal{M}_m) \pi(\boldsymbol{\theta} \mid \mathcal{M}_m) d\boldsymbol{\theta} = \mathbb{E}\left[p(\mathbf{y} \mid \boldsymbol{\theta}, \mathcal{M}_m)\right]$. For this integral to be meaningful, the prior must be proper. Using an improper prior has unintended consequences when comparing models; for instance, parsimonious models are favored by posterior odds or Bayes factors, and these values may depend on units of measure (see Chapter 3).

A nice feature of comparing models using posterior odds is that if we have an exhaustive set of competing models such that $\sum_{m=1}^{M} \pi(\mathcal{M}_m \mid \mathbf{y}) = 1$, then we can recover $\pi(\mathcal{M}_m \mid \mathbf{y})$ without calculating $p(\mathbf{y})$. In particular, given two models $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\pi(\mathcal{M}_1 \mid \mathbf{y}) + \pi(\mathcal{M}_2 \mid \mathbf{y}) = 1$, we have:

$$\pi(\mathcal{M}_1 \mid \mathbf{y}) = \frac{PO_{12}}{1 + PO_{12}} \quad \text{and} \quad \pi(\mathcal{M}_2 \mid \mathbf{y}) = 1 - \pi(\mathcal{M}_1 \mid \mathbf{y}).$$

In general,

$$\pi(\mathcal{M}_m \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathcal{M}_m) \times \pi(\mathcal{M}_m)}{\sum_{l=1}^{M} p(\mathbf{y} \mid \mathcal{M}_l) \times \pi(\mathcal{M}_l)}.$$

TABLE 1.1
Kass and Raftery guidelines.

$2 \times \log(PO_{12})$	PO_{12}	**Evidence against $\mathcal{M}_2$**
0 to 2	1 to 3	Not worth more than a bare mention
2 to 6	3 to 20	Positive
6 to 10	20 to 150	Strong
> 10	> 150	Very strong

Notes: [226] proposed these guidelines for model selection using posterior odds in a Bayesian framework.

These posterior model probabilities can be used to perform Bayesian model averaging.

Table 1.1 shows guidelines for the interpretation of $2\log(PO_{12})$ [226]. This transformation is done to replicate the structure of the likelihood ratio test statistic. However, posterior odds do not require nested models as the likelihood ratio test does.

Observe that the posterior odds ratio is a relative criterion, that is, we specify an exhaustive set of competing models and compare them. However, we may want to check the performance of a model on its own or use a non-informative prior. In this case, we can use *the posterior predictive p-value* [148, 149].[6]

The intuition behind the predictive p-value is simple: analyze the discrepancy between the model's assumptions and the data by checking a potential extreme tail-area probability. Observe that this approach does not check if a model is true; its focus is on potential discrepancies between the model and the data at hand.

This is done by simulating pseudo-data from our sampling model ($\mathbf{y}^{(s)}, s = 1, 2, \ldots, S$) using draws from the posterior distribution, and then calculating a discrepancy measure, $D(\mathbf{y}^{(s)}, \boldsymbol{\theta})$, to estimate the posterior predictive p-value,

$$p_D(\mathbf{y}) = P[D(\mathbf{y}^{(s)}, \boldsymbol{\theta}) \geq D(\mathbf{y}, \boldsymbol{\theta})],$$

using the proportion of the S draws for which $D(\mathbf{y}^{(s)}, \boldsymbol{\theta}^{(s)}) \geq D(\mathbf{y}, \boldsymbol{\theta}^{(s)})$. Extreme tail probabilities ($p_D(\mathbf{y}) \leq 0.05$ or $p_D(\mathbf{y}) \geq 0.95$) suggest potential discrepancies between the data and the model. [149] also suggests the posterior predictive p-value based on the *minimum discrepancy*,

$$D_{\min}(\mathbf{y}) = \min_{\boldsymbol{\theta}} D(\mathbf{y}, \boldsymbol{\theta}),$$

and the *average discrepancy* statistic

$$D(\mathbf{y}) = \mathbb{E}[D(\mathbf{y}, \boldsymbol{\theta})] = \int_{\boldsymbol{\Theta}} D(\mathbf{y}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta} \mid \mathbf{y}) d\boldsymbol{\theta}.$$

[6] [27] show potential issues due to using data twice in the construction of the predictive p-values. They also present alternative proposals, for instance, *the partial posterior predictive p-value.*

These alternatives can be more computationally demanding.

The Bayesian approach is also suitable to get probabilistic predictions, that is, we can obtain a posterior predictive density

$$\begin{aligned}\pi(\mathbf{y}_0 \mid \mathbf{y}, \mathcal{M}_m) &= \int_{\boldsymbol{\Theta}} \pi(\mathbf{y}_0, \boldsymbol{\theta} \mid \mathbf{y}, \mathcal{M}_m) d\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\Theta}} \pi(\mathbf{y}_0 \mid \boldsymbol{\theta}, \mathbf{y}, \mathcal{M}_m)\pi(\boldsymbol{\theta} \mid \mathbf{y}, \mathcal{M}_m) d\boldsymbol{\theta}. \end{aligned} \tag{1.9}$$

Observe that equation 1.9 is again an expectation $\mathbb{E}[\pi(\mathbf{y}_0 \mid \boldsymbol{\theta}, \mathbf{y}, \mathcal{M}_m)]$, this time using the posterior distribution.[7] Therefore, the Bayesian approach takes estimation error into account when performing prediction.

As we have shown many times, expectation (integration) is a common feature in Bayesian inference. That is why the remarkable relevance of computation based on *Monte Carlo integration* in the Bayesian framework (see Chapter 4).

Bayesian model averaging (BMA) allows for considering model uncertainty in prediction or any unknown probabilistic object. In the case of the predictive density,

$$\pi(\mathbf{y}_0 \mid \mathbf{y}) = \sum_{m=1}^{M} \pi(\mathcal{M}_m \mid \mathbf{y})\pi(\mathbf{y}_0 \mid \mathbf{y}, \mathcal{M}_m). \tag{1.10}$$

In the case of the posterior density of the parameters,

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}) = \sum_{m=1}^{M} \pi(\mathcal{M}_m \mid \mathbf{y})\pi(\boldsymbol{\theta} \mid \mathbf{y}, \mathcal{M}_m), \tag{1.11}$$

where

$$\mathbb{E}(\boldsymbol{\theta} \mid \mathbf{y}) = \sum_{m=1}^{M} \hat{\boldsymbol{\theta}}_m \pi(\mathcal{M}_m \mid \mathbf{y}), \tag{1.12}$$

and

$$Var(\theta_k \mid \mathbf{y}) = \sum_{m=1}^{M} \pi(\mathcal{M}_m \mid \mathbf{y})\widehat{Var}(\theta_{km} \mid \mathbf{y}, \mathcal{M}_m) + \sum_{m=1}^{M} \pi(\mathcal{M}_m \mid \mathbf{y})(\hat{\theta}_{km} - \mathbb{E}[\theta_{km} \mid \mathbf{y}])^2, \tag{1.13}$$

$\hat{\boldsymbol{\theta}}_m$ is the posterior mean and $\widehat{Var}(\theta_{km} \mid \mathbf{y}, \mathcal{M}_m)$ is the posterior variance of the k-th element of $\boldsymbol{\theta}$ under model $\mathcal{M}_m$.

[7] Computing expectations is a fundamental aspect of Bayesian inference. See [274] for a comprehensive review of the evolution of computational methods for this task.

Observe how the variance in equation 1.13 captures the extra variability due to potential differences between the mean posterior estimates associated with each model, and the posterior mean that incorporates model uncertainty in equation 1.12.

A significant advantage of the Bayesian approach, which is particularly useful in *state space representations* (see Chapter 8), is the way the posterior distribution updates with new sample information. Given $\mathbf{y} = \mathbf{y}_{1:t+1}$ as a sequence of observations from 1 to $t+1$, then

$$\begin{aligned}\pi(\boldsymbol{\theta} \mid \mathbf{y}_{1:t+1}) &\propto p(\mathbf{y}_{1:t+1} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta}) \\ &= p(y_{t+1} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) \times p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta}) \\ &\propto p(y_{t+1} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta} \mid \mathbf{y}_{1:t}). \end{aligned} \tag{1.14}$$

We observe in Equation 1.14 that the new prior is simply the posterior distribution based on the previous observations. This is particularly useful under the assumption of *conditional independence*, that is, $Y_{t+1} \perp \mathbf{Y}_{1:t} \mid \boldsymbol{\theta}$, so that $p(y_{t+1} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) = p(y_{t+1} \mid \boldsymbol{\theta})$, allowing the posterior to be recovered recursively [309]. This facilitates online updating because all information up to time t is captured in $\boldsymbol{\theta}$. Therefore, $\pi(\boldsymbol{\theta} \mid \mathbf{y}_{1:t+1}) \propto p(y_{t+1} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta} \mid \mathbf{y}_{1:t}) \propto \prod_{h=1}^{t+1} p(y_h \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})$. This recursive expression can be computed more efficiently at any specific point in time t, compared to a batch-mode algorithm, which requires processing all information up to time t simultaneously.

It is also important to consider the sampling properties of "Bayesian estimators". This topic has attracted the attention of statisticians and econometricians for a long time. For instance, asymptotic posterior concentration on the population parameter vector is discussed by [41]. The convergence of posterior distributions is stated by the Bernstein-von Mises theorem [243, 404], which establishes a link between *credible intervals (sets)* and confidence intervals (sets), where a credible interval is an interval in the domain of the posterior distribution within which an unknown parameter falls with a particular probability. Credible intervals treat bounds as fixed and parameters as random, whereas confidence intervals reverse this. There are many settings in parametric models where Bayesian credible intervals with an α level converge asymptotically to confidence intervals at the α level. This suggests that Bayesian inference is asymptotically correct from a sampling perspective in these settings.

A heuristic approach to demonstrate this in the simplest case, where we assume random sampling and $\theta \in \mathcal{R}$, is the following: $p(\mathbf{y} \mid \theta) = \prod_{i=1}^{N} p(y_i \mid \theta)$, so the log likelihood is $l(\mathbf{y} \mid \theta) \equiv \log p(\mathbf{y} \mid \theta) = \sum_{i=1}^{N} \log p(y_i \mid \theta) = N \times \bar{l}(\mathbf{y} \mid \theta)$, where $\bar{l} \equiv \frac{1}{N} \sum_{i=1}^{N} \log p(y_i \mid \theta)$ is the mean likelihood.[8] Then, the posterior distribution is proportional to

[8]Note that in the likelihood function the argument is θ, but we keep the notation for convenience in exposition.

$$\begin{aligned}\pi(\theta \mid \mathbf{y}) &\propto p(\mathbf{y} \mid \theta) \times \pi(\theta) \\ &= \exp\left\{N \times \bar{l}(\mathbf{y} \mid \theta)\right\} \times \pi(\theta). \end{aligned} \tag{1.15}$$

Observe that as the sample size increases, that is, as $N \to \infty$, the exponential term should dominate the prior distribution as long as the prior does not depend on N, such that the likelihood determines the posterior distribution asymptotically.

Maximum likelihood theory shows that $\lim_{N\to\infty} \bar{l}(\mathbf{y} \mid \theta) \to \bar{l}(\mathbf{y} \mid \theta_0)$, where θ_0 is the population parameter of the data-generating process. In addition, performing a second-order Taylor expansion of the log likelihood at the maximum likelihood estimator,

$$\begin{aligned} l(\mathbf{y} \mid \theta) &\approx l(\mathbf{y} \mid \hat{\theta}) + \left.\frac{dl(\mathbf{y} \mid \theta)}{d\theta}\right|_{\hat{\theta}} (\theta - \hat{\theta}) + \frac{1}{2} \left.\frac{d^2 l(\mathbf{y} \mid \theta)}{d\theta^2}\right|_{\hat{\theta}} (\theta - \hat{\theta})^2 \\ &= l(\mathbf{y} \mid \hat{\theta}) + \frac{1}{2} \sum_{i=1}^{N} \left.\frac{d^2 l(y_i \mid \theta)}{d\theta^2}\right|_{\hat{\theta}} (\theta - \hat{\theta})^2 \\ &= l(\mathbf{y} \mid \hat{\theta}) - \frac{1}{2} N \left[-\bar{l}''\big|_{\hat{\theta}}\right] (\theta - \hat{\theta})^2 \\ &= l(\mathbf{y} \mid \hat{\theta}) - \frac{N}{2\sigma^2} (\theta - \hat{\theta})^2 \end{aligned}$$

where $\left.\frac{dl(\mathbf{y}|\theta)}{d\theta}\right|_{\hat{\theta}} = 0$, $\bar{l}'' \equiv \frac{1}{N} \sum_{i=1}^{N} \left.\frac{d^2 l(y_i|\theta)}{d\theta^2}\right|_{\hat{\theta}}$ and $\sigma^2 := \left[-\bar{l}''\big|_{\hat{\theta}}\right]^{-1}$.[9] Then,

$$\begin{aligned} \pi(\theta \mid \mathbf{y}) &\propto \exp\left\{l(\mathbf{y} \mid \theta)\right\} \times \pi(\theta) \\ &\approx \exp\left\{l(\mathbf{y} \mid \hat{\theta}) - \frac{N}{2\sigma^2}(\theta - \hat{\theta})^2\right\} \times \pi(\theta) \\ &\propto \exp\left\{-\frac{N}{2\sigma^2}(\theta - \hat{\theta})^2\right\} \times \pi(\theta) \end{aligned}$$

Observe that the posterior density is proportional to the kernel of a normal density with mean $\hat{\theta}$ and variance σ^2/N, as long as $\pi(\hat{\theta}) \neq 0$. This kernel dominates as the sample size increases due to the N in the exponential term. It is also important to note that the prior should not exclude values of θ that are logically possible, such as $\hat{\theta}$.

[9] The last definition follows from standard theory in maximum likelihood estimation (see [69, Chap. 10] and [420, Chap. 13]).

1.2.1 Example: Health insurance

Suppose that you are analyzing whether to buy health insurance next year. To make a better decision, you want to know *what is the probability that you will visit your doctor at least once next year?* To answer this question, you have records of the number of times you have visited your doctor over the last 5 years, $\mathbf{y} = \{0, 3, 2, 1, 0\}$. How should you proceed?

Assuming that this is a random sample[10] from a data-generating process (statistical model) that is Poisson, i.e., $Y_i \sim P(\lambda)$, and your probabilistic prior beliefs about λ are well described by a Gamma distribution with shape and scale parameters α_0 and β_0, i.e., $\lambda \sim G(\alpha_0, \beta_0)$, then you are interested in calculating the probability $P(Y_0 > 0 \mid \mathbf{y})$. To answer this, you need to calculate the posterior predictive density $\pi(y_0 \mid \mathbf{y})$ in a Bayesian way.

In this example, $p(\mathbf{y} \mid \lambda)$ is Poisson, and $\pi(\lambda)$ is Gamma. Therefore, using Equation 1.9.

$$\pi(y_0 \mid \mathbf{y}) = \int_0^\infty \frac{\lambda^{y_0} \exp\{-\lambda\}}{y_0!} \times \pi(\lambda \mid \mathbf{y}) d\lambda,$$

where the posterior distribution is $\pi(\lambda \mid \mathbf{y}) \propto \lambda^{\sum_{i=1}^N y_i + \alpha_0 - 1} \exp\left\{-\lambda\left(\frac{\beta_0 N+1}{\beta_0}\right)\right\}$ by equation 1.3.

Observe that the last expression is the kernel of a Gamma distribution with parameters $\alpha_n = \sum_{i=1}^N y_i + \alpha_0$ and $\beta_n = \frac{\beta_0}{\beta_0 N+1}$. Given that $\int_0^\infty \pi(\lambda \mid \mathbf{y}) d\lambda = 1$, the constant of proportionality in the last expression is $\Gamma(\alpha_n)\beta_n^{\alpha_n}$, where $\Gamma(\cdot)$ is the Gamma function. Thus, the posterior density function $\pi(\lambda \mid \mathbf{y})$ is $G(\alpha_n, \beta_n)$.

Observe that

$$\begin{aligned}
\mathbb{E}[\lambda \mid \mathbf{y}] &= \alpha_n \beta_n \\
&= \left(\sum_{i=1}^N y_i + \alpha_0\right)\left(\frac{\beta_0}{\beta_0 N + 1}\right) \\
&= \bar{y}\left(\frac{N\beta_0}{N\beta_0 + 1}\right) + \alpha_0\beta_0\left(\frac{1}{N\beta_0 + 1}\right) \\
&= w\bar{y} + (1 - w)\mathbb{E}[\lambda],
\end{aligned}$$

where $\bar{y}$ is the sample mean estimate, which is the maximum likelihood estimate of λ in this example, $w = \left(\frac{N\beta_0}{N\beta_0+1}\right)$, and $\mathbb{E}[\lambda] = \alpha_0\beta_0$ is the prior mean. The posterior mean is a weighted average of the maximum likelihood estimator (sample information) and the prior mean. Observe that $\lim_{N\to\infty} w = 1$, that is, the sample information asymptotically dominates.

[10] Independent and identically distributed draws.

The predictive distribution is

$$\begin{aligned}
\pi(y_0 \mid \mathbf{y}) =& \int_0^\infty \frac{\lambda^{y_0} \exp\{-\lambda\}}{y_0!} \times \frac{1}{\Gamma(\alpha_n)\beta_n^{\alpha_n}} \lambda^{\alpha_n - 1} \exp\{-\lambda/\beta_n\}\, d\lambda \\
=& \frac{1}{y_0!\Gamma(\alpha_n)\beta_n^{\alpha_n}} \int_0^\infty \lambda^{y_0+\alpha_n-1} \exp\left\{-\lambda\left(\frac{1+\beta_n}{\beta_n}\right)\right\} d\lambda \\
=& \frac{\Gamma(y_0+\alpha_n)\left(\frac{\beta_n}{\beta_n+1}\right)^{y_0+\alpha_n}}{y_0!\Gamma(\alpha_n)\beta_n^{\alpha_n}} \\
=& \binom{y_0+\alpha_n-1}{y_0}\left(\frac{\beta_n}{\beta_n+1}\right)^{y_0}\left(\frac{1}{\beta_n+1}\right)^{\alpha_n}.
\end{aligned}$$

The third equality follows from the kernel of a Gamma density, and the fourth from

$$\binom{y_0+\alpha_n-1}{y_0} = \frac{(y_0+\alpha_n-1)(y_0+\alpha_n-2)\ldots\alpha_n}{y_0!} = \frac{\Gamma(y_0+\alpha_n)}{\Gamma(\alpha_n)y_0!}$$

using a property of the Gamma function.

Observe that this is a Negative Binomial density, that is, $Y_0 \mid \mathbf{y} \sim \mathrm{NB}(\alpha_n, p_n)$ where $p_n = \frac{\beta_n}{\beta_n+1}$.

Up to this point, we have said nothing about the hyperparameters, which are required to give a concrete response to this exercise. Thus, we show two approaches to set them. First, we set $\alpha_0 = 0.001$ and $\beta_0 = \frac{1}{0.001}$, which imply vague prior information about λ due to having a large degree of variability compared to the mean information.[11] In particular, $\mathbb{E}[\lambda] = 1$ and $\mathbb{V}ar[\lambda] = 1000$.

In this setting, $P(Y_0 > 0 \mid \mathbf{y}) = 1 - P(Y_0 = 0 \mid \mathbf{y}) \approx 0.67$. That is, the probability of visiting the doctor at least once next year is approximately 0.67.

Another approach is using *Empirical Bayes*, where we set the hyperparameters maximizing the logarithm of the marginal likelihood,[12] that is,

$$\left[\hat{\alpha}_0 \ \hat{\beta}_0\right]^\top = \underset{\alpha_0,\beta_0}{\operatorname{argmax}} \ \ln p(\mathbf{y})$$

[11] We should be aware that there may be technical problems using this kind of hyperparameters in this setting [151].

[12] Empirical Bayes methods are criticized due to double-using the data. First to set the hyperparameters, and second, to perform Bayesian inference.

where

$$
\begin{aligned}
p(\mathbf{y}) &= \int_0^\infty \left\{ \frac{1}{\Gamma(\alpha_0)\beta_0^{\alpha_0}} \lambda^{\alpha_0 - 1} \exp\{-\lambda/\beta_0\} \prod_{i=1}^N \frac{\lambda^{y_i} \exp\{-\lambda\}}{y_i!} \right\} d\lambda \\
&= \frac{\int_0^\infty \lambda^{\sum_{i=1}^N y_i + \alpha_0 - 1} \exp\left\{-\lambda \left(\frac{\beta_0 N + 1}{\beta_0}\right)\right\} d\lambda}{\Gamma(\alpha_0)\beta_0^{\alpha_0} \prod_{i=1}^N y_i!} \\
&= \frac{\Gamma(\sum_{i=1}^N y_i + \alpha_0) \left(\frac{\beta_0}{N\beta_0 + 1}\right)^{\sum_{i=1}^N y_i} \left(\frac{1}{N\beta_0 + 1}\right)^{\alpha_0}}{\Gamma(\alpha_0) \prod_{i=1}^N y_i}
\end{aligned}
$$

Using the empirical Bayes approach, we get $\hat{\alpha}_0 = 51.8$ and $\hat{\beta}_0 = 0.023$, then $P(Y_0 > 0 \mid \mathbf{y}) = 1 - P(Y_0 = 0 \mid \mathbf{y}) \approx 0.70$.

Observe that we can calculate the posterior odds comparing the model using an Empirical Bayes prior (model 1) versus the vague prior (model 2). We assume that $\pi(\mathcal{M}_1) = \pi(\mathcal{M}_2) = 0.5$, then

$$
\begin{aligned}
PO_{12} &= \frac{p(\mathbf{y} \mid \text{Empirical Bayes})}{p(\mathbf{y} \mid \text{Vague prior})} \\
&= \frac{\frac{\Gamma(\sum_{i=1}^N y_i + 51.807)\left(\frac{0.023}{N \times 0.023 + 1}\right)^{\sum_{i=1}^N y_i}\left(\frac{1}{N \times 0.023 + 1}\right)^{51.807}}{\Gamma(51.807)}}{\frac{\Gamma(\sum_{i=1}^N y_i + 0.001)\left(\frac{1/0.001}{N/0.001 + 1}\right)^{\sum_{i=1}^N y_i}\left(\frac{1}{N/0.001 + 1}\right)^{0.001}}{\Gamma(0.001)}} \\
&\approx 919.
\end{aligned}
$$

Then, $2 \times \log(PO_{12}) = 13.64$, which provides very strong evidence against the vague prior model (see Table 1.1). In particular, $\pi(\text{Empirical Bayes} \mid \mathbf{y}) = \frac{919}{1+919} = 0.999$ and $\pi(\text{Vague prior} \mid \mathbf{y}) = 1 - 0.999 = 0.001$. These probabilities can be used to perform Bayesian model averaging (BMA). In particular,

$$
\begin{aligned}
\mathbb{E}(\lambda \mid \mathbf{y}) &= 1.2 \times 0.999 + 1.2 \times 0.001 = 1.2 \\
Var(\lambda \mid \mathbf{y}) &= 0.025 \times 0.999 + 0.24 \times 0.001 \\
&\quad + (1.2 - 1.2)^2 \times 0.999 + (1.2 - 1.2)^2 \times 0.001 = 0.025.
\end{aligned}
$$

The BMA predictive distribution is a mix of negative binomial distributions, that is, $Y_0 \mid \mathbf{y} \sim 0.999 \times NB(57.8, 0.02) + 0.001 \times NB(6.001, 0.17)$.

The following code shows how to perform this exercise in **R**.

R code. Health insurance, predictive distribution using vague hyperparameters

```
set.seed(010101)
y <- c(0, 3, 2, 1, 0) # Data
N <- length(y)
ProbBo <- function(y, a0, b0){
  N <- length(y)
  #sample size
  an <- a0 + sum(y)
  # Posterior shape parameter
  bn <- b0 / ((b0 * N) + 1)
  # Posterior scale parameter
  p <- bn / (bn + 1)
  # Probability negative binomial density
  Pr <- 1 - pnbinom(0, size=an,prob=(1 - p))
  # Probability of visiting the Doctor at least once next year
  # Observe that in R there is a slightly different parametrization.
  return(Pr)
}
# Using a vague prior:
a0 <- 0.001 # Prior shape parameter
b0 <- 1 / 0.001 # Prior scale parameter
PriMeanV <- a0 * b0 # Prior mean
PriVarV <- a0 * b0^2 # Prior variance
Pp <- ProbBo(y, a0 = 0.001, b0 = 1 / 0.001)
# This setting is vague prior information.
Pp
0.67
```

R code. Health insurance, predictive distribution using empirical Bayes

```
# Using Empirical Bayes
LogMgLik <- function(theta, y){
 N <- length(y)
 #sample size
 a0 <- theta[1]
 # prior shape hyperparameter
 b0 <- theta[2]
 # prior scale hyperparameter
 an <- sum(y) + a0
 # posterior shape parameter
 if(a0 <= 0 || b0 <= 0){
  #Avoiding negative values
  lnp <- -Inf
  }else{
  lnp <- lgamma(an) + sum(y)*log(b0/(N*b0+1)) - a0*log(N*b0
    +1) - lgamma(a0)
 }
 # log marginal likelihood
 return(-lnp)
}
theta0 <- c(0.01, 1/0.1)
# Initial values
control <- list(maxit = 1000)
# Number of iterations in optimization
EmpBay <- optim(theta0, LogMgLik, method = "BFGS", control =
     control, hessian = TRUE, y = y)
# Optimization
EmpBay$convergence
0
a0EB <- EmpBay$par[1]
# Prior shape using empirical Bayes
a0EB
51.81
b0EB <- EmpBay$par[2]
# Prior scale using empirical Bayes
b0EB
0.023
PriMeanEB <- a0EB * b0EB
# Prior mean
PriVarEB <- a0EB * b0EB^2
# Prior variance
PpEB <- ProbBo(y, a0 = a0EB, b0 = b0EB)
# This setting is using emprical Bayes.
PpEB
0.70
```

R code. Health insurance, density plots

```r
# Density figures:
# This code helps plotting densities
lambda <- seq(0.01, 10, 0.01)
# Values of lambda
VaguePrior <- dgamma(lambda,shape=a0,scale = b0)
EBPrior <- dgamma(lambda,shape=a0EB,scale = b0EB)
PosteriorV <- dgamma(lambda, shape = a0 + sum(y), scale = b0 / ((b0 * N) + 1))
PosteriorEB <- dgamma(lambda, shape = a0EB+sum(y), scale = b0EB / ((b0EB * N) + 1))
# Likelihood function
Likelihood <- function(theta, y){
 LogL <- dpois(y, theta, log = TRUE)
 Lik <- prod(exp(LogL))
 return(Lik)
}
Liks <- sapply(lambda, function(par) {Likelihood(par, y = y)})
Sc <- max(PosteriorEB)/max(Liks)
#Scale for displaying in figure
LiksScale <- Liks * Sc
data <- data.frame(cbind(lambda, VaguePrior, EBPrior, PosteriorV, PosteriorEB, LiksScale)) #Data frame
require(ggplot2) # Cool figures
require(latex2exp) # LaTeX equations in figures
require(ggpubr) # Multiple figures in one page
fig1 <- ggplot(data = data, aes(lambda, VaguePrior)) + geom_line() +  xlab(TeX("$\\lambda$")) + ylab("Density") + ggtitle("Prior: Vague Gamma")
fig2 <- ggplot(data = data, aes(lambda, EBPrior)) + geom_line() + xlab(TeX("$\\lambda$")) + ylab("Density") + ggtitle("Prior: Empirical Bayes Gamma")
fig3 <- ggplot(data = data, aes(lambda, PosteriorV)) + geom_line() + xlab(TeX("$\\lambda$")) + ylab("Density") + ggtitle("Posterior: Vague Gamma")
fig4 <- ggplot(data = data, aes(lambda, PosteriorEB)) + geom_line() + xlab(TeX("$\\lambda$")) + ylab("Density") + ggtitle("Posterior: Empirical Bayes Gamma")
FIG <- ggarrange(fig1, fig2, fig3, fig4, ncol = 2, nrow = 2)
annotate_figure(FIG, top = text_grob("Vague versus Empirical Bayes: Poisson-Gamma model", color = "black", face = "bold", size = 14))
dataNew <- data.frame(cbind(rep(lambda, 3), c(EBPrior, PosteriorEB, LiksScale), rep(1:3, each = 1000))) #Data frame
colnames(dataNew) <- c("Lambda", "Density", "Factor")
dataNew$Factor <- factor(dataNew$Factor, levels=c("1", "3", "2"),
labels=c("Prior", "Likelihood", "Posterior"))
ggplot(data = dataNew, aes_string(x = "Lambda", y = "Density", group = "Factor")) + geom_line(aes(color = Factor)) + xlab(TeX("$\\lambda$")) + ylab("Density") + ggtitle("Prior, likelihood and posterior: Empirical Bayes Poisson-Gamma model") + guides(color=guide_legend(title="Information")) + scale_color_manual(values = c("red", "yellow", "blue"))
```

FIGURE 1.2
Vague versus Empirical Bayes: Poisson-Gamma model.

FIGURE 1.3
Prior, likelihood and posterior: Empirical Bayes Poisson-Gamma model.

Figure 1.2 displays the prior and posterior densities based on vague and Empirical Bayes hyperparameters. We observe that the prior and posterior densities using the latter are more informative, as expected.

Figure 1.3 shows the prior, scaled likelihood, and posterior densities of λ based on the hyperparameters from the Empirical Bayes approach. The posterior density is a compromise between prior and sample information.

R code. Health insurance, Predictive density

```
# Predictive distributions
PredDen <- function(y, y0, a0, b0){
  N <- length(y)
  #sample size
  an <- a0 + sum(y)
  # Posterior shape parameter
  bn <- b0 / ((b0 * N) + 1)
  # Posterior scale parameter
  p <- bn / (bn + 1)
  # Probability negative binomial density
  Pr <- dnbinom(y0, size=an, prob=(1 - p))
  # Predictive density
  # Observe that in R there is a slightly different parametrization.
  return(Pr)
}
y0 <- 0:10
PredVague <- PredDen(y=y, y0=y0, a0=a0, b0=b0)
PredEB <- PredDen(y=y, y0=y0, a0=a0EB, b0=b0EB)
dataPred <- as.data.frame(cbind(y0, PredVague, PredEB))
colnames(dataPred) <- c("y0", "PredictiveVague", "PredictiveEB")
ggplot(data = dataPred) + geom_point(aes(y0, PredictiveVague, color = "red")) +
xlab(TeX("$y_0$")) + ylab("Density") + ggtitle("Predictive density: Vague and Empirical Bayes priors") + geom_point(aes(y0, PredictiveEB, color = "yellow")) +
guides(color = guide_legend(title="Prior")) + scale_color_manual(labels = c("Vague", "Empirical Bayes"), values = c("red", "yellow")) + scale_x_continuous(breaks=seq(0,10,by=1))
```

Figure 1.4 displays the predictive probability mass of not having any visits to a physician next year, as well as having one, two, and so on, using Empirical Bayes and vague hyperparameters. The predictive probabilities of not having any visits are approximately 30% and 33% based on the Empirical Bayes and vague hyperparameters, respectively.

FIGURE 1.4
Predictive density: Vague and Empirical Bayes.

R code. Health insurance, Bayesian model average

```
# Posterior odds: Vague vs Empirical Bayes
PO12 <- exp(-LogMgLik(c(a0EB, b0EB), y = y))/exp(-LogMgLik(c
    (a0, b0), y = y))
PO12
919
PostProMEM <- PO12/(1 + PO12)
PostProMEM
0.998
# Posterior model probability Empirical Bayes
PostProbMV <- 1 - PostProMEM
PostProbMV
0.002
# Posterior model probability vague prior
# Bayesian model average (BMA)
PostMeanEB <- (a0EB + sum(y)) * (b0EB / (b0EB * N + 1))
# Posterior mean Empirical Bayes
PostMeanV <- (a0 + sum(y)) * (b0 / (b0 * N + 1))
# Posterior mean vague priors
BMAmean <- PostProMEM * PostMeanEB + PostProbMV * PostMeanV
BMAmean
1.2
# BMA posterior mean
PostVarEB <- (a0EB + sum(y)) * (b0EB/(b0EB * N + 1))^2
# Posterior variance Empirical Bayes
PostVarV <- (a0 + sum(y)) * (b0 / (b0 * N + 1))^2
# Posterior variance vague prior
BMAVar <- PostProMEM * PostVarEB + PostProbMV*PostVarV +
    PostProMEM * (PostMeanEB - BMAmean)^2 + PostProbMV * (
    PostMeanV - BMAmean)^2
# BMA posterior variance
BMAVar
0.025
```

R code. Health insurance, Bayesian model average

```
# BMA: Predictive
BMAPred <- PostProMEM * PredEB+PostProbMV * PredVague
dataPredBMA <- as.data.frame(cbind(y0, BMAPred))
colnames(dataPredBMA) <- c("y0", "PredictiveBMA")
ggplot(data = dataPredBMA) + geom_point(aes(y0,
    PredictiveBMA, color = "red")) +  xlab(TeX("$y_0$")) +
    ylab("Density") + ggtitle("Predictive density: BMA") +
    guides(color = guide_legend(title="BMA")) + scale_color_
    manual(labels = c("Probability"), values = c("red")) +
    scale_x_continuous(breaks=seq(0,10,by=1))

```

FIGURE 1.5
Bayesian model average: Predictive density.

Figure 1.5 displays the predictive density using Bayesian model averaging based on the vague and Empirical Bayes hyperparameters. This figure closely resembles the predictive probability mass function based on the Empirical Bayes framework, as the posterior model probability for that setting is nearly one.

Figure 1.6 shows how the posterior distribution updates with new sample information, starting from an initial non-informative prior (iteration 1). We

FIGURE 1.6
Bayesian updating: Posterior densities.

observe that iteration 5 incorporates all the sample information in our example. As a result, the posterior density in iteration 5 is identical to the posterior density shown in Figure 1.3.

R code. Health insurance, Bayes updating

```
# Bayesian updating
BayUp <- function(y, lambda, a0, b0){
  N <- length(y)
  #sample size
  an <- a0 + sum(y)
  # Posterior shape parameter
  bn <- b0 / ((b0 * N) + 1)
  # Posterior scale parameter
  p <- dgamma(lambda, shape = an, scale = bn)
  # Posterior density
  return(list(Post = p, a0New = an, b0New = bn))
}
PostUp <- NULL
for(i in 1:N){
  if(i == 1){
    PostUpi <- BayUp(y[i], lambda, a0 = 0.001, b0 = 1/0.001)
    }
  else{
    PostUpi <- BayUp(y[i], lambda, a0 = PostUpi$a0New, b0 =
    PostUpi$b0New)
  }
  PostUp <- cbind(PostUp, PostUpi$Post)
}
DataUp <- data.frame(cbind(rep(lambda, 5), c(PostUp), rep
    (1:5, each = 1000))) #Data frame
colnames(DataUp) <- c("Lambda", "Density", "Factor")
DataUp$Factor <- factor(DataUp$Factor, levels=c("1", "2", "3
    ", "4", "5"),
labels=c("Iter 1", "Iter 2", "Iter 3", "Iter 4", "Iter 5"))
ggplot(data = DataUp, aes_string(x = "Lambda", y = "Density"
    , group = "Factor")) + geom_line(aes(color = Factor)) +
    xlab(TeX("$\\lambda$")) + ylab("Density") + ggtitle("
    Bayesian updating: Poisson-Gamma model with vague prior"
    ) + guides(color=guide_legend(title="Update")) + scale_
    color_manual(values = c("red", "purple", "blue", "yellow
    ", "black"))
S <- 100000 # Posterior draws
PostMeanLambdaUps <- sapply(1:N, function(i) {mean(sample(
    lambda, S, replace = TRUE, prob = PostUp[ , i]))}) #
    Posterior mean update i
paste("Posterior means using all information and sequential
    updating are:", round(PostMeanV, 2), "and", round(
    PostMeanLambdaUps[5], 2), sep = " ")
Posterior means using all information and sequential
    updating are: 1.2 and 1.2
PostVarLambdaUps <- sapply(1:N, function(i) {var(sample(
    lambda, S, replace = TRUE, prob = PostUp[ , i]))}) #
    Posterior variance update i
paste("Posterior variances using all information and
    sequential updating are:", round(PostVarV, 2), "and",
    round(PostVarLambdaUps[5], 2), sep = " ")
Posterior variances using all information and sequential
    updating are: 0.24 and 0.24
```

1.3 Bayesian reports: Decision theory under uncertainty

The Bayesian framework allows reporting the full posterior distributions. However, some situations require reporting a specific value of the posterior distribution (point estimate), an informative interval (set), point or interval predictions, and/or selecting a specific model. Decision theory offers an elegant framework to make decisions regarding the optimal posterior values to report [36].

The starting point is a *loss function*, which is a non-negative real-valued function whose arguments are the unknown *state of nature* ($\boldsymbol{\Theta}$), and a set of *actions* to be taken ($\mathcal{A}$), that is,

$$L(\boldsymbol{\theta}, a) : \boldsymbol{\Theta} \times \mathcal{A} \to \mathcal{R}^+.$$

This function is a mathematical representation of the loss incurred from making mistakes. In particular, selecting action $a \in \mathcal{A}$ when $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is the true state. In our case, the unknown state of nature can refer to parameters, functions of them, future or unknown realizations, models, etc.

From a Bayesian perspective, we should choose the action that minimizes the posterior expected loss ($a^*(\mathbf{y})$), that is, the *posterior risk function* ($\mathbb{E}[L(\boldsymbol{\theta}, a) \mid \mathbf{y}]$),

$$a^*(\mathbf{y}) = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\ \mathbb{E}[L(\boldsymbol{\theta}, a) \mid \mathbf{y}],$$

where $\mathbb{E}[L(\boldsymbol{\theta}, a) \mid \mathbf{y}] = \int_{\boldsymbol{\Theta}} L(\boldsymbol{\theta}, a)\pi(\boldsymbol{\theta} \mid \mathbf{y})d\boldsymbol{\theta}$.[13]

Different loss functions imply different optimal decisions. We illustrate this assuming $\theta \in \mathcal{R}$.

- The quadratic loss function, $L(\theta, a) = [\theta - a]^2$, gives as the optimal decision the posterior mean, $a^*(\mathbf{y}) = \mathbb{E}[\theta \mid \mathbf{y}]$, that is:

  $$\mathbb{E}[\theta \mid \mathbf{y}] = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \int_{\Theta} [\theta - a]^2 \pi(\theta \mid \mathbf{y}) d\theta.$$

  To obtain this result, let's use the first-order condition, differentiate the risk function with respect to a, interchange the differential and integral order, and set the result equal to zero:

  $$-2 \int_{\Theta} [\theta - a^*]\pi(\theta \mid \mathbf{y}) d\theta = 0.$$

[13] [75] propose Laplace-type estimators (LTE) based on the *quasi-posterior*, $p(\boldsymbol{\theta}) = \frac{\exp\{L_n(\boldsymbol{\theta})\}\pi(\boldsymbol{\theta})}{\int_{\boldsymbol{\Theta}} \exp\{L_n(\boldsymbol{\theta})\}\pi(\boldsymbol{\theta})d\theta}$, where $L_n(\boldsymbol{\theta})$ is not necessarily a log-likelihood function. The LTE minimizes the *quasi-posterior risk*.

This implies that

$$a^* \int_\Theta \pi(\theta \mid \mathbf{y}) d\theta = a^*(\mathbf{y}) = \int_\Theta \theta \pi(\theta \mid \mathbf{y}) d\theta = \mathbb{E}[\theta \mid \mathbf{y}],$$

that is, the posterior mean is the Bayesian optimal action. This means that we should report the posterior mean as a point estimate of θ when facing the quadratic loss function.

- The generalized quadratic loss function, $L(\theta, a) = w(\theta)[\theta - a]^2$, where $w(\theta) > 0$ is a weighting function, gives as the optimal decision rule the weighted mean. We should follow the same steps as the previous result to obtain

$$a^*(\mathbf{y}) = \frac{\mathbb{E}[w(\theta) \times \theta \mid \mathbf{y}]}{\mathbb{E}[w(\theta) \mid \mathbf{y}]}.$$

  Observe that the weighted average is driven by the weighting function $w(\theta)$.

- The absolute error loss function, $L(\theta, a) = |\theta - a|$, gives as the optimal action the posterior median (Exercise 5).

- The generalized absolute error function,

$$L(\theta, a) = \begin{cases} K_0(\theta - a), & \theta - a \geq 0, \\ K_1(a - \theta), & \theta - a < 0, \end{cases} \quad K_0, K_1 > 0,$$

  implies the following risk function:

$$\mathbb{E}[L(\theta, a) \mid \mathbf{y}] = \int_{-\infty}^{a} K_1(a - \theta) \pi(\theta \mid \mathbf{y}) d\theta + \int_{a}^{\infty} K_0(\theta - a) \pi(\theta \mid \mathbf{y}) d\theta.$$

  Differentiating with respect to a, interchanging differentials and integrals, and equating to zero, we get:

$$K_1 \int_{-\infty}^{a^*} \pi(\theta \mid \mathbf{y}) d\theta - K_0 \int_{a^*}^{\infty} \pi(\theta \mid \mathbf{y}) d\theta = 0.$$

  Thus, we have

$$\int_{-\infty}^{a^*} \pi(\theta \mid \mathbf{y}) d\theta = \frac{K_0}{K_0 + K_1},$$

  that is, any $\frac{K_0}{K_0+K_1}$-percentile of $\pi(\theta \mid \mathbf{y})$ is an optimal Bayesian estimate of θ.

We can also use decision theory under uncertainty in hypothesis testing. In particular, testing $H_0 : \theta \in \Theta_0$ versus $H_1 : \theta \in \Theta_1$, where $\Theta = \Theta_0 \cup \Theta_1$ and $\emptyset = \Theta_0 \cap \Theta_1$, there are two actions of interest, a_0 and a_1, where a_j denotes not rejecting H_j, for $j = \{0, 1\}$.

Given the $0 - K_j$ loss function:

$$L(\theta, a_j) = \begin{cases} 0, & \text{if } \theta \in \Theta_j, \\ K_j, & \text{if } \theta \in \Theta_i, j \neq i, \end{cases}$$

where there is no loss if the right decision is made, for instance, not rejecting H_0 when $\theta \in \Theta_0$, and the loss is K_j when an error is made. For example, a type I error occurs when rejecting the null hypothesis (H_0) when it is true ($\theta \in \Theta_0$), which results in a loss of K_1 due to choosing action a_1, not rejecting H_1.

The posterior expected loss associated with decision a_j, i.e., not rejecting H_j, is:

$$\mathbb{E}[L(\theta, a_j) \mid \mathbf{y}] = 0 \times P(\Theta_j \mid \mathbf{y}) + K_j P(\Theta_i \mid \mathbf{y}) = K_j P(\Theta_i \mid \mathbf{y}), \quad j \neq i.$$

Therefore, the Bayes optimal decision is the one that minimizes the posterior expected loss. That is, the null hypothesis is rejected (a_1 is not rejected) when

$$K_0 P(\Theta_1 \mid \mathbf{y}) > K_1 P(\Theta_0 \mid \mathbf{y}).$$

Given our framework, $\Theta = \Theta_0 \cup \Theta_1$ and $\emptyset = \Theta_0 \cap \Theta_1$, we have $P(\Theta_0 \mid \mathbf{y}) = 1 - P(\Theta_1 \mid \mathbf{y})$. As a result, the rejection region of the Bayesian test is:

$$R = \left\{ \mathbf{y} : P(\Theta_1 \mid \mathbf{y}) > \frac{K_1}{K_1 + K_0} \right\}.$$

Decision theory also helps to construct interval (region) estimates. Let $\Theta_{C(\mathbf{y})} \subset \Theta$ be a *credible set* for θ, and let the loss function be defined as:

$$L(\theta, \Theta_{C(\mathbf{y})}) = 1 - \mathbb{1}\left\{\theta \in \Theta_{C(\mathbf{y})}\right\},$$

where

$$\mathbb{1}\left\{\theta \in \Theta_{C(\mathbf{y})}\right\} = \begin{cases} 1, & \text{if } \theta \in \Theta_{C(\mathbf{y})}, \\ 0, & \text{if } \theta \notin \Theta_{C(\mathbf{y})}. \end{cases}$$

Thus, the loss function becomes:

$$L(\theta, \Theta_{C(\mathbf{y})}) = \begin{cases} 0, & \text{if } \theta \in \Theta_{C(\mathbf{y})}, \\ 1, & \text{if } \theta \notin \Theta_{C(\mathbf{y})}. \end{cases}$$

This is a 0–1 loss function, which equals zero when $\theta \in \Theta_{C(\mathbf{y})}$ and equals one when $\theta \notin \Theta_{C(\mathbf{y})}$. Consequently, the risk function is:

$$1 - P(\theta \in \Theta_{C(\mathbf{y})}).$$

Given a *measure of credibility* $\alpha(\mathbf{y})$ that defines the level of trust that $\theta \in \Theta_{C(\mathbf{y})}$, we can measure the accuracy of the report by the loss function:

$$L(\theta, \alpha(\mathbf{y})) = \left[\mathbb{1}\left\{\theta \in \Theta_{C(\mathbf{y})}\right\} - \alpha(\mathbf{y})\right]^2.$$

This loss function could be used to suggest a choice of the report $\alpha(\mathbf{y})$. Given that this is a quadratic loss function, the optimal action is the posterior mean, that is,

$$\mathbb{E}[\mathbb{1}\left\{\theta \in \Theta_{C(\mathbf{y})}\right\} \mid \mathbf{y}] = P(\theta \in \Theta_{C(\mathbf{y})} \mid \mathbf{y}).$$

This probability can be calculated given the posterior distribution as

$$P(\theta \in \Theta_{C(\mathbf{y})} \mid \mathbf{y}) = \int_{\Theta_{C(\mathbf{y})}} \pi(\theta \mid \mathbf{y}) d\theta.$$

This represents a measure of the belief that $\theta \in \Theta_{C(\mathbf{y})}$ given the prior beliefs and sample information.

The set $\Theta_{C(\mathbf{y})} \subset \Theta$ is a $100(1-\alpha)\%$ credible set with respect to $\pi(\theta \mid \mathbf{y})$ if

$$P(\theta \in \Theta_{C(\mathbf{y})} \mid \mathbf{y}) = \int_{\Theta_{C(\mathbf{y})}} \pi(\theta \mid \mathbf{y}) d\theta = 1 - \alpha.$$

Two alternatives for reporting credible sets are the *symmetric credible set* and the *highest posterior density set* (HPD). The former is based on the $\frac{\alpha}{2}\%$ and $(1 - \frac{\alpha}{2})\%$ percentiles of the posterior distribution, and the latter is a $100(1-\alpha)\%$ credible interval for θ with the property that it has the smallest distance compared to any other $100(1-\alpha)\%$ credible interval for θ based on the posterior distribution. Specifically,

$$C(\mathbf{y}) = \{\theta : \pi(\theta \mid \mathbf{y}) \geq k(\alpha)\},$$

where $k(\alpha)$ is the largest number such that

$$\int_{\theta : \pi(\theta \mid \mathbf{y}) \geq k(\alpha)} \pi(\theta \mid \mathbf{y}) d\theta = 1 - \alpha.$$

The HPD set can be a collection of disjoint intervals when working with multimodal posterior densities. Additionally, HPD sets have the limitation of not necessarily being invariant under transformations.

Decision theory can also be used to perform prediction (point, sets, or probabilistic). Suppose that there is a loss function $L(Y_0, a)$ involving the prediction of Y_0. Then, the expected loss is

$$\mathbb{E}_{Y_0}[L(Y_0, a)] = \int_{\mathcal{Y}_0} L(y_0, a) \pi(y_0 \mid \mathbf{y})\, dy_0,$$

where $\pi(y_0 \mid \mathbf{y})$ is the predictive density function. Thus, we make an

optimal choice for prediction that minimizes the risk function given a specific loss function.

Although Bayesian Model Averaging (BMA) allows for incorporating model uncertainty in a regression framework, sometimes it is desirable to select just one model. A compelling alternative is to choose the model with the highest posterior model probability. This model is the best alternative for prediction in the case of a 0-1 loss function [95].

1.3.1 Example: Health insurance continues

We show some optimal rules in the health insurance example, specifically the best point estimates of λ under the quadratic, absolute, and generalized absolute loss functions. For the generalized absolute loss function, we assume that underestimating λ is twice as costly as overestimating it, i.e., $K_0 = 2$ and $K_1 = 1$.

Given that the posterior distribution of λ is $G(\alpha_0 + \sum_{i=1}^{N} y_i, \frac{\beta_0}{\beta_0 N+1})$, and using the hyperparameters from empirical Bayes, we obtain the following optimal point estimates:

- The posterior mean: $\mathbb{E}[\lambda \mid \mathbf{y}] = \alpha_n \beta_n = 1.2$,
- The posterior median: 1.19,
- The 2/3-th quantile: 1.26.

These are the optimal point estimates for the quadratic, absolute, and generalized absolute loss functions, respectively.

In addition, we test the null hypothesis $H_0 : \lambda \in [0, 1)$ versus the alternative hypothesis $H_1 : \lambda \in [1, \infty)$, setting $K_0 = K_1 = 1$. We should reject the null hypothesis since $P(\lambda \in [1, \infty)) = 0.9 > \frac{K_1}{K_0+K_1} = 0.5$.

The 95% symmetric credible interval is $(0.91, 1.53)$, and the highest posterior density (HPD) interval is $(0.90, 1.51)$. Finally, the optimal point prediction under the quadratic loss function is 1.2, which is the mean value of the posterior predictive distribution. The optimal model, assuming a 0-1 loss function, is the model using the hyperparameters from the empirical Bayes procedure, since the posterior model probability of this model is approximately 1, whereas the posterior model probability of the model using vague hyperparameters is approximately 0.

R code. Health insurance, Bayesian reports

```
an <- sum(y) + a0EB
# Posterior shape parameter
bn <- b0EB / (N*b0EB + 1)
# Posterior scale parameter
S <- 1000000
# Number of posterior draws
Draws <- rgamma(1000000, shape = an, scale = bn)
# Posterior draws
###### Point estimation ########
OptQua <- an*bn
# Mean: Optimal choice quadratic loss function
OptQua
1.200952
OptAbs <- qgamma(0.5, shape = an, scale = bn)
# Median: Optimal choice absolute loss function
OptAbs
1.194034
# Setting K0 = 2 and K1 = 1, that is, to underestimate
    lambda is twice as costly as to overestimate it.
K0 <- 2; K1 <- 1
OptGenAbs <- quantile(Draws, K0/(K0 + K1))
# Median: Optimal choice generalized absolute loss function
OptGenAbs
66.66667%
1.262986
###### Hypothesis test ########
# H0: lambda in [0,1) vs H1: lambda in [1, Inf]
K0 <- 1; K1 <- 1
ProbH0 <- pgamma(1, shape = an, scale = bn)
ProbH0 # Posterior  probability H0
0.09569011
ProbH1 <- 1 -ProbH0
ProbH1 # Posterior  probability H1
0.9043099
# We should reject H0 given ProbH1 > K1 / (K0 + K1)
###### Credible intervals ########
LimInf <- qgamma(0.025, shape = an, scale = bn) # Lower
    bound
LimInf
0.9114851
LimSup <- qgamma(0.975, shape = an, scale = bn) # Upper
    bound
LimSup
1.529724
HDI <- HDInterval::hdi(Draws, credMass = 0.95) # Highest
    posterior density credible interval
HDI
    lower     upper
0.8971505 1.5125911
attr(,"credMass")
[1] 0.95
###### Predictive optimal choices ########
p <- bn / (bn + 1) # Probability negative binomial density
OptPred <- p/(1-p)*an # Optimal point prediction given a
    quadratic loss function in prediction
OptPred
1.200952
```

1.4 Summary

We introduce Bayes' rule to update probabilistic statements using humorous examples. We then study the three key probabilistic objects in Bayesian inference: the posterior distribution, the marginal likelihood, and the predictive density. The posterior distribution allows for inference regarding parameters, the marginal likelihood is required for hypothesis testing and model selection using the Bayes factor, and the predictive density enables probabilistic predictions. We also review some sampling properties of Bayesian estimators and the process of Bayes updating. All of these concepts were illustrated using a simple example in **R** software. Finally, we introduce decision theory concepts that can be applied to report summary statistics while minimizing posterior expected losses.

1.5 Exercises

1. *The Court Case: The Blue or Green Cab*

   A cab was involved in a hit-and-run accident at night. There are two cab companies in the town: Blue and Green. The Blue company has 150 cabs, while the Green company has 850 cabs. A witness stated that a blue cab was involved in the accident. The court tested the reliability of the witness under similar circumstances and found that the witness correctly identified the color of the cab 80% of the time, but made an incorrect identification 25% of the time. *What is the probability that the cab involved in the accident was actually blue, given that the witness said it was blue?*

2. *The Monty Hall Problem*

   What is the probability of winning a car in the *Monty Hall problem* if you switch your decision, when there are four doors, three goats, and one car? Solve this problem both analytically and computationally. What if there are n doors, $n-1$ goats, and one car?

3. Solve the health insurance example using a Gamma prior in the rate parametrization, that is, $\pi(\lambda) = \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \lambda^{\alpha_0 - 1} \exp\{-\lambda\beta_0\}$.

4. Suppose you are analyzing the decision to buy car insurance for the next year. To make a better decision, you want to know: *What is the probability that you will have a car claim next year?* You have the records of your car claims over the last 15 years, $\mathbf{y} = \{0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0\}$.

   Assume that this is a random sample from a data-generating process

(statistical model) that is Bernoulli, $Y_i \sim \text{Ber}(p)$. Your prior beliefs about p are well described by a Beta distribution with parameters α_0 and β_0, i.e., $p \sim B(\alpha_0, \beta_0)$. You are interested in calculating the probability of a claim the next year, $P(Y_0 = 1 \mid \mathbf{y})$.

Solve this using both an empirical Bayes approach and a non-informative approach where $\alpha_0 = \beta_0 = 1$ (uniform distribution).

5. Show that, given the loss function $L(\theta, a) = |\theta - a|$, the optimal decision rule minimizing the risk function, $a^*(\mathbf{y})$, is the median.

2

Conceptual differences between the Bayesian and Frequentist approaches

We outline some of the conceptual differences between the Bayesian and Frequentist inferential approaches. We emphasize Bayesian concepts, as most readers may already be familiar with the Frequentist statistical framework. We illustrate the differences between these two inferential approaches using a simple example. In addition, we provide some potential explanations for why the Bayesian inferential framework is not well known at the introductory level among practitioners and applied researchers.

2.1 The concept of probability

Let's begin with the following thought experiment: Assume that you are watching the international game show "Who Wants to Be a Millionaire?". The contestant is asked to answer a very simple question: **What is the last name of the brothers who are credited with inventing the world's first successful motor-operated airplane?**

- What is the probability that the contestant answers this question correctly?

  Unless you have:

  1. watched this particular contestant participate in this show many times,
  2. seen her asked this same question each time,
  3. and computed the relative frequency with which she gives the correct answer,

  you need to answer this question as a Bayesian!

  Uncertainty about the event *answering this question* needs to be expressed as a "degree of belief," informed by both data on the skill of the particular participant and how much she knows about inventors, as well as possibly prior knowledge of her performance in other game shows. Of course, your prior knowledge of the contestant may be minimal, or it may be well-informed.

Either way, your final answer remains a degree of belief about an uncertain, and inherently unrepeatable, state of nature.

The point of this hypothetical, light-hearted scenario is simply to highlight that a key distinction between the Frequentist and Bayesian approaches to inference is not the use (or nature) of prior information, but the manner in which probability is used. To the Bayesian, probability is the mathematical construct used to quantify uncertainty about an unknown state of nature, conditional on observed data and prior knowledge about the context in which that state occurs. To the Frequentist, probability is intrinsically linked to the concept of a repeated experiment, and the relative frequency with which a particular outcome occurs, conditional on that unknown state. This distinction remains key whether the Bayesian chooses to be *informative or subjective* in the specification of prior information, or chooses to be *non-informative or objective*.

Frequentists consider probability to be a physical phenomenon, like mass or wavelength, whereas Bayesians stipulate that probability exists in the mind of scientists, as any scientific construct [305].

It seems that the understanding of the concept of probability for the common human being is more associated with "degrees of belief" rather than relative frequency. Peter Diggle, President of The Royal Statistical Society between 2014 and 2016, was asked in an interview, "A different trend which has surged upwards in statistics during Peter's career is the popularity of Bayesian statistics. Does Peter consider himself a Bayesian?" He replied, "... you can't not believe in Bayes' theorem because it's true. But that doesn't make you a Bayesian in the philosophical sense. When people are making personal decisions – even if they don't formally process Bayes' theorem in their mind – they are adapting what they think they should believe in response to new evidence as it comes in. Bayes' theorem is just the formal mathematical machinery for doing that."

However, we should mention that psychological experiments suggest that human beings suffer from *anchoring*, a cognitive bias that causes us to rely too heavily on previous information (the prior), so that the updating process (posterior) due to new information (likelihood) is not as strong as Bayes' rule would suggest [220].

2.2 Subjectivity is not the key

The concepts of *subjectivity* and *objectivity* indeed characterize both statistical paradigms in differing ways. Among Bayesians, there are those who are immersed in *subjective* rationality [334, 103, 364, 252], but others who adopt *objective* prior distributions such as Jeffreys', reference, empirical, or robust priors [29, 241, 209, 35] to operationalize Bayes' rule and thereby weight quan-

titative (data-based) evidence. Among Frequentists, there are choices made about significance levels which, if not explicitly subjective, are typically not grounded in any objective and documented assessment of the relative losses of Type I and Type II errors.[1] In addition, both Frequentist and Bayesian Econometrician/Statisticians make decisions about the form of the data generating process, or "model", which – if not subject to rigorous diagnostic assessment – retains a subjective element that potentially influences the final inferential outcome. Although we all know that by definition, a model is a schematic and simplified approximation to reality,

"Since all models are wrong, the scientist cannot obtain a *correct* one by excessive elaboration. On the contrary, following William of Occam, he should seek an economical description of natural phenomena." [51].

We also know that "All models are wrong, but some are useful" [54], which is why model diagnostics are important. This task can be performed in both approaches. Particularly, the Bayesian framework can use predictive p-values for absolute testing [148, 27] or posterior odds ratios for relative statements [208, 226]. This is because the marginal likelihood, conditional on data, is interpreted as the evidence for the prior distribution [34].

In addition, what does objectivity mean in a Frequentist approach? For example, why should we use a 5% or 1% significance level rather than any other value? As someone said, the apparent objectivity is really a consensus [252]. In fact, "Student" (William Gosset) saw statistical significance at any level as being "nearly valueless" in itself [432]. But, this is not just a situation in the Frequentist approach. The cut-offs used to "establish" scientific evidence against a null hypothesis, in terms of log_{10} scale [209] or log_e scale [226] as shown in Table 1.1, are also *ad hoc*.

Although the true state of nature in Bayesian inference is expressed in "degrees of belief", the distinction between the two paradigms does not reside in one being more, or less, *subjective* than the other. Rather, the differences are philosophical, pedagogical, and methodological.

2.3 Estimation, hypothesis testing and prediction

All that is required to perform estimation, hypothesis testing (model selection), and prediction in the Bayesian approach is to apply Bayes' rule. This ensures coherence under a probabilistic view. However, there is no free lunch: coherence reduces flexibility. On the other hand, the Frequentist approach may not be coherent from a probabilistic point of view, but it is highly flexible. This approach can be seen as a toolkit that offers inferential solutions under

[1] Type I error is rejecting the null hypothesis when it is true, and Type II error is not rejecting the null hypothesis when it is false.

the umbrella of understanding probability as relative frequency. For instance, a point estimator in a Frequentist approach is found such that it satisfies good sampling properties like unbiasedness, efficiency, or a large sample property such as consistency.

A notable difference is that optimal Bayesian decisions are calculated by minimizing the expected value of the loss function with respect to the posterior distribution, i.e., conditional on observed data. In contrast, Frequentist "optimal" actions are based on the expected values over the distribution of the estimator (a function of data), conditional on the unknown parameters. This involves considering sampling variability.

The Bayesian approach allows for the derivation of the posterior distribution of any unknown object, such as parameters, latent variables, future or unobserved variables, or models. A major advantage is that predictions can account for estimation error, and predictive distributions (probabilistic forecasts) can be easily derived.

Hypothesis testing (model selection) in the Bayesian framework is based on *inductive logic* reasoning (*inverse probability*). Based on observed data, we evaluate which hypothesis is most tenable, performing this evaluation using posterior odds. These odds are in turn based on Bayes factors, which assess the evidence in favor of a null hypothesis while explicitly considering the alternative [226], following the rules of probability [252]. This approach compares how well hypotheses predict data [169], minimizes the weighted sum of type I and type II error probabilities [106, 308], and takes into account the implicit balance of losses [209, 38]. Posterior odds allow for the use of the same framework to analyze nested and non-nested models and perform model averaging.

However, Bayes factors cannot be based on improper or vague priors [234], the practical interplay between model selection and posterior distributions is not as straightforward as it may be in the Frequentist approach, and the computational burden can be more demanding due to the need to solve potentially difficult integrals.

On the other hand, the Frequentist approach establishes most of its estimators as the solution to a system of equations. Observe that optimization problems often reduce to solving systems. We can potentially obtain the distribution of these estimators, but most of the time, asymptotic arguments or resampling techniques are required. Hypothesis testing relies on pivotal quantities and/or resampling, and prediction is typically based on a *plug-in approach*, which means that estimation error is not taken into account.[2]

Comparing models depends on their structure. For instance, there are different Frequentist statistical approaches to compare nested and non-nested models. A nice feature in some situations is that there is a practical interplay between hypothesis testing and confidence intervals. For example, in the normal population mean hypothesis framework, you cannot reject a null hy-

[2]A pivot quantity is a function of unobserved parameters and observations whose probability distribution does not depend on the unknown parameters.

pothesis $H_0 : \mu = \mu^0$ at the α significance level (Type I error) if μ^0 is in the $1 - \alpha$ confidence interval. Specifically,

$$P\left(\mu \in \left[\hat{\mu} - |t_{N-1}^{\alpha/2}| \times \hat{\sigma}_{\hat{\mu}}, \hat{\mu} + |t_{N-1}^{\alpha/2}| \times \hat{\sigma}_{\hat{\mu}}\right]\right) = 1 - \alpha,$$

where $\hat{\mu}$ and $\hat{\sigma}_{\hat{\mu}}$ are the maximum likelihood estimators of the mean and standard error, $t_{N-1}^{\alpha/2}$ is the quantile value of the Student's t-distribution at the $\alpha/2$ probability level with $N - 1$ degrees of freedom, and N is the sample size.

A remarkable difference between the Bayesian and Frequentist inferential frameworks is the interpretation of credible/confidence intervals. Observe that once we have estimates, such that for example the previous interval is $[0.2, 0.4]$ given a 95% confidence level, we cannot say that $P(\mu \in [0.2, 0.4]) = 0.95$ in the Frequentist framework. In fact, this probability is either 0 or 1 in this approach, as μ is either in the interval or it is not. The problem is that we will never know for certain in applied settings. This is because $P(\mu \in [\hat{\mu} - |t_{N-1}^{0.025}| \times \hat{\sigma}_{\hat{\mu}}, \hat{\mu} + |t_{N-1}^{0.025}| \times \hat{\sigma}_{\hat{\mu}}]) = 0.95$ is interpreted in the context of repeated sampling. On the other hand, once we have the posterior distribution in the Bayesian framework, we can say that $P(\mu \in [0.2, 0.4]) = 0.95$.

Following common practice, most researchers and practitioners conduct hypothesis testing based on the p-value in the Frequentist framework. But **what is a *p*-value?** Most users do not know the answer, as statistical inference is often not performed by statisticians [35].[3] A p-value is the probability of obtaining a statistical summary of the data equal to or *more extreme* than what was actually observed, assuming that the null hypothesis is true.

Therefore, p-value calculations involve not just the observed data, but also more *extreme* hypothetical observations. Thus,

"What the use of p implies, therefore, is that a hypothesis that may be true may be rejected because it has not predicted observable results that have not occurred." [209]

Some researchers and practitioners using Frequentist inference often intertwines two distinct logical frameworks: Fisher's p-value approach [130] and the Neyman–Pearson significance testing framework [295]. The p-value serves as an informal, data-dependent measure of evidence against the null hypothesis. It is rooted in *reduction to absurdity* reasoning, where the extremeness of the observed data is assessed under the assumption that the null hypothesis is true. However, the p-value is frequently misinterpreted as the probability that the null hypothesis is false—a misconception known as the p-value fallacy [169].

In contrast, the Neyman–Pearson framework adopts a deductive, long-run perspective: it defines decision rules that control the frequency of Type I errors over repeated sampling, irrespective of the evidence in any particular case. Conflating these two frameworks leads to interpretational inconsistencies, especially when the p-value is used both as a measure of evidence and

[3] https://fivethirtyeight.com/features/not-even-scientists-can-easily-explain-p-values/

as a decision-making threshold. A clearer separation of these paradigms is essential for coherent statistical reasoning.

The American Statistical Association has several concerns regarding the use of the p-value as a cornerstone for hypothesis testing in science. This concern motivates the ASA's statement on p-values [412], which can be summarized in the following principles:

- "P-values can indicate how incompatible the data are with a specified statistical model."
- "P-values do not measure the probability that the studied hypothesis is true, or the probability that the data were produced by random chance alone."
- "Scientific conclusions and business or policy decisions should not be based solely on whether a p-value passes a specific threshold."
- "Proper inference requires full reporting and transparency."
- "A p-value, or statistical significance, does not measure the size of an effect or the importance of a result."
- "By itself, a p-value does not provide a good measure of evidence regarding a model or hypothesis."

To sum up, Fisher proposed the p-value as a witness rather than a judge. So, a p-value lower than the significance level means more inspection of the null hypothesis, but it is not a final conclusion about it.

Another key distinction between frequentist and Bayesian approaches lies in how scientific hypotheses are evaluated. Users of the Frequentist approach rely on the p-value, which quantifies the probability of observing data as extreme as—or more extreme than—the sample under the assumption that the null hypothesis is true. Bayesians, in contrast, use the Bayes factor, which compares the predictive performance of two competing hypotheses by evaluating the ratio of their marginal likelihoods. While the p-value reflects $P(\text{data} \mid \text{hypothesis})$, the Bayes factor is more aligned with $P(\text{hypothesis} \mid \text{data})$, though not equivalent. Notably, there exists an approximate relationship between the t-statistic and the Bayes factor in the context of regression coefficients [320], which offers some practical interpretability across paradigms. In particular, $|t| > (\log(N) + 6)^{1/2}$ corresponds to strong evidence in favor of rejecting the null hypothesis of no relevance of a control in a regression. Observe that, in this setting, the threshold of the t statistic, and as a consequence the significance level, depends on the sample size. This setting agrees with the idea in experimental designs of selecting the sample size such that we control Type I and Type II errors. In observational studies, we cannot control the sample size, but we can select the significance level.

See also [369] and [33] for nice exercises that reveal potential flaws of the

p-value (p) due to $p \sim U[0,1]$ under the null hypothesis,[4] and calibrations of the p-value to interpret it as the odds ratio and the error probability. In particular, $B(p) = -e \times p \times \log(p)$ when $p < e^{-1}$, and interpret this as the Bayes factor of H_0 to H_1, where H_1 denotes the unspecified alternative to H_0, and $\alpha(p) = \left(1 + [-e \times p \times \log(p)]^{-1}\right)^{-1}$ as the error probability α in rejecting H_0. Take into account that $B(p)$ and $\alpha(p)$ are lower bounds.

The logic of argumentation in the Frequentist approach is based on *deductive logic*, which means that it starts from a statement about the true state of nature (null hypothesis) and predicts what should be observed if this statement were true. On the other hand, the Bayesian approach is based on *inductive logic*, which means that it defines which hypothesis is more consistent with what is observed. The former inferential approach establishes that the truth of the premises implies the truth of the conclusion, which is why we reject or fail to reject hypotheses. The latter establishes that the premises supply some evidence, but not full assurance, of the truth of the conclusion, which is why we get probabilistic statements.

Here, there is a distinction between the effects of causes (forward causal inference) and the causes of effects (reverse causal inference) [153, 102]. To illustrate this point, imagine that a firm increases the price of a specific good. Economic theory would suggest that, as a result, demand for the good decreases. In this case, the premise (null hypothesis) is the price increase, and the consequence is the decrease in the firm's demand.

Alternatively, one could observe a reduction in a firm's demand and attempt to identify the cause behind it. For example, a reduction in quantity could be due to a negative supply shock. The Frequentist approach typically follows the first view (effects of causes), while Bayesian reasoning focuses on determining the probability of potential causes (causes of effects).

2.4 The likelihood principle

The **likelihood principle** states that in making inferences or decisions about the state of nature, all the relevant *experimental* information is given by the *likelihood function*. The Bayesian framework follows this statement, i.e., it is conditional on observed data.

We follow [34], who in turn followed [253], to illustrate the likelihood principle. We are given a coin and are interested in the probability, θ, of it landing heads when flipped. We wish to test $H_0 : \theta = 1/2$ versus $H_1 : \theta > 1/2$. An experiment is conducted by flipping the coin (independently) in a series of trials, with the result being the observation of 9 heads and 3 tails.

[4] https://joyeuserrance.wordpress.com/2011/04/22/proof-that-p-values-under-the-null-are-uniformly-distributed/ for a simple proof.

This is not yet enough information to specify $p(y|\theta)$, since the series of trials has not been explained. Two possibilities arise:

1. The experiment consisted of a predetermined 12 flips, so that $Y = [\text{Heads}]$ follows a $B(12, \theta)$ distribution. In this case, $p_1(y|\theta) = \binom{12}{y}\theta^y(1-\theta)^{12-y} = 220 \times \theta^9(1-\theta)^3$.
2. The experiment consisted of flipping the coin until 3 tails were observed ($r = 3$). In this case, Y, the number of heads (failures) before obtaining 3 tails, follows a $NB(3, 1-\theta)$ distribution. Here, $p_2(y|\theta) = \binom{y+r-1}{r-1}(1-(1-\theta)^y)(1-\theta)^r = 55 \times \theta^9(1-\theta)^3$.

Using a Frequentist approach, the significance level of $y = 9$ using the Binomial model against $\theta = 1/2$ would be:

$$\alpha_1 = P_{1/2}(Y \geq 9) = p_1(9|1/2) + p_1(10|1/2) + p_1(11|1/2) + p_1(12|1/2) = 0.073.$$

R code. The likelihood principle: Binomial model

```
success <- 9
# Number of observed success in n trials
n <- 12
# Number of trials
siglevel <- sum(sapply(9:n,function(y)dbinom(y,n,0.5)))
siglevel
0.073
```

For the Negative Binomial model, the significance level would be:

$$\alpha_2 = P_{1/2}(Y \geq 9) = p_2(9|1/2) + p_2(10|1/2) + \ldots = 0.0327.$$

R code. The likelihood principle: Negative Binomial model

```
success <- 3
# Number of target success (tails)
failures <- 9
# Number of failures
siglevel <- 1 - pnbinom((failures - 1),success,0.5)
siglevel
0.0327
```

We arrive at different conclusions using a significance level of 5%, whereas we obtain the same outcomes using a Bayesian approach because the kernels of both distributions are identical ($\theta^9 \times (1-\theta)^3$).

2.5 Why is not the Bayesian approach that popular?

At this stage, one might wonder why the Bayesian statistical framework is not the dominant inferential approach, despite its historical origin in 1763 [30], whereas the Frequentist statistical framework was largely developed in the early 20th century. The scientific debate over the Bayesian inferential approach lasted for 150 years, and this may be explained by some of the following factors.

One issue is the *apparent subjectivity* of the Bayesian approach, which runs counter to the strong conviction that science demands objectivity. Bayesian probability is considered a measure of degrees of belief, where the initial prior may be just a guess. This was not accepted as objective and rigorous science. Initial critics argued that Bayes was quantifying ignorance by assigning equal probabilities to all potential outcomes. As a consequence, prior distributions were dismissed [282].

Bayes himself seemed not to have believed in his idea. Although it seems that Bayes made his breakthrough in the late 1740s, he did not submit it for publication to the Royal Society. It was his friend, Richard Price, another Presbyterian minister, who rediscovered Bayes' idea, polished it, and published it.

However, it was Laplace who independently generalized Bayes' theorem in 1781. Initially, he applied it to gambling problems and soon thereafter to astronomy, combining various sources of information to advance research in situations where data was scarce. He later sought to apply his discovery to

finding the probability of causes, which he thought required large datasets, thus turning to demography. In this field, he had to perform large-scale calculations, leading to the development of Laplace's approximation and the central limit theorem [241]. Unfortunately, this came at the cost of abandoning his research on Bayesian inference.

Once *Laplace passed away in 1827*, Bayes' rule disappeared from the scientific discourse for almost a century. In part, personal attacks against Laplace led to the rule being forgotten. Moreover, there was a prevailing belief that statistics should not address causation, and that the prior was too subjective to be compatible with science. Nonetheless, practitioners continued to use Bayes' rule to solve problems in astronomy, communication, medicine, military affairs, and social issues with remarkable results.

Thus, the concept of degrees of belief to operationalize probability was abandoned in favor of scientific objectivity. Probability was then defined as the frequency with which an event occurs in many repeatable trials, which became the accepted norm. Critics of Laplace argued that these two concepts were diametrically opposed, although Laplace considered them to be basically equivalent when large sample sizes are involved [282].

The era of Frequentists, or sampling theorists, began, led by Karl Pearson and his nemesis, Ronald Fisher. Both were brilliant and persuasive characters, opposing the inverse probability (Bayesian) approach and making it nearly impossible to argue against their ideas. Pearson's legacy was carried on by his son, Egon, and Egon's friend, Jerzy Neyman. Both inherited the anti-Bayesian and anti-Fisher sentiments.

Despite the anti-Bayesian campaign among statisticians, some independent thinkers continued to develop Bayesian ideas, including Borel, Ramsey, and de Finetti, who were isolated in different countries: France, England, and Italy. However, the anti-Bayesian trio of Fisher, Neyman, and Egon Pearson dominated the spotlight in the 1920s and 1930s. Only a geophysicist, Harold Jeffreys, kept Bayesian inference alive during the 1930s and 1940s. Jeffreys was a quiet, reserved gentleman working in the astronomy department at Cambridge. He was Fisher's friend due to their shared character, although they were intellectual opposites when it came to Bayesian inference, leading to intense intellectual battles. Unfortunately for the Bayesian approach, *Jeffreys lost*. His work was highly technical, using confusing high-level mathematics. He focused on inference from scientific evidence, rather than guiding future actions based on decision theory, which was crucial in that era for mathematical statistics, especially during the Second World War. In contrast, Fisher was a dominant figure, persuasive in public and a master of practical applications, with his techniques written in a popular style with minimal mathematics.

Nevertheless, Bayes' rule achieved remarkable results in applied settings such as at AT&T and the U.S. Social Security system. Bayesian inference also played a significant role during the Second World War and the Cold War. Alan Turing used inverse probability at Bletchley Park to crack German messages encoded using the Enigma machine, which was employed by U-

boats. Andrei Kolmogorov used Bayesian methods to improve firing tables for Russian artillery. Bernard Koopman applied it for searching targets at sea, and the RAND Corporation used it during the Cold War. Unfortunately, *these Bayesian developments remained top secret for almost 40 years*, keeping the contribution of inverse probability hidden from modern history.

In the 1950s and 1960s, three mathematicians led the resurgence of the Bayesian approach: Good, Savage, and Lindley. However, it seems that they were reluctant to apply their theories to real-world problems. Despite the fact that the Bayesian approach proved its worth in various areas such as business decisions, naval searches, and lung cancer detection, it was largely applied to simple models due to its *mathematical complexity and requirement for large computations*. However, some breakthroughs changed this. First, hierarchical models were introduced by Lindley and Smith, where a complex model is decomposed into many smaller, easier-to-solve models. Second, Markov chain Monte Carlo (MCMC) methods were developed by Hastings in the 1970s [184] and the Geman brothers in the 1980s [155]. These methods were incorporated into the Bayesian inferential framework in the 1990s by Gelfand and Smith [146], and Tierney [398], when desktop computers gained sufficient computational power to solve complex models. Since then, the Bayesian inferential framework has gained increasing popularity among both practitioners and scientists.

2.6 A simple working example

We will illustrate some conceptual differences between the Bayesian and Frequentist statistical approaches by performing inference on a random sample $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_N]$, where $Y_i \overset{iid}{\sim} N(\mu, \sigma^2)$ for $i = 1, 2, \ldots, N$.

In particular, we set $\pi(\mu, \sigma) = \pi(\mu)\pi(\sigma) \propto \frac{1}{\sigma}$. This is a standard *non-informative improper* prior (Jeffreys prior, see Chapter 3). That is, this prior is perfectly compatible with the sample information. Thus, the posterior distribution is

$$\begin{aligned}
\pi(\mu,\sigma|\mathbf{y}) &\propto \frac{1}{\sigma}\times(\sigma^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-\mu)^2\right\}\\
&= \frac{1}{\sigma}\times(\sigma^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}((y_i-\bar{y})-(\mu-\bar{y}))^2\right\}\\
&= \frac{1}{\sigma}\exp\left\{-\frac{N}{2\sigma^2}(\mu-\bar{y})^2\right\}\times(\sigma)^{-N}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-\bar{y})^2\right\}\\
&= \frac{1}{\sigma}\exp\left\{-\frac{N}{2\sigma^2}(\mu-\bar{y})^2\right\}\times(\sigma)^{-(\alpha_n+1)}\exp\left\{-\frac{\alpha_n\hat{\sigma}^2}{2\sigma^2}\right\},
\end{aligned}$$

where $\bar{y}=\frac{\sum_{i=1}^{N}}{N}$, $\alpha_n=N-1$ and $\hat{\sigma}^2=\frac{\sum_{i=1}^{N}(y_i-\bar{y})^2}{N-1}$.

The first term in the last expression is the kernel of a normal density, $\mu|\sigma,\mathbf{y}\sim N(\bar{y},\sigma^2/N)$. The second term is the kernel of an inverted gamma density [429, p. 371], $\sigma|\mathbf{y}\sim IG(\alpha_n,\hat{\sigma}^2)$. Therefore, $\pi(\mu|\sigma,\mathbf{y})=(2\pi\sigma^2/N)^{-1/2}\exp\left\{\frac{-N}{2\sigma^2}(\mu-\bar{y})^2\right\}$ and $\pi(\sigma|\mathbf{y})=\frac{2}{\Gamma(\alpha_n/2)}\left(\frac{\alpha_n\hat{\sigma}^2}{2}\right)^{\alpha_n/2}\frac{1}{\sigma^{\alpha_n+1}}\times\exp\left\{-\frac{\alpha_n\hat{\sigma}^2}{2\sigma^2}\right\}$.

Observe that $\mathbb{E}[\mu|\sigma,\mathbf{y}]=\bar{y}$; this is also the maximum likelihood (Frequentist) point estimate of μ in this setting. In addition, the Frequentist $(1-\alpha)\%$ confidence interval and the Bayesian $(1-\alpha)\%$ credible interval have exactly the same form, $\bar{y}\pm|z_{\alpha/2}|\frac{\sigma}{\sqrt{N}}$, where $z_{\alpha/2}$ is the $\alpha/2$ percentile of a standard normal distribution. However, the interpretations are entirely different. The confidence interval has a probabilistic interpretation under sampling variability of $\bar{Y}$: in repeated sampling, $(1-\alpha)\%$ of the intervals $\bar{Y}\pm|z_{\alpha/2}|\frac{\sigma}{\sqrt{N}}$ would include μ. However, given an observed realization of $\bar{Y}$, say $\bar{y}$, the probability of $\bar{y}\pm|z_{\alpha/2}|\frac{\sigma}{\sqrt{N}}$ including μ is either 1 or 0. This is why we refer to it as a $(1-\alpha)\%$ confidence interval. On the other hand, $\bar{y}\pm|z_{\alpha/2}|\frac{\sigma}{\sqrt{N}}$ has a straightforward probabilistic interpretation in the Bayesian framework: there is a $(1-\alpha)\%$ probability that μ lies within this interval.

If we want to get the marginal posterior density of μ,

$$\begin{aligned}
\pi(\mu|\mathbf{y}) &= \int_0^\infty \pi(\mu,\sigma|\mathbf{y})d\sigma \\
&\propto \int_0^\infty \frac{1}{\sigma} \times (\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^N (y_i-\mu)^2\right\} d\sigma \\
&= \int_0^\infty \left(\frac{1}{\sigma}\right)^{N+1} \exp\left\{-\frac{N}{2\sigma^2}\frac{\sum_{i=1}^N (y_i-\mu)^2}{N}\right\} d\sigma \\
&= \left[\frac{2}{\Gamma(N/2)}\left(\frac{N\sum_{i=1}^N (y_i-\mu)^2}{2N}\right)^{N/2}\right]^{-1} \\
&\propto \left[\sum_{i=1}^N (y_i-\mu)^2\right]^{-N/2} \\
&= \left[\sum_{i=1}^N ((y_i-\bar{y})-(\mu-\bar{y}))^2\right]^{-N/2} \\
&= [\alpha_n\hat{\sigma}^2 + N(\mu-\bar{y})^2]^{-N/2} \\
&\propto \left[1+\frac{1}{\alpha_n}\left(\frac{\mu-\bar{y}}{\hat{\sigma}/\sqrt{N}}\right)^2\right]^{-(\alpha_n+1)/2}.
\end{aligned}$$

The fourth line arises from the kernel of an inverted gamma density with N degrees of freedom in the integral [429].

The last expression represents the kernel of a Student's t density function with $\alpha_n = N-1$ degrees of freedom, expected value equal to $\bar{y}$, and variance $\frac{\hat{\sigma}^2}{N}\left(\frac{\alpha_n}{\alpha_n-2}\right)$. Therefore, $\mu|\mathbf{y} \sim t\left(\bar{y}, \frac{\hat{\sigma}^2}{N}\left(\frac{\alpha_n}{\alpha_n-2}\right), \alpha_n\right)$.

Observe that a $(1-\alpha)\%$ confidence interval and a $(1-\alpha)\%$ credible interval have the same form, $\bar{y} \pm |t^{\alpha_n}_{\alpha/2}|\frac{\hat{\sigma}}{\sqrt{N}}$, where $t^{\alpha_n}_{\alpha/2}$ is the $\alpha/2$ percentile of a Student's t distribution. However, the interpretations are entirely different.

The mathematical similarity between the Frequentist and Bayesian expressions in this example arises from the use of an improper prior.

2.6.1 Example: Math test

You have a random sample of math scores of size $N = 50$ from a normal distribution, $Y_i \sim N(\mu, \sigma^2)$. The sample mean and variance are equal to 102 and 10, respectively. Assuming an improper prior equal to $1/\sigma$, we proceed with the following tasks:

- Compute the 95% confidence and credible intervals for μ.
- Determine the posterior probability that $\mu > 103$.

Using the fact that $\mu|\mathbf{y} \sim t\left(\bar{y}, \frac{\hat{\sigma}^2}{N}\left(\frac{\alpha_n}{\alpha_n - 2}\right), \alpha_n\right)$, which implies that the confidence and credible intervals for μ are given by

$$\bar{y} \pm |t^{\alpha_n}_{\alpha/2}| \frac{\hat{\sigma}}{\sqrt{N}},$$

where $\bar{y} = 102$, $\hat{\sigma}^2 = 10$, and $\alpha_n = 49$. Thus, the 95% confidence and credible intervals for μ are the same, namely $(101.1, 102.9)$, and the posterior probability that $\mu > 103$ is 1.49% given the sample information.

R code. Example: Math test

```
N <- 50 # Sample size
y_bar <- 102 # Sample mean
s2 <- 10 # Sample variance
alpha <- N - 1
serror <- (s2/N)^0.5
LimInf <- y_bar - abs(qt(0.025, alpha)) * serror
LimInf
101.101
# Lower bound
LimSup <- y_bar + abs(qt(0.025, alpha)) * serror
LimSup
102.898
# Upper bound
y.cut <- 103
P <- 1-metRology::pt.scaled(y.cut, df = alpha, mean = y_bar,
      sd = serror)
P
0.0149
# Probability of mu greater than y.cut
```

2.7 Summary

The differences between the Bayesian and Frequentist inferential approaches are philosophical, particularly with regard to the role of probability; pedagogical, especially in relation to the use of inference for decision-making; and methodological, due to differences in their mathematical and computational frameworks. Although, at the methodological level, the debate has become considerably muted—except for certain aspects of inference—there is widespread recognition that each approach has much to contribute to statistical practice [168, 28, 225]. As Bradley Efron stated, "Computer-age statistical

inference at its most successful **combines** elements of the two philosophies" [123].

2.8 Exercises

1. **Jeffreys-Lindley's Paradox**

   The **Jeffreys-Lindley's paradox** [209, 254] represents an apparent disagreement between the Bayesian and Frequentist frameworks in a hypothesis testing scenario.

   In particular, assume that in a city, 49,581 boys and 48,870 girls have been born over 20 years. Assume that the male births follow a Binomial distribution with probability θ. We wish to test the null hypothesis $H_0 : \theta = 0.5$ versus the alternative hypothesis $H_1 : \theta \neq 0.5$.

   - Show that the posterior model probability for the null model is approximately 0.95. Assume $\pi(H_0) = \pi(H_1) = 0.5$, and that $\pi(\theta)$ follows a uniform distribution, i.e., $U(0, 1)$, under H_1.
   - Show that the p-value for this hypothesis test is 0.0235 using the normal approximation, $Y \sim N(N \times \theta, N \times \theta \times (1 - \theta))$.

2. We want to test $H_0 : \mu = \mu_0$ versus $H_1 : \mu \neq \mu_0$ given $Y_i \overset{iid}{\sim} N(\mu, \sigma^2)$.

   Assume $\pi(H_0) = \pi(H_1) = 0.5$, and that $\pi(\mu, \sigma) \propto 1/\sigma$ under the alternative hypothesis.

   Show that

$$p(\mathbf{y}|\mathcal{M}_1) = \frac{\pi^{-N/2}}{2}\Gamma(N/2)\left(\frac{1}{\alpha_n\hat{\sigma}^2}\right)^{N/2}\left(\frac{N}{\alpha_n\hat{\sigma}^2}\right)^{-1/2}\frac{\Gamma(1/2)\Gamma(\alpha_n/2)}{\Gamma((\alpha_n+1)/2)}$$

   and

$$p(\mathbf{y}|\mathcal{M}_0) = (2\pi)^{-N/2}\left[\frac{2}{\Gamma(N/2)}\left(\frac{N}{2}\frac{\sum_{i=1}^{N}(y_i-\mu_0)^2}{N}\right)^{N/2}\right]^{-1}.$$

   Then, the posterior odds ratio is:

$$\begin{aligned} PO_{01} &= \frac{p(\mathbf{y}|\mathcal{M}_0)}{p(\mathbf{y}|\mathcal{M}_1)} \\ &= \frac{\Gamma((\alpha_n+1)/2)}{\Gamma(1/2)\Gamma(\alpha_n/2)}(\alpha_n\hat{\sigma}^2/N)^{-1/2}\left[1+\frac{(\mu_0-\bar{y})^2}{\alpha_n\hat{\sigma}^2/N}\right]^{-\left(\frac{\alpha_n+1}{2}\right)}, \end{aligned}$$

where $\alpha_n = N - 1$ and $\hat{\sigma}^2 = \frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N-1}$.

Find the relationship between the posterior odds ratio and the classical test statistic for the null hypothesis.

3. **Math Test Continues**

   Using the setting of the **Example: Math Test** in subsection 2.6.1, test $H_0 : \mu = \mu_0$ versus $H_1 : \mu \neq \mu_0$ where $\mu_0 = \{100, 100.5, 101, 101.5, 102\}$.

   - What is the p-value for these hypothesis tests?
   - Find the posterior model probability of the null model for each μ_0.

3

Cornerstone models: Conjugate families

We will introduce conjugate families, which are distributions for which the posterior distribution belongs to the same family as the prior distribution, given the likelihood. We provide some examples and solve them both analytically and computationally. We begin with simple examples of discrete and continuous distributions and then study the linear model in detail, both univariate and multivariate, deriving the posterior distributions, the marginal likelihood, and the predictive distribution analytically. Additionally, we will include mathematical and computational exercises in **R**.

3.1 Motivation of conjugate families

By observing the three fundamental pieces of Bayesian analysis –the posterior distribution (parameter inference), the marginal likelihood (hypothesis testing), and the predictive distribution (prediction)– as given in equations 3.1, 3.2, and 3.3, respectively, we can understand that some of the initial limitations of Bayesian analysis were due to the absence of algorithms for sampling from non-standard posterior distributions (equation 3.1), and the lack of analytical solutions for the marginal likelihood (equation 3.2) and the predictive distribution (equation 3.3), both of which require significant computational power.

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})}{p(\boldsymbol{y})}, \tag{3.1}$$

$$p(\boldsymbol{y}) = \int_{\boldsymbol{\Theta}} p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{3.2}$$

and

$$p(\boldsymbol{y}_0 \mid \boldsymbol{y}) = \int_{\boldsymbol{\Theta}} p(\boldsymbol{y}_0 \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta}, \tag{3.3}$$

Although algorithms for sampling from non-standard posterior distributions have existed since the second half of the last century [284, 184, 155],

their application within the Bayesian framework emerged later [146, 398], likely coinciding with the rise in computational power of desktop computers. However, it is still common practice today to use models with standard conditional posterior distributions in order to reduce computational requirements. In addition, mathematical techniques coupled with computational algorithms [147, 84, 89] and approximations [399, 216] are employed to obtain the marginal likelihood (prior predictive).

Despite these advances, two potentially conflicting desirable model specification features are evident from equations 3.1, 3.2, and 3.3: (1) analytical solutions and (2) the posterior distribution belonging to the same family as the prior distribution for a given likelihood. The latter is known as *conjugate priors*, a family of priors that is closed under sampling [365].

These features are desirable because the former facilitates hypothesis testing and predictive analysis, while the latter ensures invariance in the prior-to-posterior updating process. Both features reduce computational burden.

Although each of these features can be achieved independently –such as using improper priors for analytical tractability and broadly defining the family of priors for conjugacy– these features are in conflict.

Fortunately, we can achieve both characteristics if we assume that the data-generating process follows a distribution function in the *exponential family*. That is, given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \dots\ Y_N]^\top$, a probability density function $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ belongs to the exponential family if it has the form:

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{\theta}) &= \prod_{i=1}^{N} h(y_i) C(\boldsymbol{\theta}) \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{T}(y_i)\right\} \\ &= h(\boldsymbol{y}) C(\boldsymbol{\theta})^N \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{T}(\boldsymbol{y})\right\} \\ &= h(\boldsymbol{y}) \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{T}(\boldsymbol{y}) - A(\boldsymbol{\theta})\right\}, \end{aligned} \tag{3.4}$$

where $h(\boldsymbol{y}) = \prod_{i=1}^{N} h(y_i)$ is a non-negative function, $\eta(\boldsymbol{\theta})$ is a known function of the parameters, and $A(\boldsymbol{\theta}) = \log\left\{\int_{\boldsymbol{Y}} h(\boldsymbol{y}) \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{T}(\boldsymbol{y})\right\} d\boldsymbol{y}\right\} = -N \log\left(C(\boldsymbol{\theta})\right)$ is the normalization factor. Additionally, $\boldsymbol{T}(\boldsymbol{y}) = \sum_{i=1}^{N} \boldsymbol{T}(y_i)$ is the vector of sufficient statistics for the distribution (by the factorization theorem).

If the support of $\boldsymbol{Y}$ is independent of $\boldsymbol{\theta}$, the family is said to be *regular*; otherwise, it is *irregular*. Furthermore, if we set $\eta = \eta(\boldsymbol{\theta})$, the exponential family is said to be in the *canonical form*.

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{\eta}) &= h(\boldsymbol{y}) D(\boldsymbol{\eta})^N \exp\left\{\eta^\top \boldsymbol{T}(\boldsymbol{y})\right\} \\ &= h(\boldsymbol{y}) \exp\left\{\eta^\top \boldsymbol{T}(\boldsymbol{y}) - B(\boldsymbol{\eta})\right\}. \end{aligned}$$

A nice feature of this representation is that $\mathbb{E}[\boldsymbol{T}(\boldsymbol{y}) \mid \boldsymbol{\eta}] = \nabla B(\boldsymbol{\eta})$ and $Var[\boldsymbol{T}(\boldsymbol{y}) \mid \boldsymbol{\eta}] = \nabla^2 B(\boldsymbol{\eta})$.

3.1.1 Examples of exponential family distributions

1. **Discrete distributions**

   Let's show that some of the most common distributions for random variables, which can take values on a finite or countably infinite set, are part of the exponential family.

   Poisson distribution

   Given a random sample $\boldsymbol{Y} = [Y_1 \; Y_2 \; \ldots \; Y_N]^\top$ from a *Poisson distribution* let's show that $p(\boldsymbol{y} \mid \lambda)$ is in the exponential family.

$$\begin{aligned} p(\boldsymbol{y} \mid \lambda) &= \prod_{i=1}^{N} \frac{\lambda^{y_i} \exp(-\lambda)}{y_i!} \\ &= \frac{\lambda^{\sum_{i=1}^{N} y_i} \exp(-N\lambda)}{\prod_{i=1}^{N} y_i!} \\ &= \frac{\exp(-N\lambda) \exp(\sum_{i=1}^{N} y_i \log(\lambda))}{\prod_{i=1}^{N} y_i!}, \end{aligned}$$

   then $h(\boldsymbol{y}) = \left[\prod_{i=1}^{N} y_i!\right]^{-1}$, $\eta(\lambda) = \log(\lambda)$, $T(\boldsymbol{y}) = \sum_{i=1}^{N} y_i$ (sufficient statistic) and $C(\lambda) = \exp(-\lambda)$.

   If we set $\eta = \log(\lambda)$, then

$$p(\boldsymbol{y} \mid \eta) = \frac{\exp(\eta \sum_{i=1}^{N} y_i - N \exp(\eta))}{\prod_{i=1}^{N} y_i!},$$

   such that $B(\eta) = N \exp(\eta)$, then $\nabla(B(\eta)) = N \exp(\eta) = N\lambda = \mathbb{E}\left[\sum_{i=1}^{N} y_i \middle| \lambda\right]$, that is, $\mathbb{E}\left[\frac{\sum_{i=1}^{N} y_i}{N} \middle| \lambda\right] = \mathbb{E}[\bar{y} \mid \lambda] = \lambda$, and $\nabla^2(B(\eta)) = N \exp(\eta) = N\lambda = Var\left[\sum_{i=1}^{N} y_i \middle| \lambda\right] = N^2 \times Var\,[\bar{y}|\lambda]$, then $Var\,[\bar{y}|\lambda] = \frac{\lambda}{N}$.

   Bernoulli distribution

   Given a random sample $\boldsymbol{Y} = [Y_1 \; Y_2 \; \ldots \; Y_N]^\top$ from a *Bernoulli distribution* let's show that $p(\boldsymbol{y} \mid \theta)$ is in the exponential family.

$$\begin{aligned} p(\boldsymbol{y} \mid \theta) &= \prod_{i=1}^{N} \theta^{y_i} (1-\theta)^{1-y_i} \\ &= \theta^{\sum_{i=1}^{N} y_i} (1-\theta)^{N - \sum_{i=1}^{N} y_i} \\ &= (1-\theta)^N \exp\left\{\sum_{i=1}^{N} y_i \log\left(\frac{\theta}{1-\theta}\right)\right\}, \end{aligned}$$

then $h(\boldsymbol{y}) = \mathbb{1}[y_i \in \{0,1\}]$ (indicator function), $\eta(\theta) = \log\left(\frac{\theta}{1-\theta}\right)$, $T(\boldsymbol{y}) = \sum_{i=1}^N y_i$ and $C(\theta) = 1-\theta$.

Write this distribution in the canonical form, and find the mean and variance of the sufficient statistic (Exercise 1).

Multinomial distribution

Given a random sample $\boldsymbol{Y} = [\boldsymbol{Y}_1\ \boldsymbol{Y}_2\ \dots\ \boldsymbol{Y}_N]$ from a *m-dimensional multinomial distribution*, where $\boldsymbol{Y}_i = [Y_{i1}\ Y_{i2}\ \dots\ Y_{im}]$, $\sum_{l=1}^m Y_{il} = n$, n independent trials each of which leads to a success for exactly one of m categories with probabilities $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \dots\ \theta_m]$, $\sum_{l=1}^m \theta_l = 1$. Let's show that $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ is in the exponential family.

$$
\begin{aligned}
p(\boldsymbol{y} \mid \boldsymbol{\theta}) &= \prod_{i=1}^N \frac{n!}{\prod_{l=1}^m y_{il}!} \prod_{l=1}^m \theta_l^{y_{il}} \\
&= \frac{(n!)^N}{\prod_{i=1}^N \prod_{l=1}^m y_{il}!} \exp\left\{\sum_{i=1}^N \sum_{l=1}^m y_{il} \log(\theta_l)\right\} \\
&= \frac{(n!)^N}{\prod_{i=1}^N \prod_{l=1}^m y_{il}!} \exp\left\{\left(N \times n - \sum_{i=1}^N \sum_{l=1}^{m-1} y_{il}\right) \log(\theta_m)\right. \\
&\left. + \sum_{i=1}^N \sum_{l=1}^{m-1} y_{il} \log(\theta_l)\right\} \\
&= \frac{(n!)^N}{\prod_{i=1}^N \prod_{l=1}^m y_{il}!} \theta_m^{N \times n} \exp\left\{\sum_{i=1}^N \sum_{l=1}^{m-1} y_{il} \log(\theta_l/\theta_m)\right\},
\end{aligned}
$$

then $h(\boldsymbol{y}) = \frac{(n!)^N}{\prod_{i=1}^N \prod_{l=1}^m y_{il}!}$, $\eta(\boldsymbol{\theta}) = \left[\log\left(\frac{\theta_1}{\theta_m}\right) \dots \log\left(\frac{\theta_{m-1}}{\theta_m}\right)\right]$, $T(\boldsymbol{y}) = \left[\sum_{i=1}^N y_{i1} \cdots \sum_{i=1}^N y_{im-1}\right]$ and $C(\boldsymbol{\theta}) = \theta_m^n$.

2. **Continuous distributions**

Let's show that some of the most common distributions for random variables, which can take any value within a certain range or interval –an infinite number of possible values– are part of the exponential family.

Normal distribution

Given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \dots\ Y_N]^\top$ from a *normal dis-*

tribution let's show that $p(\boldsymbol{y} \mid \mu, \sigma^2)$ is in the exponential family.

$$\begin{aligned}
p(\boldsymbol{y} \mid \mu, \sigma^2) &= \prod_{i=1}^{N} \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{1}{2\sigma^2}\left(y_i - \mu\right)^2\right\} \\
&= (2\pi)^{-N/2}(\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y_i - \mu\right)^2\right\} \\
&= (2\pi)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N} y_i^2 + \frac{\mu}{\sigma^2}\sum_{i=1}^{N} y_i \right. \\
&\left. - N\frac{\mu^2}{2\sigma^2} - \frac{N}{2}\log(\sigma^2)\right\},
\end{aligned}$$

then $h(\boldsymbol{y}) = (2\pi)^{-N/2}$, $\eta(\mu, \sigma^2) = \left[\frac{\mu}{\sigma^2} \ \frac{-1}{2\sigma^2}\right]$, $T(\boldsymbol{y}) = \left[\sum_{i=1}^{N} y_i \ \sum_{i=1}^{N} y_i^2\right]$ and $C(\mu, \sigma^2) = \exp\left\{-\frac{\mu^2}{2\sigma^2} - \frac{\log(\sigma^2)}{2}\right\}$.
Observe that

$$p(\boldsymbol{y} \mid \mu, \sigma^2) = (2\pi)^{-N/2} \exp\left\{\eta_1 \sum_{i=1}^{N} y_i + \eta_2 \sum_{i=1}^{N} y_i^2 - \frac{N}{2}\log(-2\eta_2) + \frac{N}{4}\frac{\eta_1^2}{\eta_2}\right\},$$

where $B(\boldsymbol{\eta}) = \frac{N}{2}\log(-2\eta_2) - \frac{N}{4}\frac{\eta_1^2}{\eta_2}$. Then,

$$\nabla B(\boldsymbol{\eta}) = \begin{bmatrix} -\frac{N}{2}\frac{\eta_1}{\eta_2} \\ -\frac{N}{2}\frac{1}{\eta_2} + \frac{N}{4}\frac{\eta_1^2}{\eta_2^2} \end{bmatrix} = \begin{bmatrix} N \times \mu \\ N \times (\mu^2 + \sigma^2) \end{bmatrix} = \begin{bmatrix} \mathbb{E}\left[\sum_{i=1}^{N} y_i \middle| \mu, \sigma^2\right] \\ \mathbb{E}\left[\sum_{i=1}^{N} y_i^2 \middle| \mu, \sigma^2\right] \end{bmatrix}.$$

Multivariate normal distribution

Given $\boldsymbol{Y} = [\boldsymbol{Y}_1 \ \boldsymbol{Y}_2 \ \ldots \ \boldsymbol{Y}_p]$ a $N \times p$ matrix such that $\boldsymbol{Y}_i \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $i = 1, 2, \ldots, N$, that is, each i-th row of $\boldsymbol{Y}$ follows a *multivariate normal distribution*. Then, assuming independence between rows, let's show that $p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is in the exponential family.

$$\begin{aligned}
p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \prod_{i=1}^{N} (2\pi)^{-p/2}|\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}\left(\boldsymbol{y}_i - \boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_i - \boldsymbol{\mu}\right)\right\} \\
&= (2\pi)^{-pN/2}|\Sigma|^{-N/2} \exp\left\{-\frac{1}{2} tr\left[\sum_{i=1}^{N}\left(\boldsymbol{y}_i - \boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_i - \boldsymbol{\mu}\right)\right]\right\} \\
&= (2\pi)^{-pN/2}|\Sigma|^{-N/2} \exp\left\{-\frac{1}{2} tr\left[\left(\boldsymbol{S} + N\left(\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\right)\left(\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\right)^\top\right)\boldsymbol{\Sigma}^{-1}\right]\right\} \\
&= (2\pi)^{-pN/2} \exp\left\{-\frac{1}{2}\left[\left(vec\left(\boldsymbol{S}\right)^\top + N vec\left(\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top\right)^\top\right) vec\left(\boldsymbol{\Sigma}^{-1}\right)\right.\right. \\
&\left.\left. -2N\hat{\boldsymbol{\mu}}^\top vec\left(\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\right) + N tr\left(\boldsymbol{\mu}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\right) + N\log(|\boldsymbol{\Sigma}|)\right]\right\},
\end{aligned}$$

where the second line uses the trace operator (tr), and its invariance under cyclic permutation is applied in the third line. Additionally, we add and subtract $\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i$ inside each parenthesis, resulting in $\boldsymbol{S} = \sum_{i=1}^{N} (\boldsymbol{y}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}})^\top$. The fourth line is obtained after collecting terms and using properties of the trace operator to introduce the vectorization operator (vec), specifically, $\mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{B}) = \mathrm{vec}(\boldsymbol{A})^\top \mathrm{vec}(\boldsymbol{B})$, and $\mathrm{vec}(\boldsymbol{A}+\boldsymbol{B}) = \mathrm{vec}(\boldsymbol{A}) + \mathrm{vec}(\boldsymbol{B})$.

Then $h(\boldsymbol{y}) = (2\pi)^{-pN/2}$, $\eta(\boldsymbol{\mu}, \boldsymbol{\Sigma})^\top = \left[\left(vec\left(\boldsymbol{\Sigma}^{-1}\right)\right)^\top \quad \left(vec\left(\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\right)\right)^\top\right]$, $T(\boldsymbol{y}) = \left[-\frac{1}{2}\left(vec\left(\boldsymbol{S}\right)^\top + N vec\left(\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top\right)^\top\right) \quad -N\hat{\boldsymbol{\mu}}^\top\right]^\top$ and $C(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \exp\left\{-\frac{1}{2}\left(tr\left(\boldsymbol{\mu}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\right) + \log(|\Sigma|)\right)\right\}$.

3.2 Conjugate prior to exponential family

Theorem 4.2.1

The prior distribution $\pi(\boldsymbol{\theta}) \propto C(\boldsymbol{\theta})^{b_0} \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{a}_0\right\}$ is conjugate to the exponential family (equation 3.4).

Proof

$$\begin{aligned}\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) &\propto C(\boldsymbol{\theta})^{b_0} \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{a}_0\right\} \times h(\boldsymbol{y}) C(\boldsymbol{\theta})^N \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{T}(\boldsymbol{y})\right\} \\ &\propto C(\boldsymbol{\theta})^{N+b_0} \exp\left\{\eta(\boldsymbol{\theta})^\top (\boldsymbol{T}(\boldsymbol{y}) + \boldsymbol{a}_0)\right\}.\end{aligned}$$

Observe that the posterior is in the exponential family, $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto C(\boldsymbol{\theta})^{\beta_n} \exp\left\{\eta(\boldsymbol{\theta})^\top \boldsymbol{\alpha}_n\right\}$, $\beta_n = N + b_0$ and $\boldsymbol{\alpha}_n = \boldsymbol{T}(\boldsymbol{y}) + \boldsymbol{a}_0$.

Remarks

We observe, by comparing the prior and the likelihood, that b_0 plays the role of a hypothetical sample size, and $\boldsymbol{a}_0$ plays the role of hypothetical sufficient statistics. This perspective aids the elicitation process, that is, integrating non-sample information into the prior distribution.

We established this result in the *standard form* of the exponential family. We can also establish it in the *canonical form* of the exponential family. Observe that, given $\boldsymbol{\eta} = \boldsymbol{\eta}(\boldsymbol{\theta})$, another way to derive a prior for $\boldsymbol{\eta}$ is to use the change of variable theorem, given a bijective function.

In the case where there is a regular conjugate prior, [109] show that the posterior expectation of the sufficient statistics is a weighted average between the prior expectation and the likelihood estimate.

3.2.1 Examples: Theorem 4.2.1

1. **Likelihood functions from discrete distributions**

The Poisson-gamma model

Given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \ldots\ Y_N]^\top$ from a Poisson distribution then a conjugate prior density for λ has the form

$$\begin{aligned}\pi(\lambda) &\propto (\exp(-\lambda))^{b_0} \exp\{a_0 \log(\lambda)\} \\ &= \exp(-\lambda b_0)\lambda^{a_0} \\ &= \exp(-\lambda\beta_0)\lambda^{\alpha_0 - 1}.\end{aligned}$$

This is the kernel of a gamma density in the *rate parametrization*, $G(\alpha_0, \beta_0)$, where $\alpha_0 = a_0 + 1$ and $\beta_0 = b_0$.[1] Thus, a prior conjugate distribution for the Poisson likelihood is a gamma distribution.

Since $\sum_{i=1}^N Y_i$ is a sufficient statistic for the Poisson distribution, we can interpret a_0 as the number of occurrences in b_0 experiments.

Observe that

$$\begin{aligned}\pi(\lambda \mid \boldsymbol{y}) &\propto \exp(-\lambda\beta_0)\lambda^{\alpha_0-1} \times \exp(-N\lambda)\lambda^{\sum_{i=1}^N y_i} \\ &= \exp(-\lambda(N+\beta_0))\lambda^{\sum_{i=1}^N y_i + \alpha_0 - 1}.\end{aligned}$$

As expected, this is the kernel of a gamma distribution, which means $\lambda \mid \boldsymbol{y} \sim G(\alpha_n, \beta_n)$, $\alpha_n = \sum_{i=1}^N y_i + \alpha_0$ and $\beta_n = N + \beta_0$.

Observe that α_0/β_0 is the prior mean, and α_0/β_0^2 is the prior variance. Then, $\alpha_0 \to 0$ and $\beta_0 \to 0$ imply a non-informative prior such that the posterior mean converges to the maximum likelihood estimate $\bar{y} = \frac{\sum_{i=1}^N y_i}{N}$,

$$\begin{aligned}\mathbb{E}[\lambda \mid \boldsymbol{y}] &= \frac{\alpha_n}{\beta_n} \\ &= \frac{\sum_{i=1}^N y_i + \alpha_0}{N + \beta_0} \\ &= \frac{N\bar{y}}{N+\beta_0} + \frac{\alpha_0}{N+\beta_0}.\end{aligned}$$

The posterior mean is a weighted average of the sample and prior information. This is a general result for regular conjugate priors [109]. Note that $\lim_{N\to\infty} \mathbb{E}[\lambda \mid \boldsymbol{y}] = \bar{y}$.

Additionally, $\alpha_0 \to 0$ and $\beta_0 \to 0$ corresponds to $\pi(\lambda) \propto \frac{1}{\lambda}$, which is an improper prior. Improper priors may have undesirable consequences for Bayes factors (hypothesis testing); see below for a discussion of this in the linear regression framework. In this example, we can obtain analytical solutions for the marginal likelihood

[1] Another parametrization of the gamma density is the *scale parametrization*, where $\kappa_0 = 1/\beta_0$. See the health insurance example in Chapter 1.

and the predictive distribution (see the health insurance example and Exercise 3 in Chapter 1).

The Bernoulli-beta model

Given a random sample $\boldsymbol{Y} = [Y_1 \; Y_2 \; \ldots \; Y_N]^\top$ from a Bernoulli distribution then a conjugate prior density for θ has the form

$$\begin{aligned}\pi(\theta) &\propto (1-\theta)^{b_0} \exp\left\{a_0 \log\left(\frac{\theta}{1-\theta}\right)\right\} \\ &= (1-\theta)^{b_0-a_0}\theta^{a_0} \\ &= \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}.\end{aligned}$$

This is the kernel of a beta density, $B(\alpha_0, \beta_0)$, where $\alpha_0 = a_0 + 1$ and $\beta_0 = b_0 - a_0 + 1$. A prior conjugate distribution for the Bernoulli likelihood is a beta distribution. Given that b_0 is the hypothetical sample size and a_0 is the hypothetical sufficient statistic (the number of successes), $b_0 - a_0$ represents the number of failures. This implies that α_0 is the number of prior successes plus one, and β_0 is the number of prior failures plus one.

Since the mode of a beta-distributed random variable is given by $\frac{\alpha_0-1}{\alpha_0+\beta_0-2} = \frac{a_0}{b_0}$, we can interpret this as the prior probability of success. Setting $\alpha_0 = 1$ and $\beta_0 = 1$, which corresponds to a uniform distribution on the interval [0, 1], represents a setting with 0 successes (and 0 failures) in 0 experiments.

Observe that

$$\begin{aligned}\pi(\theta \mid \boldsymbol{y}) &\propto \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1} \times \theta^{\sum_{i=1}^N y_i}(1-\theta)^{N-\sum_{i=1}^N y_i} \\ &= \theta^{\alpha_0+\sum_{i=1}^N y_i-1}(1-\theta)^{\beta_0+N-\sum_{i=1}^N y_i-1}.\end{aligned}$$

The posterior distribution is beta, $\theta \mid \boldsymbol{y} \sim B(\alpha_n, \beta_n)$, $\alpha_n = \alpha_0 + \sum_{i=1}^N y_i$ and $\beta_n = \beta_0 + N - \sum_{i=1}^N y_i$, where the posterior mean $\mathbb{E}[\theta \mid \boldsymbol{y}] = \frac{\alpha_n}{\alpha_n+\beta_n} = \frac{\alpha_0+N\bar{y}}{\alpha_0+\beta_0+N} = \frac{\alpha_0+\beta_0}{\alpha_0+\beta_0+N}\frac{\alpha_0}{\alpha_0+\beta_0} + \frac{N}{\alpha_0+\beta_0+N}\bar{y}$. The posterior mean is a weighted average between the prior mean and the maximum likelihood estimate.

El marginal likelihood in this setting is

$$\begin{aligned}p(\boldsymbol{y}) &= \int_0^1 \frac{\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}}{B(\alpha_0,\beta_0)} \times \theta^{\sum_{i=1}^N y_i}(1-\theta)^{N-\sum_{i=1}^N y_i} d\theta \\ &= \frac{B(\alpha_n,\beta_n)}{B(\alpha_0,\beta_0)},\end{aligned}$$

where $B(\cdot,\cdot)$ is the beta function.

In addition, the predictive density is

$$\begin{aligned}
p(y_0 \mid \boldsymbol{y}) &= \int_0^1 \theta^{y_0}(1-\theta)^{1-y_0} \times \frac{\theta^{\alpha_n-1}(1-\theta)^{\beta_n-1}}{B(\alpha_n,\beta_n)} d\theta \\
&= \frac{B(\alpha_n+y_0, \beta_n+1-y_0)}{B(\alpha_n,\beta_n)} \\
&= \frac{\Gamma(\alpha_n+\beta_n)\Gamma(\alpha_n+y_0)\Gamma(\beta_n+1-y_0)}{\Gamma(\alpha_n+\beta_n+1)\Gamma(\alpha_n)\Gamma(\beta_n)} \\
&= \left\{ \begin{array}{ll} \frac{\alpha_n}{\alpha_n+\beta_n}, & y_0 = 1 \\ \frac{\beta_n}{\alpha_n+\beta_n}, & y_0 = 0 \end{array} \right\}.
\end{aligned}$$

This is a Bernoulli distribution with probability of success equal to $\frac{\alpha_n}{\alpha_n+\beta_n}$.

The multinomial-Dirichlet model

Given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \ldots\ Y_N]^\top$ from a multinomial distribution then a conjugate prior density for $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \ldots\ \theta_m]$ has the form

$$\begin{aligned}
\pi(\boldsymbol{\theta}) &\propto \theta_m^{b_0} \exp\left\{\boldsymbol{\eta}(\boldsymbol{\theta})^\top \boldsymbol{a}_0\right\} \\
&= \prod_{l=1}^{m-1} \theta_l^{a_{0l}} \theta_m^{b_0 - \sum_{l=1}^{m-1} a_{0l}} \\
&= \prod_{l=1}^{m} \theta_l^{\alpha_{0l}-1},
\end{aligned}$$

where $\boldsymbol{\eta}(\boldsymbol{\theta}) = \left[\log\left(\frac{\theta_1}{\theta_m}\right)\ \ldots\ \log\left(\frac{\theta_{m-1}}{\theta_m}\right)\right]$, $\boldsymbol{a}_0 = [a_{01}\ \ldots\ a_{0m-1}]^\top$, $\boldsymbol{\alpha}_0 = [\alpha_{01}\ \alpha_{02}\ \ldots\ \alpha_{0m}]$, $\alpha_{0l} = a_{0l} + 1$, $l = 1, 2, \ldots, m-1$ and $\alpha_{0m} = b_0 - \sum_{l=1}^{m-1} a_{0l} + 1$.

This is the kernel of a Dirichlet distribution, that is, the prior distribution is $D(\boldsymbol{\alpha}_0)$.

Observe that a_{0l} is the hypothetical number of times outcome l is observed over the hypothetical b_0 trials. Setting $\alpha_{0l} = 1$, which corresponds to a uniform distribution over the open standard simplex, implicitly sets $a_{0l} = 0$, meaning that there are 0 occurrences of category l in $b_0 = 0$ experiments.

The posterior distribution of the multinomial-Dirichlet model is

given by

$$\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \prod_{l=1}^{m} \theta_l^{\alpha_{0l}-1} \times \prod_{l=1}^{m} \theta_l^{\sum_{i=1}^{N} y_{il}}$$
$$= \prod_{l=1}^{m} \theta_l^{\alpha_{0l}+\sum_{i=1}^{N} y_{il}-1}.$$

This is the kernel of a Dirichlet distribution $D(\boldsymbol{\alpha}_n)$, $\boldsymbol{\alpha}_n = [\alpha_{n1}\ \alpha_{n2}\ \ldots\ \alpha_{nm}]$, $\alpha_{nl} = \alpha_{0l} + \sum_{i=1}^{N} y_{il}$, $l = 1, 2, \ldots, m$. Observe that

$$\mathbb{E}[\theta_j \mid \boldsymbol{y}] = \frac{\alpha_{nj}}{\sum_{l=1}^{m}\left[\alpha_{0l} + \sum_{i=1}^{N} y_{il}\right]}$$
$$= \frac{\sum_{l=1}^{m} \alpha_{0l}}{\sum_{l=1}^{m}\left[\alpha_{0l} + \sum_{i=1}^{N} y_{il}\right]} \frac{\alpha_{0j}}{\sum_{l=1}^{m} \alpha_{0l}}$$
$$+ \frac{\sum_{l=1}^{m}\sum_{i=1}^{N} y_{il}}{\sum_{l=1}^{m}\left[\alpha_{0l} + \sum_{i=1}^{N} y_{il}\right]} \frac{\sum_{i=1}^{N} y_{ij}}{\sum_{l=1}^{m}\sum_{i=1}^{N} y_{il}}.$$

We have again that the posterior mean is a weighted average between the prior mean and the maximum likelihood estimate.

The marginal likelihood is

$$p(\boldsymbol{y}) = \int_{\boldsymbol{\Theta}} \frac{\prod_{l=1}^{m} \theta_l^{\alpha_{0l}-1}}{B(\boldsymbol{\alpha}_0)} \times \prod_{i=1}^{N} \frac{n!}{\prod_{l=1}^{m} y_{il}!} \prod_{l=1}^{m} \theta_l^{y_{il}} d\boldsymbol{\theta}$$
$$= \frac{N \times n!}{B(\boldsymbol{\alpha}_0) \prod_{i=1}^{N} \prod_{l=1}^{m} y_{il}!} \int_{\boldsymbol{\Theta}} \prod_{l=1}^{m} \theta_l^{\alpha_{0l}+\sum_{i=1}^{N} y_{il}-1} d\boldsymbol{\theta}$$
$$= \frac{N \times n!}{B(\boldsymbol{\alpha}_0) \prod_{i=1}^{N} \prod_{l=1}^{m} y_{il}!} B(\boldsymbol{\alpha}_n)$$
$$= \frac{N \times n! \Gamma\left(\sum_{l=1}^{m} \alpha_{0l}\right)}{\Gamma\left(\sum_{l=1}^{m} \alpha_{0l} + N \times n\right)} \prod_{l=1}^{m} \frac{\Gamma\left(\alpha_{nl}\right)}{\Gamma\left(\alpha_{0l}\right) \prod_{i=1}^{N} y_{il}!},$$

where $B(\boldsymbol{\alpha}) = \frac{\prod_{l=1}^{m} \Gamma(\alpha_l)}{\Gamma\left(\sum_{l=1}^{m} \alpha_l\right)}$.

Following similar steps we get the predictive density

$$p(y_0 \mid \boldsymbol{y}) = \frac{n! \Gamma\left(\sum_{l=1}^{m} \alpha_{nl}\right)}{\Gamma\left(\sum_{l=1}^{m} \alpha_{nl} + n\right)} \prod_{l=1}^{m} \frac{\Gamma\left(\alpha_{nl} + y_{0l}\right)}{\Gamma\left(\alpha_{nl}\right) y_{0l}!}.$$

This is a Dirichlet-multinomial distribution with parameters $\boldsymbol{\alpha}_n$.

Example: English premier league, Liverpool vs Manchester city

Let's consider an example using data from the English Premier League. In particular, we want to calculate the probability that, in the next five matches between Liverpool and Manchester City, Liverpool wins two games and Manchester City wins three. This calculation is based on historical data from the last five matches where Liverpool played at home between January 14th, 2018, and April 10th, 2022. In those matches, Liverpool secured two wins, there were two draws, and Manchester City won one match. [2]

We use two strategies to estimate the hyperparameters. First, we estimate the hyperparameters of the Dirichlet distribution using betting odds from bookmakers at 19:05 on October 6th, 2022 (Colombia time). We obtained data from 24 bookmakers (see file *DataOddsLIVvsMAN.csv*)[3], and we transform these odds into probabilities using a simple standardization approach. Then, we apply maximum likelihood estimation to estimate the hyperparameters.

Second, we use empirical Bayes, where we estimate the hyperparameters by optimizing the marginal likelihood.

[2] https://www.11v11.com/teams/manchester-city/tab/opposingTeams/opposition/Liverpool/.

[3] https://www.oddsportal.com/soccer/england/premier-league/liverpool-manchester-city-WrqgEz5S/

R code. Multinomial-Dirichlet model: Liverpool vs Manchester city

```
# Multinomial-Dirichlet example: Liverpool vs Manchester
    city
Data <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/
    DataOddsLIVvsMAN.csv", sep = ",", header = TRUE, quote =
     "")
attach(Data)
library(dplyr)
Probs <- Data %>%
  mutate(pns1 = 1/home, pns2 = 1/draw, pns3 = 1/away)%>%
  mutate(SumInvOdds = pns1 + pns2 + pns3) %>%
  mutate(p1 = pns1/SumInvOdds, p2 = pns2/SumInvOdds, p3 =
    pns3/SumInvOdds) %>%
  select(p1, p2, p3)
# We get probabilities using simple standardization. There
    are more technical approaches to do this. See for
    instance Shin (1993) and Strumbelj (2014).
DirMLE <- sirt::dirichlet.mle(Probs)
# Use maximum likelihood to estimate parameters of the
# Dirichlet distribution
alpha0odds <- DirMLE$alpha
alpha0odds
      p1       p2       p3
1599.122 1342.703 2483.129

y <- c(2, 2, 1)
# Historical records last five mathces
# Liverpool wins (2), draws (2) and Manchester
# city wins (1)

# Marginal likelihood
MarLik <- function(a0){
  n <- sum(y)
  Res1 <- sum(sapply(1:length(y),
  function(l){lgamma(a0[l]+y[l])-lgamma(a0[l])}))
  Res <- lgamma(sum(a0))-lgamma(sum(a0)+n)+Res1
  return(-Res)
}
EmpBay <- optim(alpha0odds, MarLik, method = "BFGS")
alpha0EB <- EmpBay$par
alpha0EB
p1       p2       p3
2362.622 2660.153 1279.510
# Bayes factor empirical Bayes vs betting odds.
# This is greather than 1 by construction
BF <- exp(-MarLik(alpha0EB))/exp(-MarLik(alpha0odds))
BF
2.085819
# Posterior distribution based on empirical Bayes
alphan <- alpha0EB + y
# Posterior parameters
S <- 100000
# Simulation draws from the Dirichlet distribution
thetas <- MCMCpack::rdirichlet(S, alphan)
colnames(thetas) <- c("Liverpool","Draw","Manchester")
```

R code. Multinomial-Dirichlet model: Liverpool vs Manchester city

```
# Predictive distribution based on simulations
y0 <- c(2, 0, 3)
# Liverpool two wins and Manchester city three wins in next five matches
Pred <- apply(thetas, 1, function(p) {rmultinom(1, size = sum(y0), prob = p)})
ProY0 <- sum(sapply(1:S,function(s){sum(Pred[,s]==y0)==3}))/S
ProY0
0.01157
# Probability of y0

# Predictive distribution using analytical expression
PredY0 <- function(y0){
  n <- sum(y0)
  Res1 <- sum(sapply(1:length(y), function(l){lgamma(alphan[l]+y0[l]) - lgamma(alphan[l])-lfactorial(y0[l])}))
  Res <- lfactorial(n) + lgamma(sum(alphan)) - lgamma(sum(alphan)+n) + Res1
  return(exp(Res))
}
PredY0(y0)
0.01177531
```

We observe that the Bayes factor provides evidence in favor of the hyperparameters estimated via empirical Bayes, as these hyperparameters are specifically chosen to maximize the marginal likelihood.

Using the hyperparameters obtained from empirical Bayes, we calculate that the probability of Liverpool winning two out of the next five games, while Manchester City wins three, is 1.2%. The result obtained from the predictive distribution via simulations is similar to the probability derived using the exact predictive distribution.

2. **Likelihood functions from continuous distributions**

   The normal-normal/inverse-gamma model

   Given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \ldots\ Y_N]^\top$ from a normal dis-

tribution, then the conjugate prior density has the form

$$
\begin{aligned}
\pi(\mu,\sigma^2) &\propto \exp\left\{b_0\left(-\frac{\mu^2}{2\sigma^2}-\frac{\log\sigma^2}{2}\right)\right\}\exp\left\{a_{01}\frac{\mu}{\sigma^2}-a_{02}\frac{1}{\sigma^2}\right\}\\
&= \exp\left\{b_0\left(-\frac{\mu^2}{2\sigma^2}-\frac{\log\sigma^2}{2}\right)\right\}\exp\left\{a_{01}\frac{\mu}{\sigma^2}-a_{02}\frac{1}{\sigma^2}\right\}\\
&\times \exp\left\{-\frac{a_{01}^2}{2\sigma^2 b_0}\right\}\exp\left\{\frac{a_{01}^2}{2\sigma^2 b_0}\right\}\\
&= \exp\left\{-\frac{b_0}{2\sigma^2}\left(\mu-\frac{a_{01}}{b_0}\right)^2\right\}\left(\frac{1}{\sigma^2}\right)^{\frac{b_0+1-1}{2}}\\
&\times \exp\left\{\frac{1}{\sigma^2}\frac{-2b_0a_{02}+a_{01}^2}{2b_0}\right\}\\
&= \underbrace{\left(\frac{1}{\sigma^2}\right)^{\frac{1}{2}}\exp\left\{-\frac{b_0}{2\sigma^2}\left(\mu-\frac{a_{01}}{b_0}\right)^2\right\}}_{1}\\
&\times \underbrace{\left(\frac{1}{\sigma^2}\right)^{\frac{b_0-1}{2}}\exp\left\{-\frac{1}{\sigma^2}\frac{2b_0a_{02}-a_{01}^2}{2b_0}\right\}}_{2}.
\end{aligned}
$$

The first part is the kernel of a normal density with mean $\mu_0 = \frac{a_{01}}{\beta_0}$ and variance $\frac{\sigma^2}{\beta_0}$, where $\beta_0 = b_0$. That is, $\mu \mid \sigma^2 \sim N\left(\mu_0, \frac{\sigma^2}{\beta_0}\right)$. The second part is the kernel of an inverse gamma density with shape parameter $\frac{\alpha_0}{2} = \frac{\beta_0-3}{2}$ and scale parameter $\frac{\delta_0}{2} = \frac{2\beta_0 a_{02}-a_{01}^2}{2\beta_0}$, so $\sigma^2 \sim IG\left(\frac{\alpha_0}{2}, \frac{\delta_0}{2}\right)$.

Observe that $b_0 = \beta_0$ represents the hypothetical sample size, and a_{01} is the hypothetical sum of prior observations. Therefore, it makes sense that $\frac{a_{01}}{\beta_0}$ and $\frac{\sigma^2}{\beta_0}$ represent the prior mean and variance, respectively.

Therefore, the posterior distribution is also a normal-inverse gamma

distribution,

$$
\begin{aligned}
\pi(\mu,\sigma^2 \mid \boldsymbol{y}) \propto & \left(\frac{1}{\sigma^2}\right)^{1/2} \exp\left\{-\frac{\beta_0}{2\sigma^2}(\mu-\mu_0)^2\right\} \left(\frac{1}{\sigma^2}\right)^{\alpha_0/2+1} \exp\left\{-\frac{\delta_0}{2\sigma^2}\right\} \\
& \times (\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-\mu)^2\right\} \\
& = \left(\frac{1}{\sigma^2}\right)^{1/2} \exp\left\{-\frac{1}{2\sigma^2}\left(\beta_0(\mu-\mu_0)^2 + \sum_{i=1}^{N}(y_i-\bar{y})^2 + N(\mu-\bar{y})^2 + \delta_0\right.\right. \\
& \left.\left. + \frac{(\beta_0\mu_0+N\bar{y})^2}{\beta_0+N} - \frac{(\beta_0\mu_0+N\bar{y})^2}{\beta_0+N}\right)\right\} \times \left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_0+N}{2}+1} \\
& = \underbrace{\left(\frac{1}{\sigma^2}\right)^{1/2} \exp\left\{-\frac{1}{2\sigma^2}\left((\beta_0+N)\left(\mu-\left(\frac{\beta_0\mu_0+N\bar{y}}{\beta_0+N}\right)\right)^2\right)\right\}}_{1} \\
& \times \underbrace{\left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_0+N}{2}+1} \exp\left\{-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{N}(y_i-\bar{y})^2 + \delta_0 + \frac{\beta_0 N}{\beta_0+N}(\bar{y}-\mu_0)^2\right)\right\}}_{2}.
\end{aligned}
$$

The first term is the kernel of a normal density, $\mu \mid \sigma^2, \boldsymbol{y} \sim N\left(\mu_n, \sigma_n^2\right)$, where $\mu_n = \frac{\beta_0\mu_0+N\bar{y}}{\beta_0+N}$ and $\sigma_n^2 = \frac{\sigma^2}{\beta_n}$, $\beta_n = \beta_0 + N$. The second term is the kernel of an inverse gamma density, $\sigma^2 \mid \boldsymbol{y} \sim IG(\alpha_n/2, \delta_n/2)$ where $\alpha_n = \alpha_0 + N$ and $\delta_n = \sum_{i=1}^{N}(y_i-\bar{y})^2 + \delta_0 + \frac{\beta_0 N}{\beta_0+N}(\bar{y}-\mu_0)^2$. Observe that the posterior mean is a weighted average between prior and sample information. The weights depends on the sample sizes (β_0 and N).

The marginal posterior for σ^2 is inverse gamma with shape and scale parameters $\alpha_n/2$ and $\delta_n/2$, respectively. The marginal posterior of μ is

$$
\begin{aligned}
\pi(\mu \mid \boldsymbol{y}) \propto & \int_0^\infty \left\{\left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_n+1}{2}+1} \exp\left\{-\frac{1}{2\sigma^2}(\beta_n(\mu-\mu_n)^2+\delta_n)\right\}\right\} d\sigma^2 \\
& = \frac{\Gamma\left(\frac{\alpha_n+1}{2}\right)}{\left[\frac{\beta_n(\mu-\mu_n)^2+\delta_n}{2}\right]^{\frac{\alpha_n+1}{2}}} \\
& \propto \left[\frac{\beta_n(\mu-\mu_n)^2+\delta_n}{2}\right]^{-\frac{\alpha_n+1}{2}} \left(\frac{\delta_n}{\delta_n}\right)^{-\frac{\alpha_n+1}{2}} \\
& \propto \left[\frac{\alpha_n\beta_n(\mu-\mu_n)^2}{\alpha_n\delta_n}+1\right]^{-\frac{\alpha_n+1}{2}},
\end{aligned}
$$

The second line follows from having the kernel of an inverse gamma density with parameters $\frac{\alpha_n+1}{2}$ and $\frac{1}{2}\left(\beta_n(\mu-\mu_n)^2+\delta_n\right)$.

This corresponds to the kernel of a Student's t-distribution:

$$\mu \mid \boldsymbol{y} \sim t\left(\mu_n, \frac{\delta_n}{\beta_n \alpha_n}, \alpha_n\right),$$

where $\mathbb{E}[\mu \mid \boldsymbol{y}] = \mu_n$ and

$$\mathrm{Var}[\mu \mid \boldsymbol{y}] = \frac{\alpha_n}{\alpha_n - 2}\left(\frac{\delta_n}{\beta_n \alpha_n}\right) = \frac{\delta_n}{(\alpha_n - 2)\beta_n}, \quad \alpha_n > 2.$$

Observe that the marginal posterior distribution for μ has heavier tails than the conditional posterior distribution due to the incorporation of uncertainty regarding σ^2.
The marginal likelihood is

$$\begin{aligned}
p(\boldsymbol{y}) &= \int_{-\infty}^{\infty}\int_{0}^{\infty}\left\{(2\pi\sigma^2/\beta_0)^{-1/2}\exp\left\{-\frac{1}{2\sigma^2/\beta_0}(\mu-\mu_0)^2\right\}\frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}\left(\frac{1}{\sigma^2}\right)^{\alpha_0/2+1}\right.\\
&\quad\times\left.\exp\left\{-\frac{\delta_0}{2\sigma^2}\right\}(2\pi\sigma^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-\mu)^2\right\}\right\}d\sigma^2 d\mu\\
&= \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}(2\pi)^{-\left(\frac{N+1}{2}\right)}\beta_0^{1/2}\int_{-\infty}^{\infty}\int_{0}^{\infty}\left\{\left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_0+N+1}{2}+1}\right.\\
&\quad\times\left.\exp\left\{-\frac{1}{2\sigma^2}(\beta_0(\mu-\mu_0)^2+\sum_{i=1}^{N}(y_i-\mu)^2+\delta_0)\right\}\right\}d\sigma^2 d\mu\\
&= \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}(2\pi)^{-\left(\frac{N+1}{2}\right)}\beta_0^{1/2}\Gamma\left(\frac{N+1+\alpha_0}{2}\right)\\
&\quad\times\int_{-\infty}^{\infty}\left[\frac{\beta_0(\mu-\mu_0)^2+\sum_{i=1}^{N}(y_i-\mu)^2+\delta_0}{2}\right]^{-\frac{\alpha_0+N+1}{2}}d\mu\\
&= \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}(2\pi)^{-\left(\frac{N+1}{2}\right)}\beta_0^{1/2}\Gamma\left(\frac{N+1+\alpha_0}{2}\right)\\
&\quad\times\int_{-\infty}^{\infty}\left[\frac{\beta_n(\mu-\mu_n)^2+\delta_n}{2}\right]^{-\frac{\alpha_n+1}{2}}d\mu\left(\frac{\delta_n/2}{\delta_n/2}\right)^{-\frac{\alpha_n+1}{2}}\\
&= \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}(2\pi)^{-\left(\frac{N+1}{2}\right)}\beta_0^{1/2}\Gamma\left(\frac{\alpha_n+1}{2}\right)\left(\frac{\delta_n}{2}\right)^{-\frac{\alpha_n+1}{2}}\frac{\left(\frac{\delta_n\pi}{\beta_n}\right)^{1/2}\Gamma\left(\frac{\alpha_n}{2}\right)}{\Gamma\left(\frac{\alpha_n+1}{2}\right)}\\
&= \frac{\Gamma\left(\frac{\alpha_n}{2}\right)}{\Gamma\left(\frac{\alpha_0}{2}\right)}\frac{(\delta_0/2)^{\alpha_0/2}}{(\delta_n/2)^{\alpha_n/2}}\left(\frac{\beta_0}{\beta_n}\right)^{1/2}(\pi)^{-N/2},
\end{aligned}$$

where we take into account that $\int_{-\infty}^{\infty}\left[\frac{\beta_n(\mu-\mu_n)^2+\delta_n}{2}\right]^{-\frac{\alpha_n+1}{2}}d\mu\left(\frac{\delta_n/2}{\delta_n/2}\right)^{-\frac{\alpha_n+1}{2}} = \int_{-\infty}^{\infty}\left[\frac{\beta_n\alpha_n(\mu-\mu_n)^2}{\delta_n\alpha_n}+1\right]^{-\frac{\alpha_n+1}{2}}d\mu\left(\frac{\delta_n}{2}\right)^{-\frac{\alpha_n+1}{2}}$. The term in the integral is the kernel of a Student's t density, this means that the integral is equal to $\frac{\left(\frac{\delta_n\pi}{\beta_n}\right)^{1/2}\Gamma\left(\frac{\alpha_n}{2}\right)}{\Gamma\left(\frac{\alpha_n+1}{2}\right)}$.

The predictive density is

$$
\begin{aligned}
\pi(y_0 \mid \boldsymbol{y}) \propto & \int_{-\infty}^{\infty}\int_{0}^{\infty}\left\{\left(\frac{1}{\sigma^2}\right)^{1/2}\exp\left\{-\frac{1}{2\sigma^2}(y_0-\mu)^2\right\}\left(\frac{1}{\sigma^2}\right)^{1/2}\exp\left\{-\frac{\beta_n}{2\sigma^2}(\mu-\mu_n)^2\right\}\right. \\
& \left.\times\left(\frac{1}{\sigma^2}\right)^{\alpha_n/2+1}\exp\left\{-\frac{\delta_n}{2\sigma^2}\right\}\right\}d\sigma^2 d\mu \\
= & \int_{-\infty}^{\infty}\int_{0}^{\infty}\left\{\left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_n+2}{2}+1}\exp\left\{-\frac{1}{2\sigma^2}((y_0-\mu)^2+\beta_n(\mu-\mu_n)^2+\delta_n)\right\}\right\}d\sigma^2 d\mu \\
\propto & \int_{-\infty}^{\infty}\left[\beta_n(\mu-\mu_n)^2+(y_0-\mu)^2+\delta_n\right]^{-\left(\frac{\alpha_n}{2}+1\right)}d\mu \\
= & \int_{-\infty}^{\infty}\left[(\beta_n+1)\left(\mu-\left(\frac{\beta_n\mu_n+y_0}{\beta_n+1}\right)\right)^2+\frac{\beta_n(y_0-\mu_n)^2}{\beta_n+1}+\delta_n\right]^{-\left(\frac{\alpha_n}{2}+1\right)}d\mu \\
= & \int_{-\infty}^{\infty}\left[1+\frac{(\alpha_n+1)(\beta_n+1)^2\left(\mu-\left(\frac{\beta_n\mu_n+y_0}{\beta_n+1}\right)\right)^2}{(\alpha_n+1)(\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n)}\right]^{-\left(\frac{\alpha_n}{2}+1\right)}d\mu \\
& \times\left(\frac{\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n}{\beta_n+1}\right)^{-\left(\frac{\alpha_n}{2}+1\right)} \\
\propto & \left(\pi\frac{\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n}{(\beta_n+1)^2}\right)^{\frac{1}{2}}\left(\frac{\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n}{\beta_n+1}\right)^{-\left(\frac{\alpha_n}{2}+1\right)} \\
\propto & (\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n)^{-\left(\frac{\alpha_n+1}{2}\right)} \\
\propto & \left[1+\frac{\beta_n\alpha_n}{(\beta_n+1)\delta_n\alpha_n}(y_0-\mu_n)^2\right]^{-\left(\frac{\alpha_n+1}{2}\right)},
\end{aligned}
$$

where we have that $\left[1+\frac{(\alpha_n+1)(\beta_n+1)^2\left(\mu-\left(\frac{\beta_n\mu_n+y_0}{\beta_n+1}\right)\right)^2}{(\alpha_n+1)(\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n)}\right]^{-\left(\frac{\alpha_n}{2}+1\right)}$ is the kernel of a Student's t density with degrees of freedom α_n+1 and scale $\frac{\beta_n(y_0-\mu_n)^2+(\beta_n+1)\delta_n}{(\beta_n+1)^2(\alpha_n+1)}$.

The last expression is the kernel of a Student's t density, that is, $Y_0 \mid \boldsymbol{y} \sim t\left(\mu_n, \frac{(\beta_n+1)\delta_n}{\beta_n\alpha_n}, \alpha_n\right)$.

The multivariate normal-normal/inverse-Wishart model

We show in subsection 3.1 that the multivariate normal distribution is in the exponential family where

$$C(\boldsymbol{\mu},\boldsymbol{\Sigma}) = \exp\left\{-\frac{1}{2}\left(tr\left(\boldsymbol{\mu}\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\right)+\log(|\Sigma|)\right)\right\},$$

$$\eta(\boldsymbol{\mu},\boldsymbol{\Sigma})^\top = \left[\left(vec\left(\boldsymbol{\Sigma}^{-1}\right)\right)^\top \quad \left(vec\left(\boldsymbol{\mu}^\top\boldsymbol{\Sigma}^{-1}\right)\right)^\top\right],$$

$$T(\boldsymbol{y}) = \left[-\frac{1}{2}\left(vec\left(\boldsymbol{S}\right)^\top + Nvec\left(\hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^\top\right)^\top\right) \quad -N\hat{\boldsymbol{\mu}}^\top\right]^\top$$

and

$$h(\boldsymbol{y}) = (2\pi)^{-pN/2}.$$

Then, its conjugate prior distribution should have the form

$$\begin{aligned}
\pi(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &\propto \exp\left\{-\frac{b_0}{2}\left(tr\left(\boldsymbol{\mu}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\right) + \log(|\Sigma|)\right)\right\} \\
&\times \exp\left\{\boldsymbol{a}_{01}^\top vec\left(\boldsymbol{\Sigma}^{-1}\right) + \boldsymbol{a}_{02}^\top vec\left(\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\right)\right\} \\
&= |\Sigma|^{-b_0/2} \exp\left\{-\frac{b_0}{2}\left(tr\left(\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right) + tr\left(\boldsymbol{a}_{02}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\right\} \\
&\times \exp\left\{\boldsymbol{a}_{01}^\top vec\left(\boldsymbol{\Sigma}^{-1}\right) + \frac{\boldsymbol{a}_{02}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{a}_{02}}{2b_0} - \frac{\boldsymbol{a}_{02}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{a}_{02}}{2b_0}\right\} \\
&= |\Sigma|^{-b_0/2} \exp\left\{-\frac{b_0}{2}\left(\boldsymbol{\mu} - \frac{\boldsymbol{a}_{02}}{b_0}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu} - \frac{\boldsymbol{a}_{02}}{b_0}\right)\right\} \\
&\times \exp\left\{-\frac{1}{2} tr\left(\left(\boldsymbol{A}_{01} - \frac{\boldsymbol{a}_{02}\boldsymbol{a}_{02}^\top}{b_0}\right)\boldsymbol{\Sigma}^{-1}\right)\right\} \\
&= \underbrace{|\Sigma|^{-1/2} \exp\left\{-\frac{b_0}{2}\left(\boldsymbol{\mu} - \frac{\boldsymbol{a}_{02}}{b_0}\right)^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu} - \frac{\boldsymbol{a}_{02}}{b_0}\right)\right\}}_{1} \\
&\times \underbrace{|\Sigma|^{-(\alpha_0+p+1)/2} \exp\left\{-\frac{1}{2} tr\left(\left(\boldsymbol{A}_{01} - \frac{\boldsymbol{a}_{02}\boldsymbol{a}_{02}^\top}{b_0}\right)\boldsymbol{\Sigma}^{-1}\right)\right\}}_{2}.
\end{aligned}$$

Here, we set $b_0 = 1 + \alpha_0 + p + 1$, and represents the hypothetical sample size, $\boldsymbol{a}_{01}$ and $\boldsymbol{a}_{02}$ are p^2-dimensional and p-dimensional vectors of prior sufficient statistics, respectively. Specifically, $\boldsymbol{a}_{01} = -\frac{1}{2}\text{vec}(\boldsymbol{A}_{01})$, where $\boldsymbol{A}_{01}$ is a $p \times p$ positive semi-definite matrix.

We observe that the first part of the last expression is the kernel of a multivariate normal density with mean $\boldsymbol{\mu}_0 = \frac{\boldsymbol{a}_{02}}{b_0}$ and covariance $\frac{\boldsymbol{\Sigma}}{b_0}$, i.e.,

$$\boldsymbol{\mu} \mid \boldsymbol{\Sigma} \sim N_p\left(\boldsymbol{\mu}_0, \frac{\boldsymbol{\Sigma}}{\beta_0}\right),$$

where $b_0 = \beta_0$. This choice of hyperparameters is intuitive because $\boldsymbol{a}_{02}$ represents the hypothetical sum of prior observations, and b_0 represents the hypothetical prior sample size.

Additionally, the second part of the last expression corresponds to the kernel of an inverse Wishart distribution with scale matrix $\boldsymbol{\Psi}_0 = \left(\boldsymbol{A}_{01} - \frac{\boldsymbol{a}_{02}\boldsymbol{a}_{02}^\top}{b_0}\right)$ and α_0 degrees of freedom, i.e.,

$$\boldsymbol{\Sigma} \sim IW_p(\boldsymbol{\Psi}_0, \alpha_0).$$

Observe that $\boldsymbol{\Psi}_0$ has the same structure as the first part of the sufficient statistics in $T(\boldsymbol{y})$, except that it should be understood as arising from prior hypothetical observations.

Therefore, the prior distribution in this setting is normal/inverse-Wishart, and, due to conjugacy, the posterior distribution belongs to the same family.

$$\begin{aligned}\pi(\boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \boldsymbol{y}) \propto (2\pi)^{-pN/2}|\boldsymbol{\Sigma}|^{-N/2} \exp\left\{-\frac{1}{2}tr\left[\left(\boldsymbol{S} + N\left(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}}\right)\left(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}}\right)^\top\right)\boldsymbol{\Sigma}^{-1}\right]\right\} \\ \times |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{-\frac{\beta_0}{2}tr\left[(\boldsymbol{\mu}-\boldsymbol{\mu}_0)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^\top\boldsymbol{\Sigma}^{-1}\right]\right\}|\boldsymbol{\Sigma}|^{-(\alpha_0+p+1)/2} \\ \times \exp\left\{-\frac{1}{2}tr(\boldsymbol{\Psi}_0\boldsymbol{\Sigma}^{-1})\right\}.\end{aligned}$$

Taking into account that

$$\begin{aligned}N\left(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}}\right)\left(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}}\right)^\top + \beta_0\left(\boldsymbol{\mu}-\boldsymbol{\mu}_0\right)\left(\boldsymbol{\mu}-\boldsymbol{\mu}_0\right)^\top &= (N+\beta_0)\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)^\top \\ &+ \frac{N\beta_0}{N+\beta_0}\left(\hat{\boldsymbol{\mu}}-\boldsymbol{\mu}_0\right)\left(\hat{\boldsymbol{\mu}}-\boldsymbol{\mu}_0\right)^\top,\end{aligned}$$

where $\boldsymbol{\mu}_n = \frac{N}{N+\beta_0}\hat{\boldsymbol{\mu}} + \frac{\beta_0}{N+\beta_0}\boldsymbol{\mu}_0$ is the posterior mean. We have

$$\begin{aligned}\pi(\boldsymbol{\mu}, \boldsymbol{\Sigma} \mid \boldsymbol{y}) \propto |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{-\frac{N+\beta_0}{2}tr\left[\left(\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)^\top\right)\boldsymbol{\Sigma}^{-1}\right]\right\} \\ \times |\boldsymbol{\Sigma}|^{-(N+\alpha_0+p+1)/2} \\ \times \exp\left\{-\frac{1}{2}tr\left[\left(\boldsymbol{\Psi}_0 + \boldsymbol{S} + \frac{N\beta_0}{N+\beta_0}(\hat{\boldsymbol{\mu}}-\boldsymbol{\mu}_0)(\hat{\boldsymbol{\mu}}-\boldsymbol{\mu}_0)^\top\right)\boldsymbol{\Sigma}^{-1}\right]\right\}.\end{aligned}$$

Then, $\boldsymbol{\mu} \mid \boldsymbol{\Sigma}, \boldsymbol{y} \sim N_p\left(\boldsymbol{\mu}_n, \frac{1}{\beta_n}\boldsymbol{\Sigma}\right)$, and $\boldsymbol{\Sigma} \mid \boldsymbol{y} \sim IW\left(\boldsymbol{\Psi}_n, \alpha_n\right)$ where $\beta_n = N + \beta_0$, $\alpha_n = N + \alpha_0$ and $\boldsymbol{\Psi}_n = \boldsymbol{\Psi}_0 + \boldsymbol{S} + \frac{N\beta_0}{N+\beta_0}(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}_0)(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}_0)^\top$.

The marginal posterior of $\boldsymbol{\mu}$ is given by $\int_{\mathcal{S}} \pi(\boldsymbol{\mu}, \boldsymbol{\Sigma})d\boldsymbol{\Sigma}$ where $\mathcal{S}$ is the space of positive semi-definite matrices. Then,

$$\begin{aligned}\pi(\boldsymbol{\mu} \mid \boldsymbol{y}) &\propto \int_{\mathcal{S}} \left\{|\boldsymbol{\Sigma}|^{-(\alpha_n+p+2)/2}\right. \\ &\quad \left.\exp\left\{-\frac{1}{2}tr\left[\left(\beta_n\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)^\top + \boldsymbol{\Psi}_n\right)\boldsymbol{\Sigma}^{-1}\right]\right\}\right\} d\boldsymbol{\Sigma} \\ &\propto \left|\left(\beta_n\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)^\top + \boldsymbol{\Psi}_n\right)\right|^{-(\alpha_n+1)/2} \\ &= \left[\left|\boldsymbol{\Psi}_n\right| \times \left|1 + \beta_n\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)^\top \boldsymbol{\Psi}_n^{-1}\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)\right|\right]^{-(\alpha_n+1)/2} \\ &\propto \left(1 + \frac{1}{\alpha_n+1-p}\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)^\top\left(\frac{\boldsymbol{\Psi}_n}{(\alpha_n+1-p)\beta_n}\right)^{-1}\left(\boldsymbol{\mu}-\boldsymbol{\mu}_n\right)\right)^{-(\alpha_n+1-p+p)/2},\end{aligned}$$

where the second line uses properties of the inverse Wishart distribution, and the third line uses a particular case of the Sylvester's determinant theorem.[4]

[4] $\det(\boldsymbol{X} + \boldsymbol{AB}) = \det(\boldsymbol{X})\det(\boldsymbol{I} + \boldsymbol{BX}^{-1}\boldsymbol{A})$.

We observe that the last line is the kernel of a multivariate t distribution, that is, $\boldsymbol{\mu} \mid \boldsymbol{y} \sim t_p(v_n, \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ where $v_n = \alpha_n + 1 - p$ and $\boldsymbol{\Sigma}_n = \frac{\boldsymbol{\Psi}_n}{(\alpha_n+1-p)\beta_n}$.

The marginal likelihood is given by

$$p(\boldsymbol{y}) = \frac{\Gamma_p\left(\frac{v_n}{2}\right)}{\Gamma_p\left(\frac{\alpha_0}{2}\right)} \frac{|\boldsymbol{\Psi}_0|^{\alpha_0/2}}{|\boldsymbol{\Psi}_n|^{\alpha_n/2}} \left(\frac{\beta_0}{\beta_n}\right)^{p/2} (2\pi)^{-Np/2},$$

where Γ_p is the multivariate gamma function (see Exercise 5).

The posterior predictive distribution is $\boldsymbol{Y}_0 \mid \boldsymbol{y} \sim t_p(v_n, \boldsymbol{\mu}_n, (\beta_n + 1)\boldsymbol{\Sigma}_n)$ (see Exercise 6).

Example: Tangency portfolio of US tech stocks

The tangency portfolio is the portfolio that maximizes the Sharpe ratio, which is defined as the excess return of a portfolio standardized by its risk.

We aim to find the portfolio weights $\boldsymbol{w}$ that maximize the Sharpe ratio, where $\mu_{i,T+\kappa} = \mathbb{E}\left(R_{i,T+\kappa} - R_{f,T+\kappa} \mid \mathcal{I}_T\right)$, with $R_{i,T+\kappa}$ and $R_{f,T+\kappa}$ representing the returns of stock i and the risk-free asset, respectively. Here, $\mu_{i,T+\kappa}$ is the expected value of the excess return at period $T + \kappa$, conditional on information available up to time T ($\mathcal{I}_T$), and $\boldsymbol{\Sigma}_{T+\kappa}$ is the covariance matrix of the excess returns, which quantifies the risk.

$$\underset{\boldsymbol{w}\in\mathbb{R}^p}{\arg\max} \frac{\boldsymbol{w}^\top \boldsymbol{\mu}_{T+\kappa}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma}_{T+\kappa} \boldsymbol{w}}}; \qquad \text{s.t} \quad \boldsymbol{w}^\top \mathbf{1} = 1,$$

where the solution is

$$\boldsymbol{w}^* = \frac{\boldsymbol{\Sigma}_{T+\kappa}^{-1} \boldsymbol{\mu}_{T+\kappa}}{\mathbf{1}^\top \boldsymbol{\Sigma}_{T+\kappa}^{-1} \boldsymbol{\mu}_{T+\kappa}}.$$

If we want to find the optimal portfolio for the next period under the assumption that the excess returns follow a multivariate normal distribution –a common assumption in these applications– we can set $\kappa = 1$ and use the predictive distribution of the excess returns. In this case, $\boldsymbol{\mu}_{T+1} = \boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_{T+1} = \frac{v_n}{v_n-2}(\beta_n + 1)\boldsymbol{\Sigma}_n$, based on the previous predictive result.

We apply this framework to ten tech stocks of the US market between January first, 2021, and September ninth, 2022. In particular, we use information from Yahoo Finance for Apple (AAPL), Netflix (NFLX), Amazon (AMZN), Microsoft (MSFT), Google (GOOG), Meta (META), Tesla (TSLA), NVIDIA Corporation (NVDA), Intel (INTC), and PayPal (PYPL).

We use non-informative hyperparameters, $\boldsymbol{\mu}_0 = \mathbf{0}$, $\beta_0 = 1$, $\boldsymbol{\Psi}_0 = 100\boldsymbol{I}$ and $\alpha_0 = p + 2$, where $p = 10$ is the number of stocks.

R code. Optimal tangency portfolio: Tech shares

```r
library(quantmod)
library(xts)
library(ggplot2)
library(gridExtra)
# grid.arrange
graphics.off()
rm(list=ls())
# Data Range
sdate <- as.Date("2021-01-01")
edate <- as.Date("2022-09-30")
Date <- seq(sdate, edate, by = "day")
tickers <- c("AAPL", "NFLX", "AMZN", "GOOG", "INTC","META",
    "MSFT", "TSLA", "NVDA", "PYPL")
p <- length(tickers)
# AAPL: Apple, NFLX: Netflix, AMZN: Amazon,
# MSFT: Microsoft, GOOG: Google, META: Meta,
# TSLA: Tesla, NVDA: NVIDIA Corporation
# INTC: Intel, PYPL: PayPal
ss_stock <- getSymbols(tickers, from=sdate, to=edate, auto.
    assign = T)
ss_stock <- purrr::map(tickers,function(x) Ad(get(x)))
ss_stock <- as.data.frame(purrr::reduce(ss_stock, merge))
colnames(ss_stock) <- tickers
# This is to get stock prices
ss_rtn <- as.data.frame(apply(ss_stock, 2, function(x) {diff
    (log(x), 1)}))
# Daily returns
t10yr <- getSymbols(Symbols = "DGS10", src = "FRED", from=
    sdate, to=edate, auto.assign = F)
# To get 10-Year US Treasury yield data from the Federal
    Reserve Electronic Database (FRED)
t10yrd <- (1 + t10yr/100)^(1/365)-1
# Daily returns
t10yrd <- t10yrd[row.names(ss_rtn)]
Exc_rtn <- as.matrix(ss_rtn) - kronecker(t(rep(1, p)), as.
    matrix(t10yrd))
# Excesses of return
df <- as.data.frame(Exc_rtn)
df$Date <- as.Date(rownames(df))
#  Get months
df$Month <- months(df$Date)
#  Get years
df$Year <- format(df$Date, format="%y")
#  Aggregate on months and year and get mean
Data <- sapply(1:p, function(i) {
  aggregate(df[, i] ~ Month + Year, df, mean)})
DataExcRtn <- matrix(0, length(Data[, 1]$Month), p)
for(i in 1:p){
  DataExcRtn[, i] <- as.numeric(Data[, i]$`df[, i]`)
}
colnames(DataExcRtn) <- tickers
head(DataExcRtn)
```

R code. Optimal tangency portfolio: Tech shares

```
# Hyperparameters #
N <- dim(DataExcRtn)[1]
mu0 <- rep(0, p)
beta0 <- 1
Psi0 <- 100 * diag(p)
alpha0 <- p + 2
# Posterior parameters #
alphan <- N + alpha0
vn <- alphan + 1 - p
muhat <- colMeans(DataExcRtn)
mun <- N/(N + beta0) * muhat + beta0/(N + beta0) * mu0
S <- t(DataExcRtn - rep(1, N)%*%t(muhat))%*%(DataExcRtn -
    rep(1, N) %*%t(muhat))
Psin <- Psi0 + S + N*beta0/(N + beta0)*(muhat - mu0)%*%t(
    muhat - mu0)
betan <- N + beta0
Sigman <- Psin/((alphan + 1 - p)*betan)
Covarn <- (Sigman * (1 + betan)) * vn / (vn - 2)
Covari <- solve(Covarn)
OptShare <- t(Covari%*%mun/as.numeric((t(rep(1, p))%*%Covari
    %*%mun)))
colnames(OptShare) <- tickers
OptShare
  AAPL  NFLX  AMZN   GOOG INTC  META MSFT    TSLA  NVDA
    PYPL
-0.019 0.248 0.102 -0.034 0.173 0.23 -0.022 -0.016 0.035
    0.301
```

We find that the optimal tangency portfolio is composed by 24.8%, 10.2%, 17.3%, 23%, 3.5% and 30.1% weights of Netflix, Amazon, Intel, Meta, NVIDIA and PayPal, and -1.9%, -3.4%, -2.2% and -1.6% weights of Apple, Google, Microsoft and Tesla. A negative weight means being short in financial jargon, that is, borrowing a stock to sell it.

3.3 Linear regression: The conjugate normal-normal/inverse gamma model

In this setting, we analyze the conjugate normal-normal/inverse gamma model, which is a cornerstone in econometrics. In this model, the dependent

variable y_i is related to a set of regressors $\boldsymbol{x}_i = [x_{i1}\ x_{i2}\ \dots\ x_{iK}]^\top$ in a linear way, that is:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_K x_{iK} + \mu_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \mu_i,$$

where $\boldsymbol{\beta} = [\beta_1\ \beta_2\ \dots\ \beta_K]^\top$ and $\mu_i \overset{iid}{\sim} N(0, \sigma^2)$ is a stochastic error such that $\mathbb{E}[\mu_i \mid \boldsymbol{x}_i] = 0$.

Defining the vectors and matrices:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1K} \\ x_{21} & x_{22} & \dots & x_{2K} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_N \end{bmatrix},$$

we can write the model in matrix form as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\mu},$$

where $\boldsymbol{\mu} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$. This implies that:

$$\boldsymbol{y} \sim N(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I}).^5$$

Thus, the likelihood function is:

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X}) &= (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\} \\ &\propto (\sigma^2)^{-\frac{N}{2}} \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\}. \end{aligned}$$

The conjugate priors for the parameters are

$$\begin{aligned} \boldsymbol{\beta} \mid \sigma^2 &\sim N(\boldsymbol{\beta}_0, \sigma^2 \boldsymbol{B}_0), \\ \sigma^2 &\sim IG(\alpha_0/2, \delta_0/2). \end{aligned}$$

[5] In regression analysis, to simplify notation, we depart from the conventional statistical notation, which defines lowercase letters as realizations of random variables, typically denoted by uppercase letters. We hope it is clear from the context when we refer to random vectors and matrices, and their realizations. Thus, we use bold lowercase letters for vectors and bold uppercase letters for matrices. This applies to the rest of the book.

Then, the posterior distribution is

$$\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}) &\propto (\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\right\} \\
&\quad \times (\sigma^2)^{-\frac{K}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_0)\right\} \\
&\quad \times \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)}\left(\frac{1}{\sigma^2}\right)^{\alpha_0/2+1} \exp\left\{-\frac{\delta_0}{2\sigma^2}\right\} \\
&\quad \propto (\sigma^2)^{-\frac{K}{2}} \exp\left\{-\frac{1}{2\sigma^2}[\boldsymbol{\beta}^\top(\boldsymbol{B}_0^{-1}+\boldsymbol{X}^\top\boldsymbol{X})\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}})]\right\} \\
&\quad \times \left(\frac{1}{\sigma^2}\right)^{(\alpha_0+N)/2+1} \exp\left\{-\frac{\delta_0 + \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\beta}_0^\top\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0}{2\sigma^2}\right\},
\end{aligned}$$

where $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$ is the maximum likelihood estimator.

Adding and subtracting $\boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$ to complete the square, where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}$ and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}})$,

$$\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}) \propto &\underbrace{(\sigma^2)^{-\frac{K}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)\right\}}_{1} \\
&\times \underbrace{(\sigma^2)^{-\left(\frac{\alpha_n}{2}+1\right)} \exp\left\{-\frac{\delta_n}{2\sigma^2}\right\}}_{2}.
\end{aligned}$$

The first expression is the kernel of a normal density function, $\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \sigma^2\boldsymbol{B}_n)$. The second expression is the kernel of a inverse gamma density, $\sigma^2 \mid \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2)$, where $\alpha_n = \alpha_0 + N$ and $\delta_n = \delta_0 + \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\beta}_0^\top\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 - \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$.

Taking into account that

$$\begin{aligned}
\boldsymbol{\beta}_n &= (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}}) \\
&= (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}},
\end{aligned}$$

where $(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{B}_0^{-1} = \boldsymbol{I}_K - (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}$ [381]. Setting $\boldsymbol{W} = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}$ we have $\boldsymbol{\beta}_n = (\boldsymbol{I}_K - \boldsymbol{W})\boldsymbol{\beta}_0 + \boldsymbol{W}\hat{\boldsymbol{\beta}}$, that is, the posterior mean of $\boldsymbol{\beta}$ is a weighted average between the sample and prior information, where the weights depend on the precision of each piece of information. Observe that when the prior covariance matrix is highly vague (non–informative), such that $\boldsymbol{B}_0^{-1} \to \boldsymbol{0}_K$, we obtain $\boldsymbol{W} \to I_K$, such that $\boldsymbol{\beta}_n \to \hat{\boldsymbol{\beta}}$, that is, the posterior mean location parameter converges to the maximum likelihood estimator.

In addition, we know that the posterior conditional covariance matrix of the location parameters $\sigma^2(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1} = \sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1} -$

$\sigma^2\left((\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{B}_0+(\boldsymbol{X}^\top\boldsymbol{X})^{-1})^{-1}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\right)$ is positive semi-definite.[6] Given that $\sigma^2(\boldsymbol{X}^\top\boldsymbol{X})^{-1}$ is the covariance matrix of the maximum likelihood estimator, we observe that prior information reduces estimation uncertainty.

Another way to see this model is by considering that both $\boldsymbol{y}$ and $\boldsymbol{\beta}$ are treated as random variables under the Bayesian framework. Thus, we can express the joint distribution of these two vectors as follows:

$$\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{y}\end{bmatrix}\sim N\left[\begin{pmatrix}\boldsymbol{\beta}_0\\ \boldsymbol{X}\boldsymbol{\beta}_0\end{pmatrix},\sigma^2\begin{pmatrix}\boldsymbol{B}_0 & \boldsymbol{B}_0\boldsymbol{X}^\top\\ \boldsymbol{X}\boldsymbol{B}_0^\top & \boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N\end{pmatrix}\right],$$

where we use that $Cov[\boldsymbol{\beta},\boldsymbol{y}]=\mathbb{E}[\boldsymbol{\beta}\boldsymbol{y}^\top]-\mathbb{E}[\boldsymbol{\beta}]\mathbb{E}[\boldsymbol{y}^\top]=\mathbb{E}[\boldsymbol{\beta}(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\mu})^\top]-\mathbb{E}[\boldsymbol{\beta}]\mathbb{E}[\boldsymbol{y}^\top]=[Var[\boldsymbol{\beta}]+\mathbb{E}[\boldsymbol{\beta}]\mathbb{E}[\boldsymbol{\beta}^\top]]\boldsymbol{X}^\top-\mathbb{E}[\boldsymbol{\beta}]\mathbb{E}[\boldsymbol{y}^\top]=\sigma^2\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{\beta}_0\boldsymbol{\beta}_0^\top\boldsymbol{X}^\top-\boldsymbol{\beta}_0\boldsymbol{\beta}_0^\top\boldsymbol{X}^\top=\sigma^2\boldsymbol{B}_0\boldsymbol{X}^\top$.

Then, we can obtain the conditional distribution of $\boldsymbol{\beta}\mid\boldsymbol{y}$ using the properties of the multivariate normal distribution. This distribution is normal with mean equal to

$$\boldsymbol{\beta}_0+\boldsymbol{B}_0\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N\right)^{-1}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}_0), \tag{3.5}$$

and covariance matrix

$$\sigma^2\left(\boldsymbol{B}_0-\boldsymbol{B}_0\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N\right)^{-1}\boldsymbol{X}\boldsymbol{B}_0^\top\right).$$

Observe that in this representation, the posterior mean is equal to the prior mean plus a correction term that takes into account the deviation between the observations and the prior expected value ($\boldsymbol{X}\boldsymbol{\beta}_0$). The weight of this correction is given by the matrix $\boldsymbol{B}_0\boldsymbol{X}^\top\left(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N\right)^{-1}$.

This form of expressing the posterior distribution is relevant for gaining some intuition on Bayesian inference in time series models within the *Gaussian linear state-space representation* in Chapter 8, also known as the Kalman filter in time series literature.

We can show that both conditional posterior distributions are the same. In particular, the posterior mean in Equation 3.5 is $[\boldsymbol{I}_K-\boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N)^{-1}\boldsymbol{X}]\boldsymbol{\beta}_0+\boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N)^{-1}\boldsymbol{y}$, where

$$\begin{aligned}\boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N)^{-1}&=\boldsymbol{B}_0\boldsymbol{X}^\top[\boldsymbol{I}_N-\boldsymbol{I}_N\boldsymbol{X}(\boldsymbol{B}_0^{-1}+\boldsymbol{X}^\top\boldsymbol{I}_N\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{I}_N]\\&=\boldsymbol{B}_0[\boldsymbol{I}_K-\boldsymbol{X}^\top\boldsymbol{I}_N\boldsymbol{X}(\boldsymbol{B}_0^{-1}+\boldsymbol{X}^\top\boldsymbol{I}_N\boldsymbol{X})^{-1}]\boldsymbol{X}^\top\\&=\boldsymbol{B}_0[\boldsymbol{I}_K-[\boldsymbol{I}_K-\boldsymbol{B}_0^{-1}(\boldsymbol{B}_0^{-1}+\boldsymbol{X}^\top\boldsymbol{I}_N\boldsymbol{X})^{-1}]]\boldsymbol{X}^\top\\&=(\boldsymbol{B}_0^{-1}+\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top,\end{aligned}$$

where the first equality uses the Woodbury matrix identity (matrix inversion lemma), and the third equality uses $\boldsymbol{D}(\boldsymbol{D}+\boldsymbol{E})^{-1}=\boldsymbol{I}-\boldsymbol{E}(\boldsymbol{D}+\boldsymbol{E})^{-1}$.

Thus, $[\boldsymbol{I}_K-\boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N)^{-1}\boldsymbol{X}]\boldsymbol{\beta}_0+\boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top+\boldsymbol{I}_N)^{-1}\boldsymbol{y}=$

[6] A particular case of the Woodbury matrix identity, $(\boldsymbol{A}+\boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1}+\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}$.

$[\boldsymbol{I}_K - (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}]\boldsymbol{\beta}_0 + (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} = [\boldsymbol{I}_K - (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}]\boldsymbol{\beta}_0 + (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}}$. Again, we see that the posterior mean is a weighted average between the prior mean, and the maximum likelihood estimator.

The equality of variances of both approaches is as follows:

$$\begin{aligned}
Var[\boldsymbol{\beta} \mid \boldsymbol{y}] &= \sigma^2(\boldsymbol{B}_0 - \boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{B}_0\boldsymbol{X}^\top + \boldsymbol{I}_N)^{-1}\boldsymbol{X}\boldsymbol{B}_0) \\
&= \sigma^2(\boldsymbol{B}_0 - \boldsymbol{B}_0\boldsymbol{X}^\top(\boldsymbol{I}_N - \boldsymbol{I}_N\boldsymbol{X}(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{I}_N\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{I}_N)\boldsymbol{X}\boldsymbol{B}_0) \\
&= \sigma^2(\boldsymbol{B}_0 - \boldsymbol{B}_0\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{B}_0 + \boldsymbol{B}_0\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{B}_0) \\
&= \sigma^2(\boldsymbol{B}_0 - \boldsymbol{B}_0\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{B}_0 + \boldsymbol{B}_0\boldsymbol{X}^\top\boldsymbol{X}[\boldsymbol{I}_K - (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{B}_0^{-1}]\boldsymbol{B}_0) \\
&= \sigma^2(\boldsymbol{B}_0 - \boldsymbol{B}_0\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}) \\
&= \sigma^2(\boldsymbol{B}_0[\boldsymbol{I}_K - \boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}]) \\
&= \sigma^2(\boldsymbol{B}_0[\boldsymbol{I}_K - (\boldsymbol{I}_K - \boldsymbol{B}_0^{-1}(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1})]) \\
&= \sigma^2(\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1},
\end{aligned}$$

where the second equality uses the Woodbury matrix identity, the fourth equality uses $(\boldsymbol{D} + \boldsymbol{E})^{-1}\boldsymbol{D} = \boldsymbol{I} - (\boldsymbol{D} + \boldsymbol{E})^{-1}\boldsymbol{E}$, and the seventh equality uses $\boldsymbol{D}(\boldsymbol{D} + \boldsymbol{E})^{-1} = \boldsymbol{I} - \boldsymbol{E}(\boldsymbol{D} + \boldsymbol{E})^{-1}$.

Now, we calculate the posterior marginal distribution of $\boldsymbol{\beta}$ following the standard approach,

$$\begin{aligned}
\pi(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) &= \int_0^\infty \pi(\boldsymbol{\beta}, \sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}) d\sigma^2 \\
&= \int_0^\infty \left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_n + K}{2} + 1} \exp\left\{-\frac{s}{2\sigma^2}\right\} d\sigma^2,
\end{aligned}$$

where $s = \delta_n + (\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)$. Then we can write

$$\begin{aligned}
\pi(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) &= \int_0^\infty \left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_n + K}{2} + 1} \exp\left\{-\frac{s}{2\sigma^2}\right\} d\sigma^2 \\
&= \frac{\Gamma((\alpha_n + K)/2)}{(s/2)^{(\alpha_n + K)/2}} \int_0^\infty \frac{(s/2)^{(\alpha_n + K)/2}}{\Gamma((\alpha_n + K)/2)} (\sigma^2)^{-(\alpha_n + K)/2 - 1} \exp\left\{-\frac{s}{2\sigma^2}\right\} d\sigma^2.
\end{aligned}$$

The right term is the integral of the probability density function of an inverse gamma distribution with parameters $\nu = (\alpha_n + K)/2$ and $\tau = s/2$. Since we are integrating over the whole support of σ^2, the integral is equal to

1, and therefore

$$\begin{aligned}
\pi(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) &= \frac{\Gamma((\alpha_n + K)/2)}{(s/2)^{(\alpha_n+K)/2}} \\
&\propto s^{-(\alpha_n+K)/2} \\
&= [\delta_n + (\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)]^{-(\alpha_n+K)/2} \\
&= \left[1 + \frac{(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \left(\frac{\delta_n}{\alpha_n}\boldsymbol{B}_n\right)^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)}{\alpha_n}\right]^{-(\alpha_n+K)/2} (\delta_n)^{-(\alpha_N+K)/2} \\
&\propto \left[1 + \frac{(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{H}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)}{\alpha_n}\right]^{-(\alpha_n+K)/2},
\end{aligned}$$

where $\boldsymbol{H}_n = \frac{\delta_n}{\alpha_n}\boldsymbol{B}_n$. This last expression is a multivariate t distribution, that is, $\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X} \sim t_K(\alpha_n, \boldsymbol{\beta}_n, \boldsymbol{H}_n)$.

Observe that as we have incorporated the uncertainty of the variance, the posterior for $\boldsymbol{\beta}$ changes from a normal to a t distribution, which has heavier tails, indicating more uncertainty.

The marginal likelihood of this model is

$$p(\boldsymbol{y}) = \int_0^\infty \int_{R^K} \pi(\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{B}_0, \boldsymbol{\beta}_0)\pi(\sigma^2 \mid \alpha_0/2, \delta_0/2)p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X})d\sigma^2 d\boldsymbol{\beta}.$$

Taking into account that $(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0) = (\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n) + m$, where $m = \boldsymbol{y}^\top \boldsymbol{y} + \boldsymbol{\beta}_0^\top \boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 - \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$, we have that

$$\begin{aligned}
p(\boldsymbol{y}) &= \int_0^\infty \int_{R^K} \pi(\boldsymbol{\beta} \mid \sigma^2)\pi(\sigma^2)p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X})d\sigma^2 d\boldsymbol{\beta} \\
&= \int_0^\infty \pi(\sigma^2)\frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}m\right\} \frac{1}{(2\pi\sigma^2)^{K/2}|\boldsymbol{B}_0|^{1/2}} \\
&\times \int_{R^K} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)\right\} d\sigma^2 d\boldsymbol{\beta} \\
&= \int_0^\infty \pi(\sigma^2)\frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}m\right\} \frac{|\boldsymbol{B}_n|^{1/2}}{|\boldsymbol{B}_0|^{1/2}} d\sigma^2 \\
&= \int_0^\infty \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)} \left(\frac{1}{\sigma^2}\right)^{\alpha_0/2+1} \exp\left\{\left(-\frac{\delta_0}{2\sigma^2}\right)\right\} \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2}m\right\} \frac{|\boldsymbol{B}_n|^{1/2}}{|\boldsymbol{B}_0|^{1/2}} d\sigma^2 \\
&= \frac{1}{(2\pi)^{N/2}} \frac{(\delta_0/2)^{\alpha_0/2}}{\Gamma(\alpha_0/2)} \frac{|\boldsymbol{B}_n|^{1/2}}{|\boldsymbol{B}_0|^{1/2}} \int_0^\infty \left(\frac{1}{\sigma^2}\right)^{\frac{\alpha_0+N}{2}+1} \exp\left\{\left(-\frac{\delta_0 + m}{2\sigma^2}\right)\right\} d\sigma^2 \\
&= \frac{1}{\pi^{N/2}} \frac{\delta_0^{\alpha_0/2}}{\delta_n^{\alpha_n/2}} \frac{|\boldsymbol{B}_n|^{1/2}}{|\boldsymbol{B}_0|^{1/2}} \frac{\Gamma(\alpha_n/2)}{\Gamma(\alpha_0/2)}.
\end{aligned}$$

We can show that $\delta_n = \delta_0 + \boldsymbol{y}^\top \boldsymbol{y} + \boldsymbol{\beta}_0^\top \boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 - \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n = \delta_0 + (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) + (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)^\top((\boldsymbol{X}^\top \boldsymbol{X})^{-1} + \boldsymbol{B}_0)^{-1}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ (see Exercise 7). Therefore, if we want to compare two models under this setting, the Bayes

factor is

$$
\begin{aligned}
BF_{12} &= \frac{p(\boldsymbol{y} \mid \mathcal{M}_1)}{p(\boldsymbol{y} \mid \mathcal{M}_2)} \\
&= \frac{\frac{\delta_{10}^{\alpha_{10}/2}}{\delta_{1n}^{\alpha_{1n}/2}} \frac{|\boldsymbol{B}_{1n}|^{1/2}}{|\boldsymbol{B}_{10}|^{1/2}} \frac{\Gamma(\alpha_{1n}/2)}{\Gamma(\alpha_{10}/2)}}{\frac{\delta_{20}^{\alpha_{20}/2}}{\delta_{2n}^{\alpha_{2n}/2}} \frac{|\boldsymbol{B}_{2n}|^{1/2}}{|\boldsymbol{B}_{20}|^{1/2}} \frac{\Gamma(\alpha_{2n}/2)}{\Gamma(\alpha_{20}/2)}},
\end{aligned}
$$

where subscripts 1 and 2 refer to each model.

Observe that, *ceteris paribus*, the model with better fit, coherence between sample and prior information regarding location parameters, higher prior to posterior precision, and fewer parameters is favored by the Bayes factor. The Bayes factor rewards model fit, as the sum of squared errors appears in δ_n; the better the fit (i.e., the lower the sum of squared errors), the better the Bayes factor. In addition, a weighted distance between sample and prior location parameters also appears in δ_n. The greater this distance, the worse the model support. The ratio of determinants between posterior and prior covariance matrices is also present; the higher this ratio, the better the Bayes factor supports a model due to information gains.

To see the effect of a model's parsimony, let's consider the common situation in applications where $\boldsymbol{B}_{j0} = c\boldsymbol{I}_{K_j}$, then $|\boldsymbol{B}_{j0}| = c^{K_j}$. Hence,

$$
\left(\frac{|\boldsymbol{B}_{20}|}{|\boldsymbol{B}_{10}|}\right)^{1/2} = \left(\frac{c^{K_2/2}}{c^{K_1/2}}\right),
$$

if $\frac{K_2}{K_1} > 1$ and $c \to \infty$ (the latter implying a non-informative prior), then $BF_{12} \to \infty$. This means infinite evidence supporting the parsimonious model, no matter what the sample information says.

Comparing models having the same number of regressors ($K_1 = K_2$) is not a safe ground, as $|\boldsymbol{B}_0|$ depends on the measurement units of the regressors. Conclusions regarding model selection depend on this, which is not a nice property. This prevents using non-informative priors when performing model selection in the Bayesian framework. However, this is not the case when $\alpha_0 \to 0$ and $\delta_0 \to 0$, which implies a non-informative prior for the variance parameter.[7]

We observe that $\Gamma(\alpha_{j0})$ cancels out, as $\alpha_0 \to 0$ implies $\alpha_{jn} \to N$, and

$$
\delta_{jn} \to (\boldsymbol{y} - \boldsymbol{X}_j\hat{\boldsymbol{\beta}}_j)^\top(\boldsymbol{y} - \boldsymbol{X}_j\hat{\boldsymbol{\beta}}_j) + (\hat{\boldsymbol{\beta}}_j - \boldsymbol{\beta}_{j0})^\top \left((\boldsymbol{X}_j^\top \boldsymbol{X}_j)^{-1} + \boldsymbol{B}_{j0}\right)^{-1} (\hat{\boldsymbol{\beta}}_j - \boldsymbol{\beta}_{j0}),
$$

when $\delta_0 \to 0$, therefore, there is no effect. This is due to σ^2 being a common parameter in both models.

In general, we can use non-informative priors for common parameters across all models, but we cannot use non-informative priors for non-common parameters when performing model selection using the Bayes factor. This issue raises the question of how to set informative priors. On one hand, we have

[7] See [151] for advice against this common practice.

those who advocate for *subjective* priors [334, 103, 364, 252]; on the other hand, those who prefer *objective* priors [29, 241, 209, 35].

Regarding the former, eliciting *subjective* priors, i.e., "formulating a person's knowledge and beliefs about one or more uncertain quantities into a (joint) probability distribution for those quantities" [145], is a very difficult task due to human beings' heuristics and biases associated with representativeness, information availability, conservatism, overconfidence, and anchoring-and-adjustment issues [403]. However, there have been good efforts using predictive and structural elicitation procedures [218, 219].

Regarding the latter, there are *reference priors* that are designed to have minimal impact on the posterior distribution and to be invariant to different parametrizations of the model [39, Chap. 5]. A remarkable example of *reference priors* is the *Jeffreys' prior* [210], which originated from the critique of *non-informative priors* that were not invariant to transformations of the parameter space. In particular, the *Jeffreys' prior* is given by:

$$\pi(\boldsymbol{\theta}) \propto |I(\boldsymbol{\theta})|^{1/2},$$

where $I(\boldsymbol{\theta}) = \mathbb{E}\left(-\frac{\partial^2 \log p(\boldsymbol{y}|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^\top}\right)$, i.e., $I(\boldsymbol{\theta})$ is the Fisher information matrix. However, the *Jeffreys' prior* is often improper, meaning it does not work well for model selection.

Thus, a standard *objective* approach is to use *intrinsic priors* [37], where a *minimal training* dataset is used with a *reference prior* to obtain a proper posterior distribution. This proper distribution is then used as a prior, and the standard Bayesian procedures are followed using the remaining dataset. In this way, we end up with meaningful Bayes factors for model selection.

Regardless of using a *subjective* or *objective* approach to define a prior distribution, it is always a good idea to assess the sensitivity of the posterior results to the prior assumptions. This is commonly done using local or pointwise assessments, such as partial derivatives [167, 207, 175] or, more often, in terms of multiple evaluations (*scenario analysis*) [340, 228, 7]. Recently, [206] extend these approaches to perform sensitivity analysis in high-dimensional hyperparameter settings.

Returning to the linear model, the posterior predictive is equal to

$$\begin{aligned}\pi(\boldsymbol{y}_0 \mid \boldsymbol{y}) &= \int_0^\infty \int_{R^K} p(\boldsymbol{Y}_0 \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{y})\pi(\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y})\pi(\sigma^2 \mid \boldsymbol{y}) d\boldsymbol{\beta} d\sigma^2 \\ &= \int_0^\infty \int_{R^K} p(\boldsymbol{Y}_0 \mid \boldsymbol{\beta}, \sigma^2)\pi(\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y})\pi(\sigma^2 \mid \boldsymbol{y}) d\boldsymbol{\beta} d\sigma^2,\end{aligned}$$

where we take into account independence between $\boldsymbol{y}_0$ and $\boldsymbol{y}$. Given $\boldsymbol{X}_0$,

which is the $N_0 \times K$ matrix of regressors associated with $\boldsymbol{y}_0$, Then,

$$\pi(\boldsymbol{y}_0 \mid \boldsymbol{y}) = \int_0^\infty \int_{R^K} \left\{ (2\pi\sigma^2)^{-\frac{N_0}{2}} \exp\left\{ -\frac{1}{2\sigma^2}(\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta})^\top (\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta})^\top \right\} \right.$$
$$\times (2\pi\sigma^2)^{-\frac{K}{2}} |\boldsymbol{B}_n|^{-1/2} \exp\left\{ -\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n) \right\}$$
$$\left. \times \frac{(\delta_n/2)^{\alpha_n/2}}{\Gamma(\alpha_n/2)} \left(\frac{1}{\sigma^2}\right)^{\alpha_n/2+1} \exp\left\{ -\frac{\delta_n}{2\sigma^2} \right\} \right\} d\boldsymbol{\beta} d\sigma^2.$$

Setting $\boldsymbol{M} = (\boldsymbol{X}_0^\top \boldsymbol{X}_0 + \boldsymbol{B}_n^{-1})$ and $\boldsymbol{\beta}_* = \boldsymbol{M}^{-1}(\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{X}_0^\top \boldsymbol{y}_0)$, we have $(\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta})^\top (\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta})^\top + (\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n) = (\boldsymbol{\beta} - \boldsymbol{\beta}_*)^\top \boldsymbol{M}(\boldsymbol{\beta} - \boldsymbol{\beta}_*) + \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{y}_0 - \boldsymbol{\beta}_*^\top \boldsymbol{M}\boldsymbol{\beta}_*$. Thus,

$$\pi(\boldsymbol{y}_0 \mid \boldsymbol{y}) \propto \int_0^\infty \left\{ \left(\frac{1}{\sigma^2}\right)^{-\frac{K+N_0+\alpha_n}{2}+1} \exp\left\{ -\frac{1}{2\sigma^2}(\boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{y}_0 - \boldsymbol{\beta}_*^\top \boldsymbol{M}\boldsymbol{\beta}_* + \delta_n) \right\} \right.$$
$$\left. \times \int_{R^K} \exp\left\{ -\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_*)^\top \boldsymbol{M}(\boldsymbol{\beta} - \boldsymbol{\beta}_*) \right\} d\boldsymbol{\beta} \right\} d\sigma^2,$$

where the term in the second integral is the kernel of a multivariate normal density with mean $\boldsymbol{\beta}_*$ and covariance matrix $\sigma^2 \boldsymbol{M}^{-1}$. Then,

$$\pi(\boldsymbol{y}_0 \mid \boldsymbol{y}) \propto \int_0^\infty \left(\frac{1}{\sigma^2}\right)^{\frac{N_0+\alpha_n}{2}+1} \exp\left\{ -\frac{1}{2\sigma^2}(\boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{y}_0 - \boldsymbol{\beta}_*^\top \boldsymbol{M}\boldsymbol{\beta}_* + \delta_n) \right\} d\sigma^2,$$

which is the kernel of an inverse gamma density. Thus,

$$\pi(\boldsymbol{y}_0 \mid \boldsymbol{y}) \propto \left[\frac{\boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{y}_0 - \boldsymbol{\beta}_*^\top \boldsymbol{M}\boldsymbol{\beta}_* + \delta_n}{2} \right]^{-\frac{\alpha_n+N_0}{2}}.$$

Setting $\boldsymbol{C}^{-1} = \boldsymbol{I}_{N_0} + \boldsymbol{X}_0 \boldsymbol{B}_n \boldsymbol{X}_0^\top$ such that $\boldsymbol{C} = \boldsymbol{I}_{N_0} - \boldsymbol{X}_0(\boldsymbol{B}_n^{-1} + \boldsymbol{X}_0^\top \boldsymbol{X}_0)^{-1}\boldsymbol{X}_0^\top = \boldsymbol{I}_{N_0} - \boldsymbol{X}_0 \boldsymbol{M}^{-1}\boldsymbol{X}_0^\top$,[8] and $\boldsymbol{\beta}_{**} = \boldsymbol{C}^{-1}\boldsymbol{X}_0 \boldsymbol{M}^{-1}\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$, then

$$\begin{aligned}
\boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{y}_0 - \boldsymbol{\beta}_*^\top \boldsymbol{M}\boldsymbol{\beta}_* &= \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{y}_0 - (\boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1} + \boldsymbol{y}_0^\top \boldsymbol{X}_0)\boldsymbol{M}^{-1}(\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{X}_0^\top \boldsymbol{y}_0) \\
&= \boldsymbol{\beta}_n^\top (\boldsymbol{B}_n^{-1} - \boldsymbol{B}_n^{-1}\boldsymbol{M}^{-1}\boldsymbol{B}_n^{-1})\boldsymbol{\beta}_n + \boldsymbol{y}_0^\top \boldsymbol{C}\boldsymbol{y}_0 \\
&\quad - 2\boldsymbol{y}_0^\top \boldsymbol{C}\boldsymbol{C}^{-1}\boldsymbol{X}_0 \boldsymbol{M}^{-1}\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{\beta}_{**}^\top \boldsymbol{C}\boldsymbol{\beta}_{**} - \boldsymbol{\beta}_{**}^\top \boldsymbol{C}\boldsymbol{\beta}_{**} \\
&= \boldsymbol{\beta}_n^\top (\boldsymbol{B}_n^{-1} - \boldsymbol{B}_n^{-1}\boldsymbol{M}^{-1}\boldsymbol{B}_n^{-1})\boldsymbol{\beta}_n + (\boldsymbol{y}_0 - \boldsymbol{\beta}_{**})^\top \boldsymbol{C}(\boldsymbol{y}_0 - \boldsymbol{\beta}_{**}) \\
&\quad - \boldsymbol{\beta}_{**}^\top \boldsymbol{C}\boldsymbol{\beta}_{**},
\end{aligned}$$

where $\boldsymbol{\beta}_n^\top (\boldsymbol{B}_n^{-1} - \boldsymbol{B}_n^{-1}\boldsymbol{M}^{-1}\boldsymbol{B}_n^{-1})\boldsymbol{\beta}_n = \boldsymbol{\beta}_{**}^\top \boldsymbol{C}\boldsymbol{\beta}_{**}$ and $\boldsymbol{\beta}_{**} = \boldsymbol{X}_0\boldsymbol{\beta}_n$ (see Exercise 8).

[8] Using $(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}\boldsymbol{C})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}^{-1} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}$

Then,

$$\pi(\boldsymbol{y}_0 \mid \boldsymbol{y}) \propto \left[\frac{(\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta}_n)^\top \boldsymbol{C}(\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta}_n) + \delta_n}{2}\right]^{-\frac{\alpha_n + N_0}{2}}$$
$$\propto \left[\frac{(\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta}_n)^\top \left(\frac{\boldsymbol{C}\alpha_n}{\delta_n}\right)(\boldsymbol{y}_0 - \boldsymbol{X}_0\boldsymbol{\beta}_n)}{\alpha_n} + 1\right]^{-\frac{\alpha_n + N_0}{2}}.$$

The posterior predictive is a multivariate t distribution, $\boldsymbol{y}_0 \mid \boldsymbol{y} \sim t\left(\boldsymbol{X}_0\boldsymbol{\beta}_n, \frac{\delta_n(\boldsymbol{I}_{N_0} + \boldsymbol{X}_0\boldsymbol{B}_n\boldsymbol{X}_0^\top)}{\alpha_n}, \alpha_n\right)$.

Example: Demand of electricity

We study in this example the determinants of the monthly demand for electricity by Colombian households. The data consists of information from 2103 households, including the following variables: the average price (USD/kWh), indicators of the socioeconomic conditions of the neighborhood where the household is located (with `IndSocio1` being the lowest and `IndSocio3` being the highest), an indicator for whether the household is located in a municipality that is above 1000 meters above sea level, the number of rooms in the house, the number of members in the household, the presence of children in the household (where 1 indicates yes), and the monthly income (USD). The specification is as follows:

$$\begin{aligned}\log(\text{Electricity}_i) = {} & \beta_1 \log(\text{price}_i) + \beta_2 \text{IndSocio1}_i + \beta_3 \text{IndSocio2}_i + \beta_4 \text{Altitude}_i \\ & + \beta_5 \text{Nrooms}_i + \beta_6 \text{HouseholdMem}_i + \beta_7 \text{Children}_i \\ & + \beta_8 \log(\text{Income}_i) + \beta_9 + \mu_i.\end{aligned}$$

We use a non-informative vague prior setting such that $\alpha_0 = \delta_0 = 0.001$, $\boldsymbol{\beta}_0 = \mathbf{0}$ and $\boldsymbol{B}_0 = c_0\boldsymbol{I}_K$, where $c_0 = 1000$ and K is the number of regressors.

The results from the **R** code (see below) indicate that the posterior mean of the own-price elasticity of electricity demand is -1.09, and the 95% symmetric credible interval is (-1.47, -0.71). Households in neighborhoods with low socioeconomic conditions and those located in municipalities situated more than 1000 meters above sea level consume less electricity, with reductions of 32.7% and 19.7% on average, respectively. An additional room leads to an 8.7% increase in electricity consumption, and each additional household member increases consumption by 5.9% on average. The mean estimate for income elasticity is 0.074, meaning that a 10% increase in income results in a 0.74% increase in electricity demand.

We want to check the results of the Bayes factor comparing the previous specification (model 1) with other specification without considering the price

of electricity (model 2), that is,

$$\begin{aligned}\log(\text{Electricity}_i) = {} & \beta_1 \text{IndSocio1}_i + \beta_2 \text{IndSocio2}_i + \beta_3 \text{Altitude}_i + \beta_4 \text{Nrooms}_i \\ & + \beta_5 \text{HouseholdMem}_i + \beta_6 \text{Children}_i + \beta_7 \log(\text{Income}_i) \\ & + \beta_8 + \mu_i.\end{aligned}$$

In particular, we examine what happens as c_0 increases from 10^0 to 10^{20}. We observe that when $c_0 = 1$, $BF_{12} = 8.68 \times 10^{+16}$, which indicates very strong evidence in favor of the model including the price of electricity. However, as c_0 increases, the Bayes factor decreases, which suggests evidence supporting model 2. For instance, when $c_0 = 10^{20}$, $BF_{12} = 3.11 \times 10^{-4}$. This is an example of the issue with using non-informative priors to calculate the Bayes factor: there is very strong evidence supporting the parsimonious model as $c_0 \to \infty$.

We can obtain the posterior predictive distribution of the monthly electricity demand for a household located in the lowest socioeconomic condition in a municipality situated below 1000 meters above sea level, with 2 rooms, 3 members (with children), a monthly income of USD 500, and an electricity price of USD 0.15/kWh. Figure 3.1 shows the histogram of the predictive posterior distribution. The highest posterior density credible interval at 95% is between 44.4 kWh and 373.9 kWh, and the posterior mean is 169.4 kWh.

R code. Demand of electricity, posterior predictive distribution

```
rm(list = ls())
set.seed(010101)
# Electricity demand
DataUt <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/Utilities.csv", sep = ",", header = TRUE, quote = "")
library(dplyr)
DataUtEst <- DataUt %>%
  filter(Electricity != 0)
attach(DataUtEst)
# Dependent variable: Monthly consumption (kWh) in log
Y <- log(Electricity)
# Regressors quantity including intercept
X <- cbind(LnPriceElect, IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem, Children, Lnincome, 1)
# LnPriceElect: Price per kWh (USD) in log
# IndSocio1, IndSocio2, IndSocio3: Indicators socio-economic condition (1) is the lowest and (3) the highest
# Altitude: Indicator of household location (1 is more than 1000 meters above sea level)
# Nrooms: Number of rooms in house
# HouseholdMem: Number of household members
# Children: Indicator por presence of children in household (1)
# Lnincome: Monthly income (USD) in log
k <- dim(X)[2]
N <- dim(X)[1]
# Hyperparameters
d0 <- 0.001
a0 <- 0.001
b0 <- rep(0, k)
B0 <- 1000*diag(k)
# Posterior parameters
bhat <- solve(t(X)%*%X)%*%t(X)%*%Y
Bn <- as.matrix(Matrix::forceSymmetric(solve(solve(B0) + t(X)%*%X))) # Force this matrix to be symmetric
bn <- Bn%*%(solve(B0)%*%b0 + t(X)%*%X%*%bhat)
dn <- as.numeric(d0 + t(Y)%*%Y+t(b0)%*%solve(B0)%*%b0-t(bn)%*%solve(Bn)%*%bn)
an <- a0 + N
Hn <- Bn*dn/an
# Posterior draws
S <- 10000 # Number of draws from posterior distributions
sig2 <- MCMCpack::rinvgamma(S,an/2,dn/2)
summary(coda::mcmc(sig2))
```

R code. Demand of electricity, posterior distribution

```
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each
variable, plus standard error of the mean:

Mean             SD       Naive SE  Time-series SE
2.361e-01      7.617e-03  7.617e-05 7.617e-05

2. Quantiles for each variable:

2.5%    25%    50%    75%  97.5%
0.2217 0.2309 0.2360 0.2412 0.2513

Betas <- LaplacesDemon::rmvt(S, bn, Hn, an)
summary(coda::mcmc(Betas))
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each
variable, plus standard error of the mean:

               Mean      SD    Naive SE  Time-series SE
LnPriceElect -1.09043 0.19459 0.0019459 0.0019459
IndSocio1    -0.32783 0.05294 0.0005294 0.0005294
IndSocio2    -0.05737 0.04557 0.0004557 0.0004557
Altitude     -0.19780 0.02386 0.0002386 0.0002429
Nrooms        0.08731 0.01094 0.0001094 0.0001119
HouseholdMem  0.05987 0.01334 0.0001334 0.0001334
Children      0.05696 0.03043 0.0003043 0.0003043
Lnincome      0.07447 0.01223 0.0001223 0.0001223
              2.52296 0.35077 0.0035077 0.0035077

2. Quantiles for each variable:

                   2.5%      25%      50%      75%    97.5%
LnPriceElect -1.472069 -1.22432 -1.08961 -0.95703 -0.71429
IndSocio1    -0.435957 -0.36228 -0.32731 -0.29133 -0.22588
IndSocio2    -0.147252 -0.08744 -0.05757 -0.02650  0.03254
Altitude     -0.244759 -0.21372 -0.19783 -0.18164 -0.15094
Nrooms        0.066432  0.07985  0.08709  0.09480  0.10864
HouseholdMem  0.033623  0.05089  0.05975  0.06889  0.08596
Children     -0.002259  0.03637  0.05698  0.07736  0.11681
Lnincome      0.050536  0.06614  0.07449  0.08283  0.09852
              1.835507  2.28703  2.52165  2.76364  3.21199
```

R code. Demand of electricity, Bayes factor and predictive

```
# Log marginal function (multiply by -1 due to minimization)
LogMarLikLM <- function(X, c0){
  k <- dim(X)[2]
  N <- dim(X)[1]
  # Hyperparameters
  B0 <- c0*diag(k)
  b0 <- rep(0, k)
  # Posterior parameters
  bhat <- solve(t(X)%*%X)%*%t(X)%*%Y
  # Force this matrix to be symmetric
  Bn <- as.matrix(Matrix::forceSymmetric(solve(solve(B0) + t(X)%*%X)))
  bn <- Bn%*%(solve(B0)%*%b0 + t(X)%*%X%*%bhat)
  dn <- as.numeric(d0 + t(Y)%*%Y+t(b0)%*%solve(B0)%*%b0-t(bn)%*%solve(Bn)%*%bn)
  an <- a0 + N
  # Log marginal likelihood
  logpy <- (N/2)*log(1/pi)+(a0/2)*log(d0)-(an/2)*log(dn) + 0.5*log(det(Bn)/det(B0)) + lgamma(an/2)-lgamma(a0/2)
  return(-logpy)
}
cs <- c(10^0, 10^3, 10^6, 10^10, 10^12, 10^15, 10^20)
# Observe -1 to recover the right sign
LogML <- sapply(cs, function(c) {-LogMarLikLM(c0=c, X = X)})
# Regressor without price
Xnew <- cbind(IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem, Children, Lnincome, 1)
# Observe -1 to recover the right sign
LogMLnew <- sapply(cs, function(c) {-LogMarLikLM(c0=c,X = Xnew)})
# Bayes factor
BF <- exp(LogML - LogMLnew)
BF
8.687567e+16 1.006679e+05 3.108415e+03 3.108340e+01 3.108343e+00 9.829443e-02 3.108343e-04
# Predictive distribution
Xpred <- c(log(0.15), 1, 0, 0, 2, 3, 1, log(500), 1)
Mean <- Xpred%*%bn
Hn <- dn*(1+t(Xpred)%*%Bn%*%Xpred)/an
ExpKwH <- exp(LaplacesDemon::rmvt(S, Mean, Hn, an))
summary(ExpKwH)
Min.   :  24.06
1st Qu.: 121.70
Median : 169.37
Mean   : 189.60
3rd Qu.: 234.19
Max.   :1243.68
HDI <- HDInterval::hdi(ExpKwH, credMass = 0.95) # Highest posterior density credible interval
HDI
lower  44.40203
upper 373.86494
hist(ExpKwH, main = "Histogram: Monthly demand of electricity", xlab = "Monthly kWh", col = "blue", breaks = 50)
```

FIGURE 3.1
Histogram using the posterior predictive distribution of electricity demand

3.4 Multivariate linear regression: The conjugate normal-normal/inverse Wishart model

Let's study the multivariate regression setting where there are N-dimensional vectors $\boldsymbol{y}_m$, for $m = 1, 2, \ldots, M$, such that $\boldsymbol{y}_m = \boldsymbol{X}\boldsymbol{\beta}_m + \boldsymbol{\mu}_m$. Here, $\boldsymbol{X}$ represents the set of common regressors, and $\boldsymbol{\mu}_m$ is the N-dimensional vector of stochastic errors for each equation. We assume that $\boldsymbol{U} = [\boldsymbol{\mu}_1 \ \boldsymbol{\mu}_2 \ \ldots \ \boldsymbol{\mu}_M] \sim MN_{N,M}(\mathbf{0}, \boldsymbol{I}_N, \boldsymbol{\Sigma})$, which is a matrix variate normal distribution where $\boldsymbol{\Sigma}$ is the covariance matrix of each i-th row of $\boldsymbol{U}$, for $i = 1, 2, \ldots, N$, and we assume independence between the rows. Consequently, we have that $vec(\boldsymbol{U}) \sim N_{N \times M}(\mathbf{0}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_N)$.[9]

[9] vec denotes the vectorization operation, and $\otimes$ denotes the Kronecker product.

This framework can be written in matrix form

$$\underbrace{\begin{bmatrix} y_{11} & y_{12} & \dots & y_{1M} \\ y_{21} & y_{22} & \dots & y_{2M} \\ \vdots & \vdots & \dots & \vdots \\ y_{N1} & y_{N2} & \dots & y_{NM} \end{bmatrix}}_{\boldsymbol{Y}} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1K} \\ x_{21} & x_{22} & \dots & x_{2K} \\ \vdots & \vdots & \dots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NK} \end{bmatrix}}_{\boldsymbol{X}} \underbrace{\begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1M} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2M} \\ \vdots & \vdots & \dots & \vdots \\ \beta_{K1} & \beta_{K2} & \dots & \beta_{KM} \end{bmatrix}}_{\boldsymbol{B}}$$

$$+ \underbrace{\begin{bmatrix} \mu_{11} & \mu_{12} & \dots & \mu_{1M} \\ \mu_{21} & \mu_{22} & \dots & \mu_{2M} \\ \vdots & \vdots & \dots & \vdots \\ \mu_{N1} & \mu_{N2} & \dots & \mu_{NM} \end{bmatrix}}_{\boldsymbol{U}}.$$

Therefore, $\boldsymbol{Y} \sim N_{N\times M}(\boldsymbol{X}\boldsymbol{B}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_N)$,[10]

$$\begin{aligned} p(\boldsymbol{Y} \mid \boldsymbol{B}, \boldsymbol{\Sigma}, \boldsymbol{X}) &\propto |\boldsymbol{\Sigma}|^{-N/2} \exp\left\{-\frac{1}{2} tr\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B})^\top(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{B})\boldsymbol{\Sigma}^{-1}\right]\right\} \\ &= |\boldsymbol{\Sigma}|^{-N/2} \exp\left\{-\frac{1}{2} tr\left[\left(\boldsymbol{S} + (\boldsymbol{B}-\widehat{\boldsymbol{B}})^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{B}-\widehat{\boldsymbol{B}})\right)\boldsymbol{\Sigma}^{-1}\right]\right\}, \end{aligned}$$

where $\boldsymbol{S} = (\boldsymbol{Y}-\boldsymbol{X}\widehat{\boldsymbol{B}})^\top(\boldsymbol{Y}-\boldsymbol{X}\widehat{\boldsymbol{B}})$, $\widehat{\boldsymbol{B}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{Y}$ (see Exercise 9).

The conjugate prior for this model is $\pi(\boldsymbol{B}, \boldsymbol{\Sigma}) = \pi(\boldsymbol{B} \mid \boldsymbol{\Sigma})\pi(\boldsymbol{\Sigma})$ where $\boldsymbol{B} \mid \boldsymbol{\Sigma} \sim N_{K\times M}(\boldsymbol{B}_0, \boldsymbol{V}_0, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \sim IW(\boldsymbol{\Psi}_0, \alpha_0)$, that is,

$$\begin{aligned} \pi(\boldsymbol{B}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-K/2} &\exp\left\{-\frac{1}{2} tr\left[(\boldsymbol{B}-\boldsymbol{B}_0)^\top \boldsymbol{V}_0^{-1}(\boldsymbol{B}-\boldsymbol{B}_0)\boldsymbol{\Sigma}^{-1}\right]\right\} \\ &\times |\boldsymbol{\Sigma}|^{-(\alpha_0+M+1)/2} \exp\left\{-\frac{1}{2} tr\left[\boldsymbol{\Psi}_0\boldsymbol{\Sigma}^{-1}\right]\right\}. \end{aligned}$$

The posterior distribution is given by

$$\begin{aligned} \pi(\boldsymbol{B}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X}) &\propto p(\boldsymbol{Y} \mid \boldsymbol{B}, \boldsymbol{\Sigma}, \boldsymbol{X})\pi(\boldsymbol{B} \mid \boldsymbol{\Sigma})\pi(\boldsymbol{\Sigma}) \\ &\propto |\boldsymbol{\Sigma}|^{-\frac{N+K+\alpha_0+M+1}{2}} \\ &\times \exp\left\{-\frac{1}{2} tr\left[(\boldsymbol{\Psi}_0 + \boldsymbol{S} + (\boldsymbol{B}-\boldsymbol{B}_0)^\top \boldsymbol{V}_0^{-1}(\boldsymbol{B}-\boldsymbol{B}_0)\right.\right. \\ &\left.\left. +(\boldsymbol{B}-\widehat{\boldsymbol{B}})^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{B}-\widehat{\boldsymbol{B}}))\boldsymbol{\Sigma}^{-1}\right]\right\}. \end{aligned}$$

Completing the squares on $\boldsymbol{B}$ and collecting the remaining terms in the bracket yields

$$\boldsymbol{\Psi}_0 + \boldsymbol{S} + (\boldsymbol{B}-\boldsymbol{B}_0)^\top \boldsymbol{V}_0^{-1}(\boldsymbol{B}-\boldsymbol{B}_0) + (\boldsymbol{B}-\widehat{\boldsymbol{B}})^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{B}-\widehat{\boldsymbol{B}}) = (\boldsymbol{B}-\boldsymbol{B}_n)^\top \boldsymbol{V}_n^{-1}(\boldsymbol{B}-\boldsymbol{B}_n) + \boldsymbol{\Psi}_n,$$

[10] We can write down the former expression in a more familiar way using vectorization properties, $\underbrace{vec(Y)}_{\boldsymbol{y}} = \underbrace{(\boldsymbol{I}_M \otimes \boldsymbol{X})}_{\boldsymbol{Z}} \underbrace{vec(\boldsymbol{B})}_{\boldsymbol{\beta}} + \underbrace{vec(\boldsymbol{U})}_{\mu}$, where $\boldsymbol{y} \sim N_{N\times M}(\boldsymbol{Z}\boldsymbol{\beta}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_N)$.

where

$$\begin{aligned}
\boldsymbol{B}_n =& (\boldsymbol{V}_0^{-1} + \boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \boldsymbol{X}^\top \boldsymbol{Y}) = (\boldsymbol{V}_0^{-1} + \boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \boldsymbol{X}^\top \boldsymbol{X}\widehat{\boldsymbol{B}}),\\
\boldsymbol{V}_n =& (\boldsymbol{V}_0^{-1} + \boldsymbol{X}^\top \boldsymbol{X})^{-1},\\
\boldsymbol{\Psi}_n =& \boldsymbol{\Psi}_0 + \boldsymbol{S} + \boldsymbol{B}_0^\top \boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \widehat{\boldsymbol{B}}^\top \boldsymbol{X}^\top \boldsymbol{X}\widehat{\boldsymbol{B}} - \boldsymbol{B}_n^\top \boldsymbol{V}_n^{-1}\boldsymbol{B}_n.
\end{aligned}$$

Thus, the posterior distribution can be written as

$$\begin{aligned}
\pi(\boldsymbol{B}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X}) \propto& |\boldsymbol{\Sigma}|^{-K/2} \exp\left\{-\frac{1}{2} tr\left[(\boldsymbol{B} - \boldsymbol{B}_n)^\top \boldsymbol{V}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n)\boldsymbol{\Sigma}^{-1}\right]\right\}\\
&\times |\boldsymbol{\Sigma}|^{-\frac{N+\alpha_0+M+1}{2}} \exp\left\{-\frac{1}{2} tr\left[\boldsymbol{\Psi}_n \boldsymbol{\Sigma}^{-1}\right]\right\}.
\end{aligned}$$

That is $\pi(\boldsymbol{B}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X}) = \pi(\boldsymbol{B} \mid \boldsymbol{\Sigma}, \boldsymbol{Y}, \boldsymbol{X})\pi(\boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X})$ where $\boldsymbol{B} \mid \boldsymbol{\Sigma}, \boldsymbol{Y}, \boldsymbol{X} \sim N_{K\times M}(\boldsymbol{B}_n, \boldsymbol{V}_n, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X} \sim IW(\boldsymbol{\Psi}_n, \alpha_n)$, $\alpha_n = N + \alpha_0$. Observe again that we can write down the posterior mean as a weighted average between prior and sample information such that $\boldsymbol{V}_0 \to \infty$ implies $\boldsymbol{B}_n \to \hat{\boldsymbol{B}}$, as we show in the univariate linear model.

The marginal posterior for $\boldsymbol{B}$ is given by

$$\begin{aligned}
\pi(\boldsymbol{B} \mid \boldsymbol{Y}, \boldsymbol{X}) \propto& \int_{\boldsymbol{\mathcal{S}}} |\boldsymbol{\Sigma}|^{-(\alpha_n+K+M+1)/2}\\
&\times \exp\left\{-\frac{1}{2} tr\left\{\left[(\boldsymbol{B} - \boldsymbol{B}_n)^\top \boldsymbol{V}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n) + \boldsymbol{\Psi}_n\right]\boldsymbol{\Sigma}^{-1}\right\}\right\} d\boldsymbol{\Sigma}\\
&\propto |(\boldsymbol{B} - \boldsymbol{B}_n)^\top \boldsymbol{V}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n) + \boldsymbol{\Psi}_n|^{-(K+\alpha_n)/2}\\
&= \left[|\boldsymbol{\Psi}_n| \times |\boldsymbol{I}_K + \boldsymbol{V}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n)\boldsymbol{\Psi}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n)^\top|\right]^{-(\alpha_n+1-M+K+M-1)/2}\\
&\propto |\boldsymbol{I}_K + \boldsymbol{V}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n)\boldsymbol{\Psi}_n^{-1}(\boldsymbol{B} - \boldsymbol{B}_n)^\top|^{-(\alpha_n+1-M+K+M-1)/2}.
\end{aligned}$$

The second line uses the inverse Wishart distribution, the third line the Sylverter's theorem, and the last line is the kernel of a matrix t-distribution, that is, $\boldsymbol{B} \mid \boldsymbol{Y}, \boldsymbol{X} \sim T_{K\times M}(\boldsymbol{B}_n, \boldsymbol{V}_n, \boldsymbol{\Psi}_n)$ with $\alpha_n + 1 - M$ degrees of freedom.

Observe that $vec(\boldsymbol{B})$ has mean $vec(\boldsymbol{B}_n)$ and variance $(\boldsymbol{V}_n \otimes \boldsymbol{\Psi}_n)/(\alpha_n - M - 1)$ based on its marginal distribution. On the other hand, the variance based on the conditional distribution is $\boldsymbol{V}_n \otimes \boldsymbol{\Sigma}$, where the mean of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Psi}_n/(\alpha_n - M - 1)$.

The marginal likelihood is the following,

$$
\begin{aligned}
p(\boldsymbol{Y}) &= \int_{\mathcal{B}}\int_{\mathcal{S}} \left\{ (2\pi)^{-NM/2} |\boldsymbol{\Sigma}|^{-N/2} \exp\left\{ -\frac{1}{2} tr\left[\boldsymbol{S} + (\boldsymbol{B}-\widehat{\boldsymbol{B}})^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{B}-\widehat{\boldsymbol{B}}) \right] \boldsymbol{\Sigma}^{-1} \right\} \right. \\
&\times (2\pi)^{-KM/2} |\boldsymbol{V}_0|^{-M/2} |\boldsymbol{\Sigma}|^{-K/2} \exp\left\{ -\frac{1}{2} tr\left[(\boldsymbol{B}-\boldsymbol{B}_0)^\top \boldsymbol{V}_0^{-1} (\boldsymbol{B}-\boldsymbol{B}_0) \boldsymbol{\Sigma}^{-1} \right] \right\} \\
&\left. \times \frac{|\Psi_0|^{\alpha_0/2}}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} |\boldsymbol{\Sigma}|^{-(\alpha_0+M+1)/2} \exp\left\{ -\frac{1}{2} tr\left[\boldsymbol{\Psi}_0 \boldsymbol{\Sigma}^{-1} \right] \right\} \right\} d\boldsymbol{\Sigma} d\boldsymbol{B} \\
&= (2\pi)^{-M(N+K)/2} |\boldsymbol{V}_0|^{-M/2} \frac{|\Psi_0|^{\alpha_0/2}}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} \\
&\times \int_{\mathcal{B}}\int_{\mathcal{S}} \left\{ |\boldsymbol{\Sigma}|^{-(\alpha_0+N+K+M+1)/2} \right. \\
&\left. \exp\left\{ -\frac{1}{2} tr\left[\boldsymbol{S} + (\boldsymbol{B}-\widehat{\boldsymbol{B}})^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{B}-\widehat{\boldsymbol{B}}) + (\boldsymbol{B}-\boldsymbol{B}_0)^\top \boldsymbol{V}_0^{-1}(\boldsymbol{B}-\boldsymbol{B}_0) + \boldsymbol{\Psi}_0 \right] \boldsymbol{\Sigma}^{-1} \right\} \right\} d\boldsymbol{\Sigma} d\boldsymbol{B} \\
&= (2\pi)^{-M(N+K)/2} |\boldsymbol{V}_0|^{-M/2} \frac{|\Psi_0|^{\alpha_0/2}}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} 2^{M(\alpha_n+K)/2} \Gamma_M((\alpha_n+K)/2) \\
&\times \int_{\mathcal{B}} \left| \boldsymbol{S} + (\boldsymbol{B}-\widehat{\boldsymbol{B}})^\top \boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{B}-\widehat{\boldsymbol{B}}) + (\boldsymbol{B}-\boldsymbol{B}_0)^\top \boldsymbol{V}_0^{-1}(\boldsymbol{B}-\boldsymbol{B}_0) + \boldsymbol{\Psi}_0 \right|^{-(\alpha_n+K)/2} d\boldsymbol{B} \\
&= (2\pi)^{-M(N+K)/2} |\boldsymbol{V}_0|^{-M/2} \frac{|\Psi_0|^{\alpha_0/2}}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} 2^{M(\alpha_n+K)/2} \Gamma_M((\alpha_n+K)/2) \\
&\times \int_{\mathcal{B}} \left| (\boldsymbol{B}-\widehat{\boldsymbol{B}}_n)^\top \boldsymbol{V}_n^{-1}(\boldsymbol{B}-\widehat{\boldsymbol{B}}_n) + \boldsymbol{\Psi}_n \right|^{-(\alpha_n+K)/2} d\boldsymbol{B} \\
&= (2\pi)^{-M(N+K)/2} |\boldsymbol{V}_0|^{-M/2} \frac{|\Psi_0|^{\alpha_0/2}}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} 2^{M(\alpha_n+K)/2} \Gamma_M((\alpha_n+K)/2) \\
&\times \int_{\mathcal{B}} \left[|\boldsymbol{\Psi}_n| \times |\boldsymbol{I}_K + \boldsymbol{V}_n^{-1}(\boldsymbol{B}-\widehat{\boldsymbol{B}}_n) \boldsymbol{\Psi}_n^{-1} (\boldsymbol{B}-\widehat{\boldsymbol{B}}_n)^\top| \right]^{-(\alpha_n+K)/2} d\boldsymbol{B} \\
&= |\boldsymbol{\Psi}_n|^{-(\alpha_n+K)/2} (2\pi)^{-M(N+K)/2} |\boldsymbol{V}_0|^{-M/2} \frac{|\Psi_0|^{\alpha_0/2} 2^{M(\alpha_n+K)/2} \Gamma_M((\alpha_n+K)/2)}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} \\
&\times \int_{\mathcal{B}} \left| \boldsymbol{I}_K + \boldsymbol{V}_n^{-1}(\boldsymbol{B}-\widehat{\boldsymbol{B}}_n) \boldsymbol{\Psi}_n^{-1} (\boldsymbol{B}-\widehat{\boldsymbol{B}}_n)^\top \right|^{-(\alpha_n+1-M+K+M-1)/2} d\boldsymbol{B} \\
&= |\boldsymbol{\Psi}_n|^{-(\alpha_n+K)/2} (2\pi)^{-M(N+K)/2} |\boldsymbol{V}_0|^{-M/2} \frac{|\Psi_0|^{\alpha_0/2} 2^{M(\alpha_n+K)/2} \Gamma_M((\alpha_n+K)/2)}{2^{\alpha_0 M/2} \Gamma_M(\alpha_0/2)} \\
&\times \pi^{MK/2} \frac{\Gamma_M((\alpha_n+1-M+M-1)/2)}{\Gamma_M((\alpha_n+1-M+K+M-1)/2)} |\boldsymbol{\Psi}_n|^{K/2} |\boldsymbol{V}_n|^{M/2} \\
&= \frac{|\boldsymbol{V}_n|^{M/2}}{|\boldsymbol{V}_0|^{M/2}} \frac{|\boldsymbol{\Psi}_0|^{\alpha_0/2}}{|\boldsymbol{\Psi}_n|^{\alpha_n/2}} \frac{\Gamma_M(\alpha_n/2)}{\Gamma_M(\alpha_0/2)} \pi^{-MN/2}.
\end{aligned}
$$

The third equality follows from the kernel of an inverse Wishart distribution, the fifth from Sylvester's theorem, and the seventh from the kernel of a matrix t-distribution.

Observe that this last expression is the multivariate case of the marginal likelihood of the univariate regression model. Taking into account that

$$\begin{aligned}(\boldsymbol{A}+\boldsymbol{B})^{-1} &= \boldsymbol{A}^{-1} - (\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}\boldsymbol{A}^{-1} \\ &= \boldsymbol{B}^{-1} - (\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})^{-1}\boldsymbol{B}^{-1} \\ &= \boldsymbol{A}^{-1}(\boldsymbol{A}^{-1}+\boldsymbol{B}^{-1})\boldsymbol{B}^{-1},\end{aligned}$$

we can show that $\boldsymbol{\Psi}_n = \boldsymbol{\Psi}_0 + \boldsymbol{S} + (\hat{\boldsymbol{B}} - \boldsymbol{B}_0)^\top \boldsymbol{V}_n(\hat{\boldsymbol{B}} - \boldsymbol{B}_0)$ (see Exercise 7). Therefore, the marginal likelihood rewards fit (smaller sum of squares, $\boldsymbol{S}$), similarity between prior and sample information regarding location parameters, and information gains in variability from $\boldsymbol{V}_0$ to $\boldsymbol{V}_n$.

Given a matrix of regressors $\boldsymbol{X}_0$ for N_0 unobserved units, the predictive density of $\boldsymbol{Y}_0$ given $\boldsymbol{Y}$, $\pi(\boldsymbol{Y}_0 \mid \boldsymbol{Y})$ is a matrix t distribution $T_{N_0,M}(\alpha_n - M + 1, \boldsymbol{X}_0\boldsymbol{B}_n, \boldsymbol{I}_{N_0} + \boldsymbol{X}_0\boldsymbol{V}_n\boldsymbol{X}_0^\top, \boldsymbol{\Psi}_n)$ (see Exercise 6). Observe that the prediction is centered at $\boldsymbol{X}_0\boldsymbol{B}_n$, and the covariance matrix of $vec(\boldsymbol{Y}_0)$ is $\frac{(\boldsymbol{I}_{N_0}+\boldsymbol{X}_0\boldsymbol{V}_n\boldsymbol{X}_0^\top)\otimes\boldsymbol{\Psi}_n}{\alpha_n - M - 1}$.

3.5 Summary

We introduce conjugate family models for both discrete and continuous data. These models form the foundation of the Bayesian framework due to their mathematical tractability, as they provide closed-form expressions for the posterior distributions, marginal likelihood, and predictive distribution. Additionally, we present the Bayesian linear regression frameworks for both univariate and multivariate cases under conjugate families. These frameworks are fundamental for performing regression analysis in the Bayesian setting.

3.6 Exercises

1. Write the distribution of the Bernoulli example in canonical form, and find the mean and variance of the sufficient statistic.
2. Given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \ldots\ Y_N]^\top$ from N *binomial experiments*, each with known size n_i and the same unknown probability θ, show that $p(\boldsymbol{y} \mid \theta)$ is in the exponential family. Then, find the posterior distribution, the marginal likelihood, and the predictive distribution of the binomial-Beta model assuming the number of trials is known.
3. Given a random sample $\boldsymbol{Y} = [Y_1\ Y_2\ \ldots\ Y_N]^\top$ from an *exponential*

distribution, show that $p(\boldsymbol{y} \mid \lambda)$ is in the exponential family. Additionally, find the posterior distribution, the marginal likelihood, and the predictive distribution of the exponential-Gamma model.

4. Given that $\boldsymbol{Y} \sim N_N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, that is, a *multivariate normal distribution*, show that $p(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is in the exponential family.

5. Find the marginal likelihood in the normal/inverse-Wishart model.

6. Find the posterior predictive distribution in the normal/inverse-Wishart model, and show that $\boldsymbol{Y}_0 \mid \boldsymbol{Y} \sim T_{N_0,M}(\alpha_n - M + 1, \boldsymbol{X}_0\boldsymbol{B}_n, \boldsymbol{I}_{N_0} + \boldsymbol{X}_0\boldsymbol{V}_n\boldsymbol{X}_0^\top, \boldsymbol{\Psi}_n)$.

7. Show that $\delta_n = \delta_0 + (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) + (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)^\top((\boldsymbol{X}^\top\boldsymbol{X})^{-1} + \boldsymbol{B}_0)^{-1}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)$ in the linear regression model, and that $\boldsymbol{\Psi}_n = \boldsymbol{\Psi}_0 + \boldsymbol{S} + (\hat{\boldsymbol{B}} - \boldsymbol{B}_0)^\top\boldsymbol{V}_n(\hat{\boldsymbol{B}} - \boldsymbol{B}_0)$ in the linear multivariate regression model.

8. Show that in the linear regression model $\boldsymbol{\beta}_n^\top(\boldsymbol{B}_n^{-1} - \boldsymbol{B}_n^{-1}\boldsymbol{M}^{-1}\boldsymbol{B}_n^{-1})\boldsymbol{\beta}_n = \boldsymbol{\beta}_{**}^\top\boldsymbol{C}\boldsymbol{\beta}_{**}$ and $\boldsymbol{\beta}_{**} = \boldsymbol{X}_0\boldsymbol{\beta}_n$.

9. Show that $(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B})^\top(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B}) = \boldsymbol{S} + (\boldsymbol{B} - \widehat{\boldsymbol{B}})^\top\boldsymbol{X}^\top\boldsymbol{X}(\boldsymbol{B} - \widehat{\boldsymbol{B}})$ where $\boldsymbol{S} = (\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{B}})^\top(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{B}})$, $\widehat{\boldsymbol{B}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{Y}$ in the multivariate regression model.

10. **What is the probability that the Sun will rise tomorrow?**

    This is the most famous example by Richard Price, developed in the Appendix of Bayes' theorem paper [30]. Here, we implicitly use *Laplace's Rule of Succession* to solve this problem. In particular, if we were a priori uncertain about the probability that the Sun will rise on a specified day, we can assume a uniform prior distribution over $(0, 1)$, that is, a Beta(1,1) distribution. Then, what is the probability that the Sun will rise?

11. Using information from Public Policy Polling in September 27th-28th for the 2016 presidential five-way race in USA, there are 411, 373 and 149 sampled people supporting Hillary Clinton, Donald Trump and other, respectively.

    - Find the posterior probability of the percentage difference of people supporting Hillary versus Trump according to this data using a non-informative prior, that is, $\alpha_0 = [1\ 1\ 1]$ in the multinomial-Dirichlet model. What is the probability of having more supports of Hillary vs Trump?
    - What is the probability that sampling one hundred independent individuals 44, 40 and 16 support Hillary, Trump and other, respectively?

12. **Math test example continues**

You have a random sample of math scores of size $N = 50$ from a normal distribution, $Y_i \sim N(\mu, \sigma^2)$. The sample mean and variance are equal to 102 and 10, respectively. Using the normal-normal/inverse-gamma model where $\mu_0 = 100$, $\beta_0 = 1$, $\alpha_0 = \delta_0 = 0.001$

- Get a 95% confidence and credible interval for μ.
- What is the posterior probability that $\mu > 103$?

13. **Demand of electricity example continues**

Set c_0 such that maximizes the marginal likelihood in the specifications with and without electricity price in the example of demand of electricity (empirical Bayes). Then, calculate the Bayes factor, and conclude if there is evidence supporting the inclusion of the price of electricity in the demand equation.

14. **Utility demand**

Use the file *Utilities.csv* to estimate a multivariate linear regression model where $\boldsymbol{Y}_i = [\log(\text{electricity}_i)\ \log(\text{water}_i)\ \log(\text{gas}_i)]$ as function of $\log(\text{electricity price}_i)$, $\log(\text{water price}_i)$, $\log(\text{gas price}_i)$, IndSocio1_i, IndSocio2_i, Altitude_i, Nrooms_i, HouseholdMem_i, Children_i, and $\log(\text{Income}_i)$, where electricity, water and gas are monthly consumption of electricity (kWh), water (m^3) and gas (m^3), and other definitions are given in the Example of Section 3.3. Omit households that do not consume any of the utilities in this exercise.

Set a non-informative prior framework, $\boldsymbol{B}_0 = [0]_{11\times 3}$, $\boldsymbol{V}_0 = 1000\boldsymbol{I}_{11}$, $\boldsymbol{\Psi}_0 = 1000\boldsymbol{I}_3$ and $\alpha_0 = 3$, where we have $K = 11$ (regressors plus intercept) and $M = 3$ (equations) in this exercise.

- Find the posterior mean estimates and the highest posterior density intervals at 95% of $\boldsymbol{B}$ and $\boldsymbol{\Sigma}$. Use the marginal distribution and the conditional distribution to obtain the posterior estimates of $\boldsymbol{B}$, and compare the results.
- Find the Bayes factor comparing the baseline model in this exercise with the same specification but using the income in dollars. Now, calculate the Bayes factor using the income in thousand dollars. Is there any difference?
- Find the predictive distribution for the monthly demand of electricity, water and gas in the baseline specification of a household located in the lowest socioeconomic condition in a municipality located below 1000 meters above the sea level, 2 rooms, 3 members with children, a monthly income equal to USD 500, an electricity price equal to USD/kWh 0.15, a water price equal to USD/M^3 0.70, and a gas price equal to USD/M^3 0.75.

15. **Ph.D. students sleeping hours [5, Chap. 2]**

    We are interested in learning about the proportion of Ph.D. students who sleep at least 6 hours per day. We have a sample of 52 students, where 15 report sleeping at least 6 hours, and the remaining 37 report not sleeping at least 6 hours. The prior distribution is a Beta distribution, with hyperparameters calibrated so that the prior probabilities of the proportion of students who sleep least than 6 hours being less than 0.4 and 0.75 are 0.6 and 0.95, respectively. Estimate the 95% posterior credible interval for the proportion of Ph.D. students who sleep at least 6 hours per day. Then, assume there is a group of experts whose beliefs about the proportion of Ph.D. students sleeping at least 6 hours are represented in the following table:

TABLE 3.1
Probability distribution: Ph.D students that sleep at least 6 hours per day.

h	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55
$P(p = h)$	0.05	0.07	0.10	0.12	0.15	0.17	0.15	0.11	0.06	0.01	0.01

Use Table 3.1 as prior information, and find the posterior distribution of the proportion of students that sleep at least 6 hours.

4

Simulation methods

In the previous chapters, we focused on conjugate families, where the posterior and predictive distributions have standard analytical forms (e.g., normal, Student's t, gamma, binomial, Poisson, etc.) and where the marginal likelihood has a closed-form analytical solution. However, realistic models are often more complex and lack such closed-form solutions.

To address this complexity, we rely on simulation (stochastic) methods to draw samples from posterior and predictive distributions. This chapter introduces posterior simulation, a cornerstone of Bayesian inference. We discuss Markov Chain Monte Carlo (MCMC) methods, including Gibbs sampling, Metropolis-Hastings, and Hamiltonian Monte Carlo, as well as other techniques like importance sampling and particle filtering (sequential Monte Carlo).

The simulation methods discussed in this chapter are specifically applied throughout this book. However, we do not delve into deterministic methods, such as numerical integration (quadrature), or other simulation methods, including discrete approximation, the probability integral transform, the method of composition, accept-reject sampling, and slice sampling algorithms. While these methods are also widely used, they are not as common as the approaches explicitly employed in this book.

For readers interested in these alternative methods, we recommend exploring [342, Chaps. 2 and 3], [343, Chaps. 2, 3, and 8], [173, Chap. 5], and [150, Chap. 10].

4.1 Markov chain Monte Carlo methods

Markov Chain Monte Carlo (MCMC) methods are algorithms used to approximate complex probability distributions by constructing a Markov chain. This chain is a sequence of random samples where each sample depends only on the previous one. The goal of MCMC methods is to obtain draws from the posterior distribution as the equilibrium distribution. The key point in MCMC methods is the transition kernel or density, $q(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{\theta}^{(s-1)})$, which generates a draw $\boldsymbol{\theta}^{(s)}$ at stage s that depends solely on $\boldsymbol{\theta}^{(s-1)}$. This transition distribution must be designed such that the Markov chain converges to a unique

stationary distribution, which, in our case, is the posterior distribution, that is, $\pi(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{y}) = \int_{\boldsymbol{\Theta}} q(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})d\boldsymbol{\theta}^{(s-1)}$.

Given that we start at an arbitrary point, $\boldsymbol{\theta}^{(0)}$, the algorithm requires that the Markov chain be *irreducible*, meaning that the process can reach any other state with positive probability. Additionally, the process must be *aperiodic*, meaning that for each state, the greatest common divisor of the number of steps it takes to return to the state is 1, ensuring that there are no cycles forcing the system to return to a state only after a fixed number of steps. Furthermore, the process must be *recurrent*, meaning that it will return to any state an infinite number of times with probability one. However, to ensure convergence to the stationary distribution, a stronger condition is required: the process must be *positive recurrent*, meaning that the expected return time to a state is finite. Given an *irreducible*, *aperiodic*, and *positive recurrent* transition density, the Markov chain algorithm will asymptotically converge to the stationary posterior distribution we are seeking. For more details, see [343, chap. 6].

4.1.1 Gibbs sampler

The Gibbs sampler algorithm is one of the most widely used MCMC methods for sampling from non-standard distributions in Bayesian analysis. While it is a special case of the Metropolis-Hastings (MH) algorithm, it originated from a different theoretical background [155, 146]. The key requirement for implementing the Gibbs sampling algorithm is the availability of conditional posterior distributions. The algorithm works by cycling through the conditional posterior distributions corresponding to different blocks of the parameter space under inference.

Two simplify concepts let's focus on a parameter space composed by two blocks, $\boldsymbol{\theta} = [\boldsymbol{\theta}_1 \ \boldsymbol{\theta}_2]^\top$, the Gibbs sampling algorithm uses as transition kernel $q(\boldsymbol{\theta}_1^{(s)}, \boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s-1)}, \boldsymbol{\theta}_2^{(s-1)}) = \pi(\boldsymbol{\theta}_1^{(s)} \mid \boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})$. Thus,

$$
\begin{aligned}
\int_{\boldsymbol{\Theta}} q(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})d\boldsymbol{\theta}^{(s-1)} &= \int_{\boldsymbol{\Theta}_2}\int_{\boldsymbol{\Theta}_1} \pi(\boldsymbol{\theta}_1^{(s)} \mid \boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_1^{(s-1)}, \boldsymbol{\theta}_2^{(s-1)} \mid \boldsymbol{y})d\boldsymbol{\theta}_1^{(s-1)}d\boldsymbol{\theta}_2^{(s-1)} \\
&= \pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})\int_{\boldsymbol{\Theta}_2}\int_{\boldsymbol{\Theta}_1} \pi(\boldsymbol{\theta}_1^{(s)} \mid \boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_1^{(s-1)}, \boldsymbol{\theta}_2^{(s-1)} \mid \boldsymbol{y})d\boldsymbol{\theta}_1^{(s-1)}d\boldsymbol{\theta}_2^{(s-1)} \\
&= \pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})\int_{\boldsymbol{\Theta}_2} \pi(\boldsymbol{\theta}_1^{(s)} \mid \boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_2^{(s-1)} \mid \boldsymbol{y})d\boldsymbol{\theta}_2^{(s-1)} \\
&= \pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})\int_{\boldsymbol{\Theta}_2} \pi(\boldsymbol{\theta}_1^{(s)}, \boldsymbol{\theta}_2^{(s-1)} \mid \boldsymbol{y})d\boldsymbol{\theta}_2^{(s-1)} \\
&= \pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_1^{(s)} \mid \boldsymbol{y}) \\
&= \pi(\boldsymbol{\theta}_1^{(s)}, \boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{y}).
\end{aligned}
$$

Then, $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ is the stationary distribution for the Gibbs transition kernel.

A word of caution! Even if we have well-defined conditional posterior distributions $\pi(\boldsymbol{\theta}_1^{(s)} \mid \boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})$ and $\pi(\boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{\theta}_1^{(s)}, \boldsymbol{y})$, and we can simulate from

them, the joint posterior distribution $\pi(\boldsymbol{\theta}_1^{(s)}, \boldsymbol{\theta}_2^{(s)} \mid \boldsymbol{y})$ may not correspond to any proper distribution. We should be mindful of this situation, especially when dealing with improper prior distributions (see [343, Chap. 10] for details).

Algorithm A1 demonstrates the implementation of a Gibbs sampler with d blocks. The number of iterations (S) is chosen to ensure convergence to the stationary distribution. In Section 4.4, we review several convergence diagnostics to assess whether the posterior draws have reached convergence.

Algorithm A1 Gibbs sampling

```
Set θ_2^(0), θ_3^(0), ..., θ_d^(0)
for s = 1, ..., S do
    Draw θ_1^(s) from π(θ_1^(s) | θ_2^(s-1), ..., θ_d^(s-1), y)
    Draw θ_2^(s) from π(θ_2^(s) | θ_1^(s), ..., θ_d^(s-1), y)
    ⋮
    Draw θ_d^(s) from π(θ_d^(s) | θ_1^(s), ..., θ_{d-1}^(s), y)
end for
```

Example: Mining disaster change point

Let's use the dataset *Mining.csv* provided by [64]. This dataset records the number of mining disasters per year from 1851 to 1962 in British coal mines.

We assume there is an unknown structural change point in the number of mining disasters, where the parameters of the Poisson distributions change. In particular,

$$p(y_t) = \left\{ \begin{array}{ll} \frac{\exp(-\lambda_1)\lambda_1^{y_t}}{y_t!}, & t = 1, 2, \ldots, H \\ \frac{\exp(-\lambda_2)\lambda_2^{y_t}}{y_t!}, & t = H+1, \ldots, T \end{array} \right\},$$

where H is the changing point.

We use conjugate families for λ_l, $l = 1, 2$, where $\lambda_l \sim G(\alpha_{l0}, \beta_{l0})$, and set $\pi(H) = 1/T$, which corresponds to a discrete uniform distribution for the change point. This implies that, a priori, we assume equal probability for any time to be the change point.

The posterior distribution is

$$\begin{aligned} \pi(\lambda_1, \lambda_2, H \mid \boldsymbol{y}) \propto & \prod_{t=1}^{H} \frac{\exp(-\lambda_1)\lambda_1^{y_t}}{y_t!} \prod_{t=H+1}^{T} \frac{\exp(-\lambda_2)\lambda_2^{y_t}}{y_t!} \\ & \times \exp(-\beta_{10}\lambda_1)\lambda_1^{\alpha_{10}-1} \exp(-\beta_{20}\lambda_2)\lambda_2^{\alpha_{20}-1} 1/T \\ & \propto \exp(-H\lambda_1)\lambda_1^{\sum_{t=1}^{H} y_t} \exp(-(T-H)\lambda_2)\lambda_2^{\sum_{t=H+1}^{T} y_t} \\ & \times \exp(-\beta_{10}\lambda_1)\lambda_1^{\alpha_{10}-1} \exp(-\beta_{20}\lambda_2)\lambda_2^{\alpha_{20}-1}. \end{aligned}$$

Then, the conditional posterior distribution of $\lambda_1 \mid \lambda_2, H, \boldsymbol{y}$ is

$$\pi(\lambda_1 \mid \lambda_2, H, \boldsymbol{y}) \propto \exp(-(H+\beta_{10})\lambda_1)\lambda_1^{\sum_{t=1}^{H} y_t + \alpha_{10} - 1},$$

that is, $\lambda_1 \mid \lambda_2, H, \boldsymbol{y} \sim G(\alpha_{1n}, \beta_{1n})$, $\beta_{1n} = H + \beta_{10}$ and $\alpha_{1n} = \sum_{t=1}^{H} y_t + \alpha_{10}$.

The conditional posterior distribution of $\lambda_2 \mid \lambda_1, H, \boldsymbol{y}$ is

$$\pi(\lambda_2 \mid \lambda_1, H, \boldsymbol{y}) \propto \exp(-((T-H)+\beta_{20})\lambda_2)\lambda_2^{\sum_{t=H+1}^{T} y_t + \alpha_{20} - 1},$$

that is, $\lambda_2 \mid \lambda_1, H, \boldsymbol{y} \sim G(\alpha_{2n}, \beta_{2n})$, $\beta_{2n} = (T-H) + \beta_{20}$ and $\alpha_{2n} = \sum_{t=H+1}^{T} y_t + \alpha_{20}$.

The conditional posterior distribution of the change point is

$$\begin{aligned}
\pi(H \mid \lambda_1, \lambda_2, \boldsymbol{y}) &\propto \exp(-H\lambda_1)\lambda_1^{\sum_{t=1}^{H} y_t} \exp(-(T-H)\lambda_2)\lambda_2^{\sum_{t=H+1}^{T} y_t} \\
&\propto \exp(-H(\lambda_1-\lambda_2))\lambda_1^{\sum_{t=1}^{H} y_t} \lambda_2^{\sum_{t=H+1}^{T} y_t} \exp(-T\lambda_2) \frac{\lambda_2^{\sum_{t=1}^{H} y_t}}{\lambda_2^{\sum_{t=1}^{H} y_t}} \\
&\propto \exp(-H(\lambda_1-\lambda_2)) \left(\frac{\lambda_1}{\lambda_2}\right)^{\sum_{t=1}^{H} y_t}.
\end{aligned}$$

Thus, the conditional posterior distribution of H is

$$\pi(H \mid \lambda_1, \lambda_2, \boldsymbol{y}) = \frac{\exp(-H(\lambda_1-\lambda_2)) \left(\frac{\lambda_1}{\lambda_2}\right)^{\sum_{t=1}^{H} y_t}}{\sum_{H=1}^{T} \exp(-H(\lambda_1-\lambda_2)) \left(\frac{\lambda_1}{\lambda_2}\right)^{\sum_{t=1}^{H} y_t}}, \quad H = 1, 2, \ldots, T.$$

The following code shows how to do a Gibbs sampling algorithm to perform inference of this model using the hyperparameters suggested by [173, Chap. 7], $\alpha_{l0} = 0.5$ and $\beta_{l0} = 1$, $l = 1, 2$.

R code. Gibbs sampler: The mining disaster changepoint

```
rm(list = ls()); set.seed(010101)
dataset<-read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/
    MiningDataCarlin.csv",header=T)
attach(dataset); str(dataset)
a10<-0.5; a20<-0.5
b10<-1; b20<-1
y<-Count
sumy<-sum(Count); N<-length(Count)
theta1<-NULL; theta2<-NULL
kk<-NULL; k<-60; S<-10000
for(i in 1:S){
  a1<-a10+sum(y[1:k]); b1<-b10+k
  theta11<-rgamma(1,a1,b1)
  theta1<-c(theta1,theta11)
  a2<-a20+sum(y[(1+k):N]); b2<-b20+N-k
  theta22<-rgamma(1,a2,b2)
  theta2<-c(theta2,theta22)
  pp<-NULL
  for(l in 1:N){
    p<-exp(l*(theta22-theta11))*(theta11/theta22)^(sum(y[1:l
    ]))
    pp<-c(pp,p)
  }
  prob<-pp/sum(pp); k<-sample(1:N,1,prob=prob)
  kk<-c(kk,k)
}
library(coda); summary(mcmc(theta1)); summary(mcmc(theta2))
summary(mcmc(kk)); hist(kk, main = "Histogram: Posterior
    mean change point", xlab = "Posterior mean", col = "blue
    ", breaks = 25)
```

The posterior results indicate that the rate of disasters decrease from 3.1 to 0.92 per year in 1890.

Figure 4.1 shows the histogram of the posterior draws of the change point in mining disasters.

4.1.2 Metropolis-Hastings

The Metropolis-Hastings (M-H) algorithm [284, 184] is a general MCMC method that does not require standard closed-form solutions for the conditional posterior distributions. The key idea is to use a transition kernel whose unique invariant distribution is $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$. This kernel must satisfy the *balanc-*

FIGURE 4.1
Histogram of posterior draws of change point: Mining disasters

ing condition, meaning that, given a realization $\boldsymbol{\theta}^{(s-1)}$ at stage $s-1$ from the stationary distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$, we generate a candidate draw $\boldsymbol{\theta}^c$ from the *proposal distribution* $q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})$ at stage s such that:

$$q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y}) = q(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y}),$$

which implies that the probability of moving from $\boldsymbol{\theta}^{(s-1)}$ to $\boldsymbol{\theta}^c$ is equal to the probability of moving from $\boldsymbol{\theta}^c$ to $\boldsymbol{\theta}^{(s-1)}$.

In general, the *balancing condition* is not automatically satisfied, and we must introduce an *acceptance probability* $\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c)$ to ensure that the condition holds:

$$q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) = q(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y}).$$

Thus, the acceptance probability is given by:

$$\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) = \min\left\{\frac{q(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y})}{q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})}, 1\right\},$$

where $q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})$ and $\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})$ must be nonzero, as transitioning from $\boldsymbol{\theta}^{(s-1)}$ to $\boldsymbol{\theta}^c$ is only possible under these conditions.

Algorithm A2 shows how to implement a Metropolis-Hastings algorithm. The number of iterations (S) is chosen to ensure convergence to the stationary distribution.

Some remarks: First, we do not need to know the marginal likelihood to

Algorithm A2 Metropolis-Hastings algorithm

1: Set $\boldsymbol{\theta}^{(0)}$ in the support of $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$
2: **for** $s = 1, \ldots, S$ **do**
3: Draw $\boldsymbol{\theta}^c$ from $q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})$
4: Calculate $\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) = \min\left\{\frac{q(\boldsymbol{\theta}^{(s-1)}|\boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c|\boldsymbol{y})}{q(\boldsymbol{\theta}^c|\boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)}|\boldsymbol{y})}, 1\right\}$
5: Draw U from $U(0,1)$
6: $\boldsymbol{\theta}^{(s)} = \left\{\begin{matrix} \boldsymbol{\theta}^c & \text{if } U \leq \alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) \\ \boldsymbol{\theta}^{(s-1)} & \text{otherwise} \end{matrix}\right\}$
7: **end for**

implement the M-H algorithm, as it cancels out when calculating the acceptance probability. Specifically, given that $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{\theta}) \times p(\boldsymbol{y} \mid \boldsymbol{\theta})$, we can use the right-hand side expression to compute the acceptance probability. Second, the Gibbs sampling algorithm is a particular case of the M-H algorithm where the acceptance probability is equal to 1 ([154] and [343, Chap. 10], see Exercise 2). Third, we can combine the M-H and Gibbs sampling algorithms when dealing with relatively complex posterior distributions. Specifically, the Gibbs sampling algorithm can be used for blocks with conditional posterior distributions in standard closed forms, while the M-H algorithm is applied to sample from conditional posterior distributions that do not have standard forms. This approach is known as the M-H within Gibbs sampling algorithm. Fourth, we can note that the transition kernel in the M-H algorithm is a mixture of a continuous density ($q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)})$) and a probability mass function ($\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c)$) [86].

Fifth, a crucial point associated with the proposal densities is the acceptance probability. Low or high acceptance probabilities are not ideal. A low rate implies poor mixing, meaning the chain does not move effectively through the support of the posterior distribution. Conversely, a high acceptance rate implies that the chain will converge too slowly. A sensible value depends on the dimension of the parameter space. A rule of thumb is that if the dimension is less than or equal to 2, the acceptance rate should be around 0.50. If the dimension is greater than 2, the acceptance rate should be approximately 0.25 [344]. For technical details of the Metropolis-Hastings algorithm, see [343, Chap. 7].

Regarding the proposal density, it must be positive everywhere the posterior distribution is positive. This ensures that the Markov chain can explore the entire support of the posterior distribution. Additionally, the proposal density must allow the Markov chain to reach any region of the posterior distribution's support. There are three standard approaches for choosing the proposal density: the independent proposal, the random walk proposal, and the tailored proposal.

In the independent proposal, $q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)}) = q(\boldsymbol{\theta}^c)$, which implies that

$$\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) = \min\left\{\frac{q(\boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y})}{q(\boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})}, 1\right\}.$$

In this case, a move from $\boldsymbol{\theta}^{(s-1)}$ to $\boldsymbol{\theta}^c$ is always accepted if $q(\boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y}) \geq q(\boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})$.

In the random walk proposal, $\boldsymbol{\theta}^c = \boldsymbol{\theta}^{(s-1)} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a random perturbation. If $p(\boldsymbol{\epsilon}) = p(-\boldsymbol{\epsilon})$, meaning the distribution $p(\boldsymbol{\epsilon})$ is symmetric around zero, then $q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{(s-1)}) = q(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{\theta}^c)$. This was the original Metropolis algorithm [284]. Thus, the acceptance rate is

$$\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) = \min\left\{\frac{\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y})}{\pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})}, 1\right\}.$$

In this case, a move from $\boldsymbol{\theta}^{(s-1)}$ to $\boldsymbol{\theta}^c$ is always accepted if $\pi(\boldsymbol{\theta}^c \mid \boldsymbol{y}) \geq \pi(\boldsymbol{\theta}^{(s-1)} \mid \boldsymbol{y})$.

In the tailored proposal, the density is designed to have fat tails, is centered at the mode of the posterior distribution, and its scale matrix is given by the negative inverse Hessian matrix evaluated at the mode. Specifically, for two blocks, the log posterior distribution is maximized with respect to $\boldsymbol{\theta}_1$ given $\boldsymbol{\theta}_2$. This process is repeated at each iteration of the algorithm because $\boldsymbol{\theta}_2$ changes at different stages. As a result, the algorithm can be slow since the optimization process is computationally demanding (see [173, Chap. 7 and 9] for examples).

A sensible recommendation when performing M-H algorithm is to use a random walk proposal such that $\boldsymbol{\epsilon} \sim N(\mathbf{0}, c^2\boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is the negative inverse Hessian matrix evaluated at the mode, that is, maximize with respect to all parameters, and set $c \approx 2.4/\sqrt{dim\,\{\boldsymbol{\theta}\}}$, which is the most efficient scale compared to independent sampling [150, Chap. 12] and [152]. After some iterations of the algorithm, adjust the scale matrix $\boldsymbol{\Sigma}$ as before, and increase or decrease c if the acceptance rate of the simulations is too high or low, respectively. The objective is to bring the acceptance rate to the stated rule of thumb, that is, if the dimension is less than or equal to 2, the acceptance rate should be around 0.50, and if the dimension is greater than 2, the acceptance rate should be around 0.25. Once this is achieved, we should run the algorithm without modifications and use this part of the algorithm to perform inference.

Example: Ph.D. students sleeping hours continues

In the Ph.D. students sleeping hours exercise of Chapter 3 we get a posterior distribution that is Beta with parameters 16.55 and 39.57. We can sample from this posterior distribution using the function *rbeta* from **R**. However, we want to compare the performance of a M-H algorithm using as proposal density a $U(0, 1)$ distribution.

The following code shows how to do a M-H algorithm to sample from the beta distribution using the uniform distribution.

R code. Metropolis-Hastings algorithm: Ph.D. students sleeping hours

```
rm(list = ls()); set.seed(010101)
an <- 16.55; bn <- 39.57
S <- 100000; p <- runif(S); accept <- rep(0, S)
for (s in 2:S){
  pc <- runif(1) # Candidate
  a <- dbeta(pc, an, bn)/dbeta(p[s-1], an, bn) # Acceptance
    rate
  U <- runif(1)
  if(U <= a){
    p[s] <- pc
    accept[s] <- 1
  }else{
    p[s] <- p[s-1]
    accept[s] <- 0
  }
}
mean(accept); mean(p); sd(p)
an/(an + bn); (an*bn/((an+bn)^2*(an+bn+1)))^0.5 # Population
     values
h <- hist(p, breaks=50, col="blue", xlab="Proportion Ph.D.
    students sleeping at least 6 hours", main="Beta draws
    from a Metropolis-Hastings algorithm")
pfit <- seq(min(p),max(p),length=50)
yfit<-dbeta(pfit, an, bn)
yfit <- yfit*diff(h$mids[1:2])*length(p)
lines(pfit, yfit, col="red", lwd=2)
```

The results indicate that the mean and standard deviation obtained from the posterior draws are similar to the population values. Furthermore, Figure 4.2 presents the histogram of the posterior draws alongside the density of the beta distribution, demonstrating a good match between them.

4.1.3 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) was proposed by [120] and later introduced to the statistical community by [289]. HMC extends the Metropolis algorithm to efficiently explore the parameter space by introducing *momentum variables*, which help overcome the random walk behavior of Gibbs sampling and the Metropolis-Hastings algorithm. Known also as hybrid Monte Carlo, HMC is particularly advantageous for high-dimensional posterior distributions, as it reduces the risk of getting stuck in local modes and significantly improves mixing [290].

FIGURE 4.2
Histogram of posterior draws of beta distribution and the density of the beta distribution.

However, HMC is designed to work with strictly positive target densities. Therefore, transformations are required to handle bounded parameters, such as variances and proportions. For example, logarithmic and logit transformations can be applied. These transformations necessitate the use of the change-of-variable theorem to compute the log posterior density and its gradient, which are essential for implementing the HMC algorithm.

HMC leverages concepts from physics, specifically Hamiltonian mechanics, to propose transitions in the Markov chain. In Hamiltonian mechanics, two key variables define the total energy of the system: the *position* ($\boldsymbol{\theta}$) and the *momentum* ($\boldsymbol{\delta}$). The Hamiltonian represents the total energy of the system, consisting of *potential energy* (energy due to position) and *kinetic energy* (energy associated with motion). The objective is to identify trajectories that preserve the system's total energy, meaning the Hamiltonian remains invariant, while avoiding trajectories that do not. This approach enhances the acceptance rate of proposed transitions.

To implement HMC, we solve the differential equations derived from the Hamiltonian, which involve derivatives with respect to position and momentum. However, these equations rarely have analytical solutions, requiring numerical methods for approximation. This necessitates discretizing Hamilton's equations, which introduces errors. To mitigate these errors, HMC uses the *leapfrog integrator*, a numerical method with smaller errors compared to simpler approaches like the Euler method.

HMC uses a *momentum variable* (δ_k) for each θ_k, so that the transition

kernel of $\boldsymbol{\theta}$ is determined by $\boldsymbol{\delta}$. Both vectors are updated using a Metropolis algorithm at each stage such that the distribution of $\boldsymbol{\theta}$ remains invariant [290]. The joint density in HMC is given by $p(\boldsymbol{\theta}, \boldsymbol{\delta} \mid \boldsymbol{y}) = \pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \times p(\boldsymbol{\delta})$, where $\boldsymbol{\delta} \sim N(\mathbf{0}, \boldsymbol{M})$, and $\boldsymbol{M}$ is a diagonal matrix such that $\delta_k \sim N(0, M_{kk})$.

Algorithm A3 outlines the HMC implementation. The gradient vector $\frac{d \log(\pi(\boldsymbol{\theta}|\boldsymbol{y}))}{d\boldsymbol{\theta}}$ must be computed analytically, as using finite differences can be computationally expensive. However, it is advisable to verify the analytical calculations by evaluating the gradient at the maximum posterior estimate, where the function should return values close to 0, or by comparing results with finite differences at a few points.

Algorithm A3 Hamiltonian Monte Carlo

Initiate at $\boldsymbol{\theta}^{(0)}$ in the support of $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$, and set step size ϵ, number of leapfrog steps L, and total iterations S
Draw $\boldsymbol{\delta}^{(0)}$ from $N(\mathbf{0}, \boldsymbol{M})$
for $s = 1, \ldots, S$ **do**
  for $l = 1, \ldots, L$ **do**
    if l = 1 **then**
      $\boldsymbol{\delta}^c \leftarrow \boldsymbol{\delta}^{(s-1)} + \frac{1}{2}\epsilon \frac{d \log(\pi(\boldsymbol{\theta}|\boldsymbol{y}))}{d\boldsymbol{\theta}}$
      $\boldsymbol{\theta}^c \leftarrow \boldsymbol{\theta}^{(s-1)} + \epsilon \boldsymbol{M}^{-1} \boldsymbol{\delta}^c$
    else
      if l=2,...,L-1 **then**
        $\boldsymbol{\delta}^c \leftarrow \boldsymbol{\delta}^c + \epsilon \frac{d \log(\pi(\boldsymbol{\theta}|\boldsymbol{y}))}{d\boldsymbol{\theta}}$
        $\boldsymbol{\theta}^c \leftarrow \boldsymbol{\theta}^c + \epsilon \boldsymbol{M}^{-1} \boldsymbol{\delta}^c$
      else
        $\boldsymbol{\delta}^c \leftarrow \boldsymbol{\delta}^c + \frac{1}{2}\epsilon \frac{d \log(\pi(\boldsymbol{\theta}|\boldsymbol{y}))}{d\boldsymbol{\theta}}$
        $\boldsymbol{\theta}^c \leftarrow \boldsymbol{\theta}^c + \epsilon \boldsymbol{M}^{-1} \boldsymbol{\delta}^c$
      end if
    end if
  end for
  Calculate $\alpha([\boldsymbol{\theta}\ \boldsymbol{\delta}]^{(s-1)}, [\boldsymbol{\theta}\ \boldsymbol{\delta}]^c) = \min\left\{\frac{p(\boldsymbol{\delta}^c)\pi(\boldsymbol{\theta}^c|\boldsymbol{y})}{p(\boldsymbol{\delta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)}|\boldsymbol{y})}, 1\right\}$
  Draw U from $U(0, 1)$
  $\boldsymbol{\theta}^{(s)} = \left\{\begin{matrix} \boldsymbol{\theta}^c & \text{if } U \leq \alpha(\cdot,\cdot) \\ \boldsymbol{\theta}^{(s-1)} & \text{otherwise} \end{matrix}\right\}$
end for

Note that HMC does not require the marginal likelihood, as neither the gradient vector $\frac{d \log(\pi(\boldsymbol{\theta}|\boldsymbol{y}))}{d\boldsymbol{\theta}}$ nor the acceptance rate depend on it. That is, we can use only $\pi(\boldsymbol{\theta}) \times p(\boldsymbol{y} \mid \boldsymbol{\theta})$ to implement HMC. In addition, we do not retain $\boldsymbol{\delta}$ after it is updated at the beginning of each iteration, as it is not required subsequently. To begin, the step size (ϵ) can be drawn randomly from a uniform distribution between 0 and $2\epsilon_0$, and the number of leapfrog steps (L) is set as the largest integer near $1/\epsilon$, ensuring $\epsilon \times L \approx 1$. We need to

set $\boldsymbol{M}$ to be the inverse of the posterior covariance matrix evaluated at the maximum a posteriori estimate under this setting.

The acceptance rate should be checked, with the optimal rate around 65% [150, Chap. 12]. If the acceptance rate is much higher than 65%, increase ϵ_0; if it is much lower, decrease it. This strategy may not always work, and alternative strategies can be tested, such as setting $\boldsymbol{M} = \boldsymbol{I}$ and fine-tuning ϵ and L to achieve an acceptance rate near 65%. Finally, the number of iterations (S) is chosen to ensure convergence to the stationary distribution.

Example: Sampling from a bi-variate Gaussian distribution

As a toy example, let's compare the Gibbs sampling, M-H and HMC algorithms when the posterior distribuion is a bi-variate Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$. Let's set $\rho = 0.98$.

The Gibbs sampler requires the conditional posterior distributions, which in this case are $\theta_1 \mid \theta_2 \sim N(\rho\theta_2, 1-\rho^2)$ and $\theta_2 \mid \theta_1 \sim N(\rho\theta_1, 1-\rho^2)$. We use the random walk proposal distribution for the M-H algorithm, where $\boldsymbol{\theta}^c \sim N(\boldsymbol{\theta}^{(s-1)}, \operatorname{diag}\left\{0.18^2\right\})$. We set $\epsilon = 0.05$, $L = 20$ and $\boldsymbol{M} = \boldsymbol{I}_2$ for the HMC algorithm, and given that $\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \propto \exp\left\{-\frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}\right\}$, then $\frac{d\log(\pi(\boldsymbol{\theta}|\boldsymbol{y}))}{d\boldsymbol{\theta}} = -\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}$.

The following code shows how to implement the Gibbs sampler, the random walk M-H algorithm, and the HMC in this example such that the effective number of posterior draws is 400.

R code. Gibbs, M-H and HMC: Bi-variate normal distribution

```
rm(list = ls()); set.seed(010101)
# Gibbs sampler
Gibbs <- function(theta, rho){
  thetal <- rnorm(1, mean = rho*theta, sd = (1- rho^2)^0.5)
  return(thetal)
}
# Metropolis-Hastings
MH <- function(theta, rho, sig2){
  SIGMA <- matrix(c(1, rho, rho, 1), 2, 2)
  SIGMAc <- matrix(c(1, sig2, sig2, 1), 2, 2)
  thetac <- MASS::mvrnorm(1, mu = theta, Sigma = SIGMAc)
  a <- mvtnorm::dmvnorm(thetac, c(0, 0), SIGMA)/mvtnorm::
    dmvnorm(theta, c(0, 0), SIGMA)
  U <- runif(1)
  if(U <= a){
    theta <- thetac
    accept <- 1
  }else{
    theta <- theta
    accept <- 0
  }
  return(list(theta = theta, accept = accept))
}
# Hamiltonian Monte Carlo
HMC <- function(theta, rho, epsilon, M){
  SIGMA <- matrix(c(1, rho, rho, 1), 2, 2)
  L <- ceiling(1/epsilon)
  Minv <- solve(M); thetat <- theta
  K <- length(thetat)
  mom <- t(mvtnorm::rmvnorm(1, rep(0, K), M))
  logPost_Mom_t <- mvtnorm::dmvnorm(t(theta), rep(0, K),
    SIGMA, log = TRUE) +  mvtnorm::dmvnorm(t(mom), rep(0, K)
    , M, log = TRUE)
  for(l in 1:L){
    if(l == 1 | l == L){
      mom <- mom + 0.5*epsilon*(-solve(SIGMA)%*%theta)
      theta <- theta + epsilon*Minv%*%mom
    }else{
      mom <- mom + epsilon*(-solve(SIGMA)%*%theta)
      theta <- theta + epsilon*Minv%*%mom
    }
  }
  logPost_Mom_star <- mvtnorm::dmvnorm(t(theta), rep(0, K),
    SIGMA, log = TRUE) +  mvtnorm::dmvnorm(t(mom), rep(0, K)
    , M, log = TRUE)
  alpha <- min(1, exp(logPost_Mom_star-logPost_Mom_t))
  u <- runif(1)
  if(u <= alpha){
    thetaNew <- c(theta)
  }else{
    thetaNew <- thetat
  }
  rest <- list(theta = thetaNew, Prob = alpha)
  return(rest)
}
```

R code. Gibbs, M-H and HMC: Bi-variate normal distribution

```
# Hyperparameters
rho <- 0.98; sig2 <- 0.18^2
# Posterior draws Gibbs and M-H
S <- 8000; thin <- 20; K <- 2
thetaPostGibbs <- matrix(NA, S, K)
thetaPostMH <- matrix(NA, S, K)
AcceptMH <- rep(NA, S)
thetaGibbs <- c(-2, 3); thetaMH <- c(-2, 3)
for(s in 1:S){
  theta1 <- Gibbs(thetaGibbs[2], rho)
  theta2 <- Gibbs(theta1, rho)
  thetaGibbs <- c(theta1, theta2)
  ResMH <- MH(thetaMH, rho, sig2)
  thetaMH <- ResMH$theta
  thetaPostGibbs[s,] <- thetaGibbs
  thetaPostMH[s,] <- thetaMH
  AcceptMH[s] <- ResMH$accept
}
keep <- seq(0, S, thin)
mean(AcceptMH[keep[-1]])
thetaPostGibbsMCMC <- coda::mcmc(thetaPostGibbs[keep[-1],])
summary(thetaPostGibbsMCMC)
coda::autocorr.plot(thetaPostGibbsMCMC)
thetaPostMHMCMC <- coda::mcmc(thetaPostMH[keep[-1],])
plot(thetaPostMHMCMC)
coda::autocorr.plot(thetaPostMHMCMC)
# Posterior draws HMC
S <- 400;epsilon <- 0.05;  L <- ceiling(1/epsilon); M <-
    diag(2)
thetaPostHMC <- matrix(NA, S, K)
ProbAcceptHMC  <- rep(NA, S)
thetaHMC <- c(-2, 3)
for(s in 1:S){
  ResHMC <- HMC(theta = thetaHMC, rho, epsilon, M)
  thetaHMC <- ResHMC$theta
  thetaPostHMC[s,] <- thetaHMC
  ProbAcceptHMC[s] <- ResHMC$Prob
}
thetaPostHMCMCMC <- coda::mcmc(thetaPostHMC)
plot(thetaPostHMCMCMC); coda::autocorr.plot(thetaPostHMCMCMC
    )
summary(ProbAcceptHMC)
#Figure
df <- as.data.frame(cbind(1:S, thetaPostHMC[,1], thetaPostMH
    [keep[-1],1], thetaPostGibbs[keep[-1],1]))
colnames(df) <- c("Iter", "HMC", "MH", "Gibbs")
library(latex2exp); library(ggpubr)
g1 <- ggplot(df, aes(x= Iter)) + geom_point(aes(y=HMC),
    colour="black") + labs(x = "Iteration", y = TeX("$\\
    theta_{1}$"), title = "HMC algorithm")
g2 <- ggplot(df, aes(x= Iter)) + geom_point(aes(y=MH),
    colour="black") + labs(x = "Iteration", y = TeX("$\\
    theta_{1}$"), title = "M-H algorithm")
g3 <- ggplot(df, aes(x= Iter)) + geom_point(aes(y=Gibbs),
    colour="black") + labs(x = "Iteration", y = TeX("$\\
    theta_{1}$"), title = "Gibbs sampling")
ggarrange(g3, g2, g1, labels = c("A", "B", "C"), ncol = 3,
    nrow = 1)
```

Figure 4.3 shows the posterior draws of θ_1 using the Gibbs sampler (Panel A, left), the Metropolis-Hastings algorithm (Panel B, middle), and the Hamiltonian Monte Carlo (Panel C, right). The convergence diagnostic plots (no shown) suggests that the three algorithms perform a good job. Although, the acceptance rate in HMC is higher than the M-H due to the HMC producing larger changes in $\boldsymbol{\theta}$ than a corresponding number of random-walk M-H iterations [290].

FIGURE 4.3
Posterior draws: Gibbs sampler (A), Metropolis-Hastings (B) and Hamiltonian Monte Carlo (C) in the bivariate normal example, $\rho = 0.98$.

4.2 Importance sampling

Up to this section, we have introduced MCMC methods for sampling from the posterior distribution when it does not have a standard closed form. However, MCMC methods have some limitations. First, the samples are generated sequentially, which complicates parallel computing. Although multiple MCMC chains can be run simultaneously, this approach—often referred to as brute-force parallelization—does not fully address the sequential nature of individual chains. Second, consecutive samples are correlated, which reduces the effective sample size and complicates convergence diagnostics.

Thus, in this section, we introduce *importance sampling* (IS), a simulation method for drawing samples from the posterior distribution that avoids

these limitations. Unlike MCMC, IS does not require satisfying the balancing condition, making it conceptually and mathematically simpler to implement in certain situations. Moreover, importance weights can be reused to analyze posterior quantities, compute marginal likelihoods, compare models, approximate new target distributions, and allow for straightforward parallelization in large-scale problems.

However, the critical challenge in IS lies in selecting an appropriate proposal distribution. This involves satisfying both support and stability conditions, which can be difficult to achieve, particularly in high-dimensional problems. In such cases, MCMC methods may be more suitable.

The starting point is evaluating the integral:

$$\mathbb{E}_\pi[h(\boldsymbol{\theta})] = \int_{\boldsymbol{\Theta}} h(\boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y})d\boldsymbol{\theta}, \tag{4.1}$$

where $\mathbb{E}_\pi$ denotes expected value under the posterior distribution. Thus, we can approximate Equation 4.1 by

$$\bar{h}(\boldsymbol{\theta})_S = \frac{1}{S}\sum_{s=1}^{S} h(\boldsymbol{\theta}^{(s)}), \tag{4.2}$$

where $\boldsymbol{\theta}^{(s)}$ are draws from $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$. The *strong law of large numbers* shows that $\bar{h}(\boldsymbol{\theta})_S$ converges (almost surely) to $\mathbb{E}_\pi[h(\boldsymbol{\theta})]$ as $S \to \infty$.

The challenge arises when we do not know how to obtain samples from $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$. The ingenious idea is to express Equation 4.1 in a different way using the *importance sampling fundamental identity* [343, Chap. 3]:

$$\begin{aligned}\mathbb{E}_\pi[h(\boldsymbol{\theta})] &= \int_{\boldsymbol{\Theta}} h(\boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y})\frac{q(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}d\boldsymbol{\theta} \\ &= \mathbb{E}_q\left[\frac{h(\boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y})}{q(\boldsymbol{\theta})}\right],\end{aligned} \tag{4.3}$$

where $q(\boldsymbol{\theta})$ is the proposal distribution.

Thus, we have

$$\frac{1}{S}\sum_{s=1}^{S}\left[\frac{h(\boldsymbol{\theta}^{(s)})\pi(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{y})}{q(\boldsymbol{\theta}^{(s)})}\right] = \frac{1}{S}\sum_{s=1}^{S} h(\boldsymbol{\theta}^{(s)})w(\boldsymbol{\theta}^{(s)}),$$

where $w(\boldsymbol{\theta}^{(s)}) = \left[\frac{\pi(\boldsymbol{\theta}^{(s)}|\boldsymbol{y})}{q(\boldsymbol{\theta}^{(s)})}\right]$ are called the *importance weights*, and $\boldsymbol{\theta}^{(s)}$ are samples from the proposal distribution. This expression converges to $\mathbb{E}_\pi[h(\boldsymbol{\theta})]$ given that the support of $q(\boldsymbol{\theta})$ includes the support of $\pi(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{y})$.

There are many proposal distributions that satisfy the support condition. However, the stability of the method depends heavily on the variability of the importance weights. In particular, the variance of

$$\frac{1}{S}\sum_{s=1}^{S} h(\boldsymbol{\theta}^{(s)})w(\boldsymbol{\theta}^{(s)})$$

can be large if the proposal distribution has lighter tails than the posterior distribution. In this case, the weights $w(\boldsymbol{\theta}^{(s)})$ will vary widely, assigning too much importance to a few values of $\boldsymbol{\theta}^{(s)}$. Thus, it is important to use proposals that have thicker tails than the posterior distribution. In any case, we should check the adequacy of the proposal distribution by analyzing the behavior of the importance weights. If they are distributed more or less uniformly over the support, it is a good sign. Consider, for instance, the extreme case where $q(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta} \mid \boldsymbol{y})$, then $w(\boldsymbol{\theta}^{(s)}) = 1$ everywhere.

A natural choice in Bayesian inference is to use the prior distribution as the proposal, given that it is a proper density function. The prior distribution typically has heavier tails than the posterior by construction, and it is usually a distribution that allows for easy sampling.

The most relevant point for us is that importance sampling provides a way to simulate from the posterior distribution when there is no closed-form solution. The method generates samples $\boldsymbol{\theta}^{(s)}$ from $q(\boldsymbol{\theta})$ and computes the importance weights $w(\boldsymbol{\theta}^{(s)})$. Thus, if we *resample* with replacement from $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(S)}$, selecting $\boldsymbol{\theta}^{(s)}$ with probability proportional to $w(\boldsymbol{\theta}^{(s)})$, we would get a sample $\boldsymbol{\theta}^{*(1)}, \boldsymbol{\theta}^{*(2)}, \ldots, \boldsymbol{\theta}^{*(L)}$ of size L from $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ [382, 357]. This is named *sampling/importance resampling* (SIR) algorithm. Observe that the number of times $L^{(s)}$ each particular point $\boldsymbol{\theta}^{(s)}$ is selected follows a binomial distribution with size L, and probabilities proportional to $w^{(s)}$. Consequently, the vector $L_{\boldsymbol{\theta}} = \{L_{\boldsymbol{\theta}^1}, L_{\boldsymbol{\theta}^2}, \ldots, L_{\boldsymbol{\theta}^S}\}$ follows a multinomial distribution with L trials and probabilities proportional to $w(\boldsymbol{\theta}^{(s)})$, $s = 1, 2, \ldots, S$ [62]. Therefore, the resampling step ensures that points in the first-stage sample with small importance weights are more likely to be discarded, while points with high weights are replicated in proportion to their importance weights. In most applications, it is typical to have $S \gg L$.

The intuition is that importance weights are scaling factors that correct for the bias introduced by drawing from $q(\boldsymbol{\theta}^{(s)})$ instead of $\pi(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{y})$; thus, when combined, the samples and weights effectively recreate the posterior distribution, ensuring the resampled data set reflects the posterior. Let's proof this

$$
\begin{aligned}
P(\boldsymbol{\theta}^* \in A) &= \frac{1}{S} \sum_{s=1}^{S} w^{(s)} \mathbb{1}_A(\boldsymbol{\theta}^{(s)}) \\
&\to \mathbb{E}_q \left[\mathbb{1}_{\in A}(\boldsymbol{\theta}) \frac{\pi(\boldsymbol{\theta} \mid \boldsymbol{y})}{q(\boldsymbol{\theta})} \right] \\
&= \int_A \left[\frac{\pi(\boldsymbol{\theta} \mid \boldsymbol{y})}{q(\boldsymbol{\theta})} \right] q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&= \int_A \pi(\boldsymbol{\theta} \mid \boldsymbol{y}) d\boldsymbol{\theta}.
\end{aligned}
$$

Thus, $\boldsymbol{\theta}^*$ is approximately distributed as an observation from $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$.

However, the weights $\pi(\boldsymbol{\theta}^{(s)} \mid \boldsymbol{y})/(Sq(\boldsymbol{\theta}^{(s)}))$ do not sum up to 1, and we

need to standardize them,

$$w^*(\boldsymbol{\theta}^{(s)}) = \frac{\frac{1}{S} w(\boldsymbol{\theta}^{(s)})}{\frac{1}{S}\sum_{s=1}^{S} w(\boldsymbol{\theta}^{(s)})}.$$

Note that we could alternatively arrive to these weights as follow

$$\begin{aligned}\mathbb{E}_\pi[h(\boldsymbol{\theta})] &= \int_{\boldsymbol{\Theta}} \left[\frac{h(\boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y})}{q(\boldsymbol{\theta})}\right] q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \frac{\int_{\boldsymbol{\Theta}} \left[\frac{h(\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\boldsymbol{y})}{q(\boldsymbol{\theta})}\right] q(\boldsymbol{\theta}) d\boldsymbol{\theta}}{\int_{\boldsymbol{\Theta}} \left[\frac{\pi(\boldsymbol{\theta}|\boldsymbol{y})}{q(\boldsymbol{\theta})}\right] q(\boldsymbol{\theta}) d\boldsymbol{\theta}}.\end{aligned}$$

Then,

$$\frac{\frac{1}{S}\sum_{s=1}^{S} h(\boldsymbol{\theta}^{(s)}) w(\boldsymbol{\theta}^{(s)})}{\frac{1}{S}\sum_{s=1}^{S} w(\boldsymbol{\theta})^{(s)}} = \sum_{s=1}^{S} h(\boldsymbol{\theta}^{(s)}) w^*(\boldsymbol{\theta}^{(s)}).$$

This alternative expression also converges (almost surely) to $\mathbb{E}_\pi[h(\boldsymbol{\theta})]$. In addition, this expression is very useful because if we do not have the marginal likelihood in the posterior distribution, this constant cancels out in $w^*(\boldsymbol{\theta}^{(s)})$. And, although this estimator is biased, the bias is small, and provides good gains in variance reduction compared with the non-standardized option [343, Chap. 3].

A nice by-product of implementing IS is that it easily allows the calculation of the marginal likelihood. In particular, we know from Bayes' rule that

$$p(\boldsymbol{y})^{-1} = \frac{\pi(\boldsymbol{\theta} \mid \boldsymbol{y})}{p(\boldsymbol{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})},$$

then,

$$\begin{aligned}\int_{\boldsymbol{\Theta}} p(\boldsymbol{y})^{-1} q(\boldsymbol{\theta}) d\boldsymbol{\theta} &= \int_{\Theta} \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})} \pi(\boldsymbol{\theta} \mid \boldsymbol{y}) d\boldsymbol{\theta} \\ &= \mathbb{E}_\pi \left[\frac{q(\boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})}\right].\end{aligned}$$

Thus, an estimate of the marginal likelihood is $\left[\frac{1}{S}\sum_{s=1}^{S} \frac{q(\boldsymbol{\theta}^{*(s)})}{p(\boldsymbol{y}|\boldsymbol{\theta}^{*(s)}) \times \pi(\boldsymbol{\theta}^{*(s)})}\right]^{-1}$. This is the Gelfand-Dey method to calculate the marginal likelihood [147] (see subsection 10.4.3).

Example: Cauchy distribution

Let's assume that the posterior distribution is Cauchy with parameters 0 and 1. We perform an importance sampling algorithm using as proposals a standard normal distribution and a Student's t distribution with 3 degrees of freedom. The following code shows how to do this.

R code. Importance sampling: Cauchy distribution

```
rm(list = ls()); set.seed(010101)
S <- 20000 # Size proposal
# Importance sampling from standard normal proposal
thetaNs <- rnorm(S)
wNs <- dcauchy(thetaNs)/dnorm(thetaNs)
wNstars <- wNs/sum(wNs)
L <- 10000 # Size posterior
thetaCauchyN <- sample(thetaNs, L, replace = TRUE, prob =
    wNstars)
h <- hist(thetaCauchyN, breaks=50, col="blue", xlab="x",
    main="Cauchy draws from importance sampling: Normal
    standard proposal")
pfit <- seq(min(thetaCauchyN),max(thetaCauchyN),length=50)
yfit<-dcauchy(pfit)
yfit <- yfit*diff(h$mids[1:2])*length(thetaCauchyN)
lines(pfit, yfit, col="red", lwd=2)
# Importance sampling from Student's t proposal
df <- 3
thetaTs <- rt(S, df = df)
wTs <- dcauchy(thetaTs)/dt(thetaTs, df = df)
wTstars <- wTs/sum(wTs)
thetaCauchyT <- sample(thetaTs, L, replace = TRUE, prob =
    wTstars)
h <- hist(thetaCauchyT, breaks=50, col="blue", xlab="x",
    main="Cauchy draws from importance sampling: Student's t
     proposal")
pfit <- seq(min(thetaCauchyT),max(thetaCauchyT),length=50)
yfit<-dcauchy(pfit)
yfit <- yfit*diff(h$mids[1:2])*length(thetaCauchyT)
lines(pfit, yfit, col="red", lwd=2)
plot(wNstars, main = "Importance sampling: Cauchy
    distribution", ylab = "Weights", xlab = "Iterations")
points(wTstars, col = "blue")
legend("topright", legend = c("Normal", "Student's t"), col
    = c("black", "blue"), pch = c(1, 1))
```

Figure 4.4 shows the weights using the standard normal distribution (black dots) and the Student's t-distribution with 3 degrees of freedom (blue dots) as proposals. We observe that a few draws carry too much weight when using the normal proposal; this occurs because the normal distribution has much lighter tails compared to the Cauchy distribution. In contrast, using the Student's t-distribution with 3 degrees of freedom improves this situation.

Figures 4.5 and 4.6 show the histograms of the posterior draws using the normal and Student's t-distributions, respectively, along with the density of the Cauchy distribution. The spike in the posterior draws from the standard

normal proposal arises due to the lighter tails of the standard normal compared to the Cauchy distribution, consequently assigning too much weight to a specific draw from the normal distribution.

FIGURE 4.4
Importance sampling: 1000 weights in the Cauchy distribution example using the standard normal and student's t with 3 degrees of freedom as proposals.

4.3 Particle filtering

Now, we consider the scenario where we need to sample from a posterior distribution whose dimension increases over time, $\pi(\boldsymbol{\theta}_{0:t} \mid \boldsymbol{y}_{0:t})$, for $t = 0, 1, \ldots$. The challenge arises from the fact that, even if this posterior distribution is known, the computational complexity of implementing a sampling scheme in this context increases linearly with t. This makes MCMC methods, which operate in batch mode and require a complete re-run whenever new information becomes available, less optimal. Consequently, we present sequential algorithms, which operate incrementally as new data becomes available, and are often a better alternative. These algorithms are typically faster and are well-suited for scenarios requiring real-time updates, commonly referred to as online mode.

Specifically, we consider the dynamic system in the *state-space* representation. This is a system where there is an *unobservable state vector* $\boldsymbol{\theta}_t \in \mathbb{R}^K$, and an observed variable $\boldsymbol{Y}_t$, $t = 0, 1, \ldots$ such that (i) $\boldsymbol{\theta}_t$ is a *Markov process*,

FIGURE 4.5
Importance sampling: Draws from a Cauchy distribution using the standard normal as proposal.

FIGURE 4.6
Importance sampling: Draws of a Cauchy distribution using the Student's t with 3 degrees of freedom as proposal.

this is, $\pi(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{1:t-1}) = \pi(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1})$, $t = 1, 2, \ldots$, all the relevant information to define $\boldsymbol{\theta}_t$ is in $\boldsymbol{\theta}_{t-1}$,[1] and ii) $\boldsymbol{Y}_t \perp \boldsymbol{Y}_s \mid \boldsymbol{\theta}_t$, $s < t$, there is independence between observable variables regarding their history conditional on the actual state vector. We can see in Figure 4.7 a graphical representation of the dynamic system.

Notes: The figure illustrates the structure of a *state-space* model where the latent states $\boldsymbol{\theta}_t$ evolve according to $\pi(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1})$, and the observations Y_t depend on the states via $p(Y_t \mid \boldsymbol{\theta}_t)$.

FIGURE 4.7
State-space model representation.

Formally,

$$\begin{aligned} \boldsymbol{\theta}_t &= h(\boldsymbol{\theta}_{t-1}, \boldsymbol{w}_t) && \text{(State equations)} \\ Y_t &= f(\boldsymbol{\theta}_t, \mu_t) && \text{(Observation equation)}, \end{aligned}$$

where $\boldsymbol{w}_t$ and μ_t are stochastic errors such that their probability distributions define the transition density $\pi(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1})$ and observation density $p(Y_t \mid \boldsymbol{\theta}_t)$.

We present *particle filtering*, a specific case of *sequential Monte Carlo* (SMC), which is one of the most commonly used algorithms for scenarios requiring sequential updates of the posterior distribution as described by the *state-space* model.

The starting point is *sequential importance sampling* (SIS), originally proposed by [179], which is a modification of IS to compute an estimate of $\pi(\boldsymbol{\theta}_{0:t} \mid \boldsymbol{y}_{0:t})$ without altering the past trajectories $\left\{\boldsymbol{\theta}_{1:t-1}^{(s)}, s = 1, 2, \ldots, S\right\}$. The key idea is to use a proposal density that takes the form

[1] $\boldsymbol{\theta}_0$ comes from the given distribution $\pi(\boldsymbol{\theta}_0)$.

$$
\begin{aligned}
q(\boldsymbol{\theta}_{0:t} \mid \boldsymbol{y}_{0:t}) &= q(\boldsymbol{\theta}_{0:t-1} \mid \boldsymbol{y}_{1:t-1}) q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, \boldsymbol{y}_t) \\
&= q(\boldsymbol{\theta}_0) \prod_{h=1}^{t} q(\boldsymbol{\theta}_h \mid \boldsymbol{\theta}_{h-1}, \boldsymbol{y}_h).
\end{aligned}
$$

This proposal density allows calculating the weights sequentially,

$$
\begin{aligned}
w_t(\boldsymbol{\theta}_{0:t}^{(s)}) &= \frac{\pi(\boldsymbol{\theta}_{0:t}^{(s)} \mid \boldsymbol{y}_{0:t})}{q(\boldsymbol{\theta}_{0:t}^{(s)} \mid \boldsymbol{y}_{0:t})} \\
&= \frac{p(\boldsymbol{y}_{0:t} \mid \boldsymbol{\theta}_{0:t}^{(s)}) \pi(\boldsymbol{\theta}_{0:t}^{(s)})}{p(\boldsymbol{y}_{0:t}) q(\boldsymbol{\theta}_{0:t}^{(s)} \mid \boldsymbol{y}_{0:t})} \\
&= \frac{p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t^{(s)}) p(\boldsymbol{y}_{1:t-1} \mid \boldsymbol{\theta}_{0:t-1}^{(s)}) \pi(\boldsymbol{\theta}_t^{(s)} \mid \boldsymbol{\theta}_{t-1}^{(s)}) \pi(\boldsymbol{\theta}_{0:t-1}^{(s)})}{p(\boldsymbol{y}_{0:t}) q(\boldsymbol{\theta}_t^{(s)} \mid \boldsymbol{\theta}_{t-1}^{(s)}, \boldsymbol{y}_t) q(\boldsymbol{\theta}_{0:t-1}^{(s)} \mid \boldsymbol{y}_{1:t-1})} \\
&\propto \frac{p(\boldsymbol{y}_{1:t-1} \mid \boldsymbol{\theta}_{0:t-1}^{(s)}) \pi(\boldsymbol{\theta}_{0:t-1}^{(s)})}{q(\boldsymbol{\theta}_{0:t-1}^{(s)} \mid \boldsymbol{y}_{1:t-1})} \frac{p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t^{(s)}) \pi(\boldsymbol{\theta}_t^{(s)} \mid \boldsymbol{\theta}_{t-1}^{(s)})}{q(\boldsymbol{\theta}_t^{(s)} \mid \boldsymbol{\theta}_{t-1}, \boldsymbol{y}_t^{(s)})} \\
&\propto w_{t-1}(\boldsymbol{\theta}_{0:t-1}^{(s)}) \frac{p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t^{(s)}) \pi(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}^{(s)})}{q(\boldsymbol{\theta}_t^{(s)} \mid \boldsymbol{\theta}_{t-1}^{(s)}, \boldsymbol{y}_t)} \\
&\propto w_{t-1}^{*}(\boldsymbol{\theta}_{0:t-1}^{(s)}) \frac{p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t^{(s)}) \pi(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}^{(s)})}{q(\boldsymbol{\theta}_t^{(s)} \mid \boldsymbol{\theta}_{t-1}^{(s)}, \boldsymbol{y}_t)}.
\end{aligned}
$$

Take into account that $p(\boldsymbol{y}_{0:t})$ does not depend on $\boldsymbol{\theta}_{0:t}^{(s)}$. The term $\alpha_t(\boldsymbol{\theta}_{0:t}^{(s)}) = \frac{p(\boldsymbol{y}_t|\boldsymbol{\theta}_t^{(s)})\pi(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^{(s)})}{q(\boldsymbol{\theta}_t^{(s)}|\boldsymbol{\theta}_{t-1}^{(s)},\boldsymbol{y}_t)}$ is called the *incremental importance weight*, and implies that

$$
w_t(\boldsymbol{\theta}_{0:t}^{s}) = w_0(\boldsymbol{\theta}_0^{s}) \prod_{h=1}^{t} \alpha_h(\boldsymbol{\theta}_{1:h}^{(s)}).
$$

This algorithm possesses the desirable property of maintaining fixed computational complexity. Consequently, we sequentially obtain draws $\boldsymbol{\theta}_t^{(s)}$, referred to as particles: $\boldsymbol{\theta}_0^{(s)}$ is drawn from $q(\boldsymbol{\theta}_0)$ at $t = 0$, and subsequently, $\boldsymbol{\theta}_h^{(s)}$ is drawn from $q(\boldsymbol{\theta}_h \mid \boldsymbol{\theta}_{h-1}, \boldsymbol{y}_h)$ at $t = h$ [114, 62].

A relevant case is when the proposal distribution takes the form of the prior distribution, that is, $q(\boldsymbol{\theta}_{0:t} \mid \boldsymbol{y}_{0:t}) = \pi(\boldsymbol{\theta}_{0:t}) = \pi(\boldsymbol{\theta}_0) \prod_{h=1}^{t} \pi(\boldsymbol{\theta}_h \mid \boldsymbol{\theta}_{h-1})$. This implies that

$$
w_t(\boldsymbol{\theta}_{0:t}^{(s)}) \propto w_{t-1}^{*}(\boldsymbol{\theta}_{0:t-1}^{(s)}) p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t^{(s)}),
$$

this means that the *incremental importance weight* is given by $p(\boldsymbol{y}_t \mid \boldsymbol{\theta}_t^{(s)})$.

Algorithm A4 shows how to perform SIS [62]. We set $w_t^{(s)} := w_t(\boldsymbol{\theta}_{0:t}^{(s)})$ to

Algorithm A4 Sequential importance sampling algorithm

for $s = 1, \ldots, S$ **do**
  Sample $\boldsymbol{\theta}_0^{(s)}$ from $q(\boldsymbol{\theta}_0 \mid y_0)$
  Calculate the importance weights $w_0^{(s)} \propto \frac{p(y_0|\boldsymbol{\theta}_0^{(s)})\pi(\boldsymbol{\theta}_0^{(s)})}{q(\boldsymbol{\theta}_0^{(s)}|y_0)}$
end for
for $t = 1, \ldots, T$ **do**
  for $s = 1, \ldots, S$ **do**
    Draw particles $\boldsymbol{\theta}_t^{(s)}$ from $q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, \boldsymbol{y}_t)$
    Compute the weights $w_t^{(s)} = w_{t-1}^{*(s)} \frac{p(\boldsymbol{y}_t|\boldsymbol{\theta}_t^{(s)})\pi(\boldsymbol{\theta}_t^{(s)}|\boldsymbol{\theta}_{t-1}^{(s)})}{q(\boldsymbol{\theta}_t^{(s)}|\boldsymbol{\theta}_{t-1}^{(s)},\boldsymbol{y}_t)}$
  end for
  Standardize the weights $w_t^{*(s)} = \frac{w_t^{(s)}}{\sum_{h=1}^{S} w_t^{(h)}}$, $s = 1, 2, \ldots, S$
end for

simplify notation.

Example: Dynamic linear model

Let's assume that the state-space representation is

$$\begin{aligned} \theta_t &= \theta_{t-1} + w_t && \text{(State equation)} \\ Y_t &= \phi\theta_t + \mu_t && \text{(Observation equation)}, \end{aligned}$$

where $w_t \sim N(0, \sigma_w^2)$ and $\mu_t \sim N(0, \sigma_\mu^2)$, $t = 1, 2, \ldots, 50$. In addition, we use the proposal distribution $q(\theta_t \mid y_t) = \pi(\theta_t)$, which is normal with mean θ_{t-1} and variance σ_w^2. Then, the weights are given by the recursion

$$w_t^{(s)} \propto w_{t-1}^{*(s)} p(y_t \mid \theta_t, \sigma_\mu^2),$$

where $p(y_t \mid \theta_t, \sigma_\mu^2)$ is $N(\phi\theta_t, \sigma_\mu^2)$.

We can compute the mean and standard deviation of the state at each t using

$$\hat{\theta}_t = \sum_{s=1}^{S} w_t^{*(s)} \theta_t^{(s)}$$

and

$$\hat{\sigma}_\theta = \left(\sum_{s=1}^{S} w_t^{*(s)} \theta_t^{2(s)} - \hat{\theta}_t^2 \right)^{1/2}.$$

The following code demonstrates the implementation of this algorithm, setting $\sigma_w^2 = \sigma_\mu^2 = 1$ and $\phi = 0.5$. First, we simulate the process, and then we implement the SIS algorithm.

R code. Sequential importance sampling: Dynamic linear model

```r
rm(list = ls()); set.seed(010101)
S <- 50000 # Number of particles
sigma_w <- 1  # State noise
sigma_mu <- 1  # Observation noise
phi <- 0.5 # Coefficient in observation equation
T <- 50 # Sample size
# Simulate true states and observations
theta_true <- numeric(T); y_obs <- numeric(T)
theta_true[1] <- rnorm(1, mean = 0, sd = sigma_w)  # Initial state
for (t in 2:T) {
  theta_true[t] <- rnorm(1, mean = theta_true[t-1], sd = sigma_w)
}
y_obs <- rnorm(T, mean = phi*theta_true, sd = sigma_mu)
# Sequential Importance Sampling (SIS)
particles <- matrix(0, nrow = S, ncol = T)
weights <- matrix(0, nrow = S, ncol = T)
weightsSt <- matrix(0, nrow = S, ncol = T)
# Initialization
particles[, 1] <- rnorm(S, mean = 0, sd = sigma_w)  # Sample initial particles
weights[, 1] <- dnorm(y_obs[1], mean = phi*particles[, 1], sd = sigma_mu)  # Importance weights
weightsSt[, 1] <- weights[, 1] / sum(weights[, 1])  # Standardized weights
# Sequential updating
for (t in 2:T) {
  # Propagate particles
  particles[, t] <- rnorm(S, mean = particles[, t-1], sd = sigma_w)
  # Compute weights
  weights[, t] <- weightsSt[, t-1] * dnorm(y_obs[t], mean = phi*particles[, t], sd = sigma_mu) # Recursive weight update
  weightsSt[, t] <- weights[, t] / sum(weights[, t])  # Normalize weights
}
# Estimate the states (weighted mean)
FilterDist <- colSums(particles * weightsSt)
SDFilterDist <- (colSums(particles^2 * weightsSt) - FilterDist^2)^0.5
library(dplyr); library(ggplot2); library(latex2exp)
ggplot2::theme_set(theme_bw())
df <- tibble(t = 1:T, mean = FilterDist, lower = FilterDist - 2*SDFilterDist, upper = FilterDist + 2*SDFilterDist, theta_true = theta_true)
# Function to plot
plot_filtering_estimates <- function(df) {
  p <- ggplot(data = df, aes(x = t)) + geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 1, fill = "lightblue") + geom_line(aes(y = theta_true), colour = "black", alpha = 1, linewidth = 0.5) + geom_line(aes(y = mean), colour = "blue", linewidth = 0.5) + ylab(TeX("$\\theta_{t}$")) + xlab("Time")
  print(p)
}
plot_filtering_estimates(df)
```

Figure 4.8 shows the trajectory of the true state vector (black line), the posterior mean (blue line), and the area defined by $\pm 2\hat{\sigma}_\theta$ (light blue shaded area).

FIGURE 4.8
State in linear state-space model: True and mean estimate using sequential importance sampling, $T = 50$.

Sequential importance sampling is effective for sampling from the posterior distribution in the short term. However, it is important to note that SIS is a particular case of IS and, consequently, inherits the drawbacks of importance sampling. In particular, the variance of the weights increases exponentially with t [231]. This implies that, as t increases, the importance weights tend to degenerate in the long run; that is, all probability mass concentrates on a few weights, a phenomenon known as sample impoverishment or weight degeneracy. This is because is impossible to accurately represent a distribution on a space of arbitrarily high dimension with a sample of fixed, finite size. This phenomenon can be observed, for instance, in the dynamic linear model example, where the highest standardized weight at $t = 50$ is 53%, and 7 out of 50,000 particles account for 87% of the total probability.

Given that, in practice, we are often interested in lower-dimensional marginal distributions, ideas from sampling/importance resampling can be employed. This strategy avoids the accumulation of errors due to resetting the system, although resampling introduces some additional Monte Carlo variation. [170] proposed the *Bootstrap filter*, where, at each time step, resampling is performed by drawing S particles from the current set using the standardized weights as probabilities of selection. This ensures that particles with small weights have a low probability of being selected. After resampling, the stan-

dardized weights are set equal to $1/S$. Note that the *Bootstrap filter* involves multiple iterations of the SIR algorithm, which implies that the resampled trajectories are no longer independent. This multinomial resampling provides an unbiased approximation to the posterior distribution obtained by SIS [115].

Algorithm A5 shows how to perform the *particle filter*. We set $w_t^{(s)} := w_t(\boldsymbol{\theta}_{0:t}^{(s)})$ to simplify notation. [115].

Algorithm A5 The *particle filter* algorithm

for $s = 1, \ldots, S$ **do**
  Sample $\boldsymbol{\theta}_0^{(s)}$ from $q(\boldsymbol{\theta}_0 \mid y_0)$
  Calculate the importance weights $w_0^{(s)} \propto \frac{p(y_0|\boldsymbol{\theta}_0^{(s)})\pi(\boldsymbol{\theta}_0^{(s)})}{q(\boldsymbol{\theta}_0^{(s)}|y_0)}$
end for
Standardize the weights $w_0^{*(s)} = \frac{w_0^{(s)}}{\sum_{h=1}^{S} w_0^{(h)}}$, $s = 1, 2, \ldots, S$
Select S particles from $\left\{\boldsymbol{\theta}_0^{(s)}, w_0^{*(s)}\right\}$ to obtain $\left\{\boldsymbol{\theta}_0^{r(s)}, 1/S\right\}$
for $t = 1, \ldots, T$ **do**
  for $s = 1, \ldots, S$ **do**
    Draw particles $\boldsymbol{\theta}_t^{(s)}$ from $q_t(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_{t-1}, \boldsymbol{y}_t)$
    Set $\boldsymbol{\theta}_{1:t}^{(s)} \leftarrow (\boldsymbol{\theta}_{1:t-1}^{r(s)}, \boldsymbol{\theta}_t^{(s)})$
    Compute the weights $\alpha_t^{(s)} = \frac{p(\boldsymbol{y}_t|\boldsymbol{\theta}_t^{(s)})\pi(\boldsymbol{\theta}_t^{(s)}|\boldsymbol{\theta}_{t-1}^{(s)})}{q(\boldsymbol{\theta}_t^{(s)}|\boldsymbol{\theta}_{t-1}^{(s)},\boldsymbol{y}_t)}$
  end for
  Standardize the weights $w_t^{*(s)} = \frac{w_t^{(s)}}{\sum_{h=1}^{S} w_t^{(h)}}$, $s = 1, 2, \ldots, S$
  Select S particles from $\left\{\boldsymbol{\theta}_{1:t}^{(s)}, w_t^{*(s)}\right\}$ to obtain $\left\{\boldsymbol{\theta}_{1:t}^{r(s)}, 1/S\right\}$
end for

Example: Dynamic linear model continues

Let's apply the SIS algorithm to the dynamic linear model with a sample size of 200. Figure 4.9 illustrates the performance of sequential importance sampling. We observe that the algorithm's performance deteriorates as t increases. This is due to particle degeneration; at $t = 200$, a single particle holds a weight close to 100%.

Let's perform particle filtering in this example. The following code illustrate the procedure. Figure 4.10 shows the performance of particle filtering in this example. There is the true state vector (black line), the means based on $\left\{\boldsymbol{\theta}_{1:t}^{(s)}, w_t^{*(s)}\right\}$ (blue line) and $\left\{\boldsymbol{\theta}_{1:t}^{r(s)}, 1/S\right\}$ (purple line), and the area defined by $\pm 2\hat{\sigma}_\theta$ based on the former (light blue shaded area). Note that the particle filtering algorithm has better performance than the SIS algorithm.

FIGURE 4.9
State in linear state-space model: True and mean estimate using sequential importance sampling, $T = 200$.

FIGURE 4.10
State in linear state-space model: True and mean estimate using particle filtering, $T = 200$.

R code. Particle filtering: Dynamic linear model

```
rm(list = ls()); set.seed(010101)
S <- 50000 # Number of particles
sigma_w <- 1; sigma_mu <- 1  #  # State and observation noises
phi <- 0.5 # Coefficient in observation equation
T <- 200 # Sample size
# Simulate true states and observations
theta_true <- numeric(T); y_obs <- numeric(T)
theta_true[1] <- rnorm(1, mean = 0, sd = sigma_w)
for (t in 2:T) {
  theta_true[t] <- rnorm(1, mean = theta_true[t-1], sd = sigma_w)
}
y_obs <- rnorm(T, mean = phi*theta_true, sd = sigma_mu)
# Particle filtering
particles <- matrix(0, nrow = S, ncol = T)  # Store particles
particlesT <- matrix(0, nrow = S, ncol = T)  # Store resampling particles
weights <- matrix(0, nrow = S, ncol = T)   # Store weights
weightsSt <- matrix(0, nrow = S, ncol = T)   # Store standardized weights
weightsSTT <- matrix(1/S, nrow = S, ncol = T)   # Store standardized weights
logalphas <- matrix(0, nrow = S, ncol = T)   # Store log incremental weights
particles[, 1] <- rnorm(S, mean = 0, sd = sigma_w)
weights[, 1] <- dnorm(y_obs[1], mean = phi*particles[, 1], sd = sigma_mu)  # Importance weights
weightsSt[, 1] <- weights[, 1] / sum(weights[, 1])  # Normalize weights
ind <- sample(1:S, size = S, replace = TRUE, prob = weightsSt[, 1]) # Resample
particles[, 1] <- particles[ind, 1] # Resampled particles
particlesT[, 1] <- particles[, 1] # Resampled particles
# Sequential updating
pb <- winProgressBar(title = "progress bar", min = 0, max = T, width = 300)
for (t in 2:T) {
  particles[, t] <- rnorm(S, mean = particles[, t-1], sd = sigma_w)
  logalphas[, t] <- dnorm(y_obs[t], mean = phi*particles[, t], sd = sigma_mu, log = TRUE)
  weights[, t] <- exp(logalphas[, t])
  weightsSt[, t] <- weights[, t] / sum(weights[, t])
  if(t < T){
    ind <- sample(1:S, size = S, replace = TRUE, prob = weightsSt[, t])
    particles[, 1:t] <- particles[ind, 1:t]
  }else{
    ind <- sample(1:S, size = S, replace = TRUE, prob = weightsSt[, t])
    particlesT[, 1:t] <- particles[ind, 1:t]
  }
  setWinProgressBar(pb, t, title=paste( round(t/T*100, 0), " % done"))
}
close(pb)
```

R code. Particle filtering: Dynamic linear model

```
FilterDist <- colSums(particles * weightsSt)
SDFilterDist <- (colSums(particles^2 * weightsSt) - FilterDist^2)^0.5
FilterDistT <- colSums(particlesT * weightsSTT)
SDFilterDistT <- (colSums(particlesT^2 * weightsSTT) - FilterDistT^2)^0.5
MargLik <- colMeans(weights)
plot(MargLik, type = "l")
library(dplyr)
library(ggplot2)
require(latex2exp)
ggplot2::theme_set(theme_bw())
df <- tibble(t = 1:T, mean = FilterDist, lower = FilterDist - 2*SDFilterDist, upper = FilterDist+ 2*SDFilterDist, meanT = FilterDistT, lowerT = FilterDistT - 2*SDFilterDistT,
upperT = FilterDistT + 2*SDFilterDistT, x_true = theta_true)
plot_filtering_estimates <- function(df) {
  p <- ggplot(data = df, aes(x = t)) + geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 1, fill = "lightblue") + geom_line(aes(y = x_true), colour = "black", alpha = 1, linewidth = 0.5) + geom_line(aes(y = mean), colour = "blue", linewidth = 0.5) +
  geom_line(aes(y = meanT), colour = "purple", linewidth = 0.5) +   ylab(TeX("$\\theta_{t}$")) + xlab("Time")
  print(p)
}
plot_filtering_estimates(df)
```

Algorithm A5 performs resampling at every time step. However, it is common to perform resampling only when the effective sample size of the particles ($ESS = (\sum_{s=1}^{S}(w_t^{*(s)})^2)^{-1}$) falls below a specific threshold, such as 50% of the initial number of particles. Note that when $w_t^{*(s)} = 1/S$, the effective sample size is S, the total number of particles. Additionally, we should use $\left\{\boldsymbol{\theta}_{1:t}^{(s)}, w_t^{*(s)}\right\}$ to estimate the posterior distribution, as it results in lower Monte Carlo error compared to calculations based on $\left\{\boldsymbol{\theta}_{1:t}^{r(s)}, 1/S\right\}$ [62]. Finally, an estimate of the marginal likelihood can be obtained using

$$\hat{p}(y_t) = \frac{1}{S}\sum_{s=1}^{S} w_t^{(s)}.$$

Particle filtering offers several advantages, such as being quick and easy to implement, its modularity—allowing one to simply adjust the expressions for

the importance distribution and weights when changing the problem—and its suitability for parallel algorithms. Moreover, it enables straightforward sequential inference for very complex models.

However, there are also disadvantages. The resampling step introduces extra Monte Carlo variability. Using the state transition (prior) density as the importance distribution often leads to poor performance, manifested in a lack of robustness with respect to the observed sequence. For instance, performance deteriorates when outliers occur in the data or when the variance of the observation noise is small. Furthermore, the procedure is not well suited for sampling from $\pi(\boldsymbol{\theta}_{0:t} \mid y_{1:t})$ because most particles originate from the same ancestor.

Alternative resampling approaches, such as residual resampling [258] and systematic resampling [66], preserve unbiasedness while reducing variance. Additionally, auxiliary particle filtering [63] can help decrease Monte Carlo variability.

Lastly, estimating fixed parameters such as σ_w^2, σ_μ^2, and ϕ in the dynamic linear model poses a challenge. Various methods exist to address this issue; see [221, 222] for a comprehensive review and [11] for a seminal work in *particle MCMC* methods.

4.4 Convergence diagnostics

MCMC methods rely on *irreducibility*, *positive recurrence*, and *aperiodicity*, ensuring that, after a sufficient burn-in (warm-up) period, the posterior draws are sampled from the invariant stationary posterior distribution. This can be achieved by running multiple chains initiated at different points and then mixing them, or by running a single longer chain. In most of the second part of this book, we follow the latter approach, as suggested by [165].

In this section, we present diagnostics to assess whether the sample draws come from the stationary posterior distribution. First, we calculate the numerical standard error associated with the MCMC algorithm. Next, we review the effective number of simulation draws and various convergence tests. Finally, we examine potential errors in the posterior simulator.

4.4.1 Numerical standard error

Many times, the goal in Bayesian inference is to obtain a set of independent draws $\boldsymbol{\theta}^{(s)}$, $s = 1, 2, \ldots, S$, from the posterior distribution, such that a measure of interest can be estimated with reasonable precision. In particular, we

approximate (4.1) using (4.2). By the central limit theorem, we know that

$$\frac{\bar{h}(\boldsymbol{\theta})_S - \mathbb{E}_\pi[h(\boldsymbol{\theta})]}{\sigma_h(\boldsymbol{\theta})/\sqrt{S}} \xrightarrow{d} N(0,1), \tag{4.4}$$

where $\sigma_h^2(\boldsymbol{\theta})$ is the variance of $h(\boldsymbol{\theta})$.

If we have independent draws, we can estimate $\sigma_h^2(\boldsymbol{\theta})$ using the posterior draws as follows:

$$\hat{\sigma}_{Sh}^2(\boldsymbol{\theta}) = \frac{1}{S}\sum_{s=1}^{S}\left[h(\boldsymbol{\theta}^{(s)})\right]^2 - \left[\bar{h}(\boldsymbol{\theta})_S\right]^2.$$

However, if there are dependent draws, we have

$$\hat{\sigma}_{Sh}^{2*}(\boldsymbol{\theta}) = \frac{1}{S}\left\{\sum_{s=1}^{S}\left[h(\boldsymbol{\theta}^{(s)}) - \bar{h}(\boldsymbol{\theta})_S\right]^2 + 2\sum_{l=k+1}^{K}\left(h(\boldsymbol{\theta}^{(l)}) - \bar{h}(\boldsymbol{\theta})\right)\left(h(\boldsymbol{\theta}^{(l-k)}) - \bar{h}(\boldsymbol{\theta})\right)\right\}.$$

The *numerical standard error* is given by $\sigma_h(\boldsymbol{\theta})/\sqrt{S}$ and serves as a measure of the approximation error in the Monte Carlo integration. Note that this error can be decreased by increasing S. For instance, $S = 1000$ implies an error proportional to 3.2%, while $S = 10000$ reduces the error to approximately 1%.

4.4.2 Effective number of simulation draws

MCMC posterior draws are not independent; therefore, the effective sample size of the posterior chains is not equal to S. To assess the effective sample size of the posterior draws, we use the following measure:

$$S_{\text{ef}} = \frac{S}{1 + 2\sum_{k=1}^{\infty}\rho_k(h)},$$

where $\rho_k(h)$ is the autocorrelation of the sequence $h(\boldsymbol{\theta})$ at lag k.

The sample counterpart of this expression is:

$$\hat{S}_{\text{ef}} = \frac{S}{1 + 2\sum_{k=1}^{K}\hat{\rho}_k(h)},$$

where

$$\hat{\rho}_k(h) = \frac{\sum_{l=k+1}^{K}\left(h(\boldsymbol{\theta}^{(l)}) - \bar{h}(\boldsymbol{\theta})\right)\left(h(\boldsymbol{\theta}^{(l-k)}) - \bar{h}(\boldsymbol{\theta})\right)}{\sum_{s=1}^{K}\left(h(\boldsymbol{\theta}^{(s)}) - \bar{h}(\boldsymbol{\theta})\right)^2}.$$

If $\hat{\rho}_k(h)$ declines to zero slowly as k increases, it indicates significant memory in the draws. Consequently, the effective sample size of the posterior draws is small, and it becomes necessary to either decrease the autocorrelation or increase the number of posterior draws.

Note that

$$\hat{\sigma}^{2*}_{Sh}(\boldsymbol{\theta}) = \hat{\sigma}^2_{Sh}(\boldsymbol{\theta})\,(1 + 2\sum_{k=1}^{K} \hat{\rho}_k(h)),$$

where $\hat{\sigma}^{2*}_{Sh}(\boldsymbol{\theta})$ and $\hat{\sigma}^2_{Sh}$ are the simulation variances using dependent and independent draws, and $\hat{\kappa}(h) = (1+2\sum_{k=1}^{K} \hat{\rho}_k(h))$ is called the *inefficiency factor*, which represents the inflation of the simulation variance due to autocorrelation in the draws. Values near one indicate draws with little correlation.

4.4.3 Tests of convergence

Regarding convergence issues, there are several diagnostics to assess the adequacy of the posterior chains. In particular, graphical approaches such as trace plots and autocorrelation plots are widely used. Trace plots display the sampled values of a parameter (or multiple parameters) as a function of the iteration number, while autocorrelation plots graphically represent $\hat{\rho}_k$. The latter shows how correlated the values of $\boldsymbol{\theta}$, or functions of $\boldsymbol{\theta}$, are at different lags. Trace plots should fluctuate around a stable mean, exploring the entire parameter space without becoming stuck in any particular region. Autocorrelation plots, on the other hand, should exhibit values close to zero or diminish quickly as the lag increases.

Additionally, Geweke's test [160] provides a simple two-sample test of means. If the mean of the first window (10% of the chain) is not significantly different from the mean of the second window (50% of the chain), we do not reject the null hypothesis that the two segments of the chain are drawn from the same stationary distribution.

The Raftery and Lewis test [325] is designed to calculate the approximate number of iterations (S), burn-in (b), and thinning parameter (d) required to estimate $p\,[H(\boldsymbol{\theta}) \leq h]$, where $H(\boldsymbol{\theta}) : \mathcal{R}^K \to \mathcal{R}$. This calculation is based on a specific quantile of interest (q), precision (r), and probability (p). The diagnostic is based on the dependence factor, $I = \frac{S+b}{S_{\text{Min}}}$, where $S_{\text{Min}} = \Phi^{-1}\left(\frac{1}{2}(p+1)\right)^2 q(1-q)/r^2$, and $\Phi(\cdot)$ is the standard normal cumulative distribution function. Values of I much greater than 5 indicate a high level of dependence.

Heidelberger and Welch's test [188] uses a Cramér-von Mises statistic to test the null hypothesis that the sampled values, $\boldsymbol{\theta}^{(s)}$, are drawn from a stationary distribution. The statistic is given by

$$\text{CVM}(B_S) = \int_0^1 B_S(t)^2 dt,$$

where $B_S(t) = \frac{S_{[St]} - [St]\bar{\boldsymbol{\theta}}^S}{\sqrt{Sp(0)}}$, $S_S = \sum_{s=1}^{S} \boldsymbol{\theta}^{(s)}$, $\bar{\boldsymbol{\theta}}^S = S_S/S$, and $p(0)$ is the spectral density at 0, with $0 \leq t \leq 1$. Under the null hypothesis, $B_S(t)$ converges in distribution to a Brownian bridge.

This test is recursively applied until either the null hypothesis is not rejected, or $s = 50\%$ of the chain has been discarded. Subsequently, the half-width test calculates a 95% credible interval for the mean using the portion of the chain that passed the stationarity test. If the ratio of the half-width of this interval to the mean is less than 0.1, the test is considered passed. This indicates no evidence to reject the null hypothesis that the estimated mean is accurate and stable.

There are other diagnostics in Bayesian inference that we do not mention here, such as the Gelman and Rubin test [154]. This is because we focus on the available diagnostics in our Graphical User Interface (GUI).

4.4.4 Checking for errors in the posterior simulator

In this book, we provide basic code templates to get posterior draws for performing inference under the Bayesian framework when there is no closed-form solution. We are prone to making mistakes and greatly appreciate your feedback to help improve our code and identify any other potential issues. One way to check if our code works correctly is to perform simulations where the population parameters are known. If the code is functioning properly, the posterior estimates should converge to these values as the sample size increases due to the Bayesian consistency. This is an informal approach to identifying potential mistakes.

[162] offers a more formal method for code validation. The starting point is the joint density $p(\boldsymbol{y}, \boldsymbol{\theta}) = p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$ and a test function $h(\boldsymbol{y}, \boldsymbol{\theta})$ such that $\sigma_h^2 = \mathrm{Var}[h(\boldsymbol{y}, \boldsymbol{\theta})] < \infty$.

Assume that there is a *marginal-conditional simulator* for the joint distribution of $\boldsymbol{y}$ and $\boldsymbol{\theta}$:

$$\begin{aligned}
\boldsymbol{\theta}^{(s)} &\sim \pi(\boldsymbol{\theta}) \\
\boldsymbol{y}^{(s)} &\sim p(\boldsymbol{y} \mid \boldsymbol{\theta}^{(s)}) \\
h^{(s)} &= h(\boldsymbol{y}^{(s)}, \boldsymbol{\theta}^{(s)}).
\end{aligned}$$

The sequence $\left\{\boldsymbol{y}^{(s)}, \boldsymbol{\theta}^{(s)}\right\}$ is i.i.d., $\bar{h}_S$ converges almost surely to $\mathbb{E}[h(\boldsymbol{y}, \boldsymbol{\theta})]$, and there is convergence in distribution when $\bar{h}_S$ is well standardized (see Equation 4.4) and $\hat{\sigma}_{Sh}^2(\boldsymbol{\theta})$ converges to $\sigma_h^2(\boldsymbol{\theta})$ almost surely.

A posterior simulator produces draws $\boldsymbol{\theta}^{(s)}$ given a particular realization $\boldsymbol{y}_{\text{Obs}}$, using the transition density $q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(s-1)}, \boldsymbol{y}_{\text{Obs}})$. Thus, a *successive-conditional simulator* consists of an initial draw $\boldsymbol{\theta}^{(0)}$ from $\pi(\boldsymbol{\theta})$ followed by:

$$\begin{aligned}
\boldsymbol{y}^{(l)} &\sim p(\boldsymbol{y} \mid \boldsymbol{\theta}^{(l-1)}) \\
\boldsymbol{\theta}^{(l)} &\sim q(\boldsymbol{\theta} \mid \boldsymbol{y}^{(l)}, \boldsymbol{\theta}^{(l-1)}) \\
h^{(l)} &= h(\boldsymbol{y}^{(l)}, \boldsymbol{\theta}^{(l)}),
\end{aligned}$$

where $\bar{h}_L = L^{-1} \sum_{l=1}^{L} h(\boldsymbol{y}^{(l)}, \boldsymbol{\theta}^{(l)})$ converges almost surely to $\mathbb{E}[h(\boldsymbol{y}, \boldsymbol{\theta})]$, and

there is convergence in distribution when $\bar{h}_L$ is well standardized, and $\hat{\sigma}^{*2}_{Lh}(\boldsymbol{\theta})$ converges to $\sigma^2_h(\boldsymbol{\theta})$ almost surely, for $l = 1, 2, \ldots, L$. Thus,

$$\frac{\bar{h}_S - \bar{h}_L}{\left(S^{-1}\hat{\sigma}^2_{Sh}(\boldsymbol{\theta}) + L^{-1}\hat{\sigma}^{*2}_{Lh}(\boldsymbol{\theta})\right)^{1/2}} \xrightarrow{d} N(0,1).$$

Thus, we can test H_0. $\bar{h}_S - \bar{h}_L = 0$ versus H_1. $\bar{h}_S - \bar{h}_L \neq 0$. Rejection of the null indicates potential errors in implementing the posterior simulator.

Example: Mining disaster change point continues

Let's revisit the mining disaster change point example from subsection 4.1.1 and examine some convergence diagnostics for the posterior draws of the rate of disasters after the change point (λ_2). The following code demonstrates how to perform these diagnostics using the **R** package *coda*. For clarity and replicability of the results, we present the Gibbs sampler again.

Figures 4.11 and 4.12 show the trace and autocorrelation plots. We observe that the posterior draws of λ_2 appear stationary around their mean, and the autocorrelation decreases rapidly to zero.

The mean and standard deviation of the rate after the change point are 0.92 and 0.12, respectively. The naive and time series standard errors are 0.0008245 and 0.0008945, respectively. The naive standard error assumes iid posterior draws, whereas the time series standard error accounts for autocorrelation. Both standard errors are very similar, indicating a low level of autocorrelation, which is consistent with the results shown in Figure 4.12. The effective sample size of the posterior draws is 16,991, while the total number of posterior draws is 20,000 after a burn-in period of 1,000.

The Geweke test statistic is 1.43, which implies no statistical evidence to reject the null hypothesis of equal means in the two segments of the posterior draws. The Raftery and Lewis test yields a dependence factor near 1, indicating a low level of dependence. The Heidelberger and Welch test does not reject the null hypothesis of stationarity for the posterior draws and also confirms that the mean is accurate and stable.

In summary, all posterior diagnostics indicate that the posterior draws originate from an invariant stationary distribution.

The second part of the code implements the proposal by [162] to assess the reliability of the posterior simulator. The parameter vector is defined as $\boldsymbol{\theta} = [\lambda_1 \ \lambda_2 \ H]$, and the first moments of these parameters are used as test functions. We do not reject the null hypothesis of equal means across the three test functions, indicating that the posterior simulator is functioning correctly.

To evaluate the effectiveness of the test, we run the *marginal-conditional simulator* with prior parameters $\alpha_{l0} = 0.5$ and $\beta_{l0} = 1$, $l = 1, 2$. In contrast, for the *successive-conditional simulator*, we use prior parameters $\alpha_{l0} = 1$ and $\beta_{l0} = 0.5$, $l = 1, 2$. In this case, we reject the null hypothesis in two out of three test functions, suggesting that the test performs well in this example to detect issues.

R code. Posterior diagnostics: The mining disaster changepoint

```
rm(list = ls())
set.seed(010101)
dataset<-read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/MiningDataCarlin.csv",header=T)
attach(dataset)
str(dataset)
a10 <- 0.5; a20 <- 0.5
b10 <- 1; b20 <- 1
y <- Count
sumy <- sum(Count); T <- length(Count)
theta1 <- NULL; theta2 <- NULL
kk <- NULL; H <- 60
MCMC <- 20000; burnin <- 1000; S <- MCMC + burnin; keep <- (burnin+1):S
pb <- winProgressBar(title = "progress bar", min = 0, max = S, width = 300)
for(s in 1:S){
  a1 <- a10 + sum(y[1:H])
  b1 <- b10+H
  theta11 <- rgamma(1,a1,b1)
  theta1 <- c(theta1,theta11)
  a2 <- a20 + sum(y[(1+H):T])
  b2 <- b20 + T-H
  theta22 <- rgamma(1,a2,b2)
  theta2 <- c(theta2,theta22)
  pp<-NULL
  for(l in 1:T){
    p <- exp(l*(theta22-theta11))*(theta11/theta22)^(sum(y[1:l]))
    pp <- c(pp,p)
  }
  prob <- pp/sum(pp)
  H <- sample(1:T,1,prob=prob)
  kk <- c(kk,H)
  setWinProgressBar(pb, s, title=paste( round(s/S*100, 0),"% done"))
}
close(pb)
library(coda); library(latex2exp)
theta1Post <- mcmc(theta1[keep]); summary(theta1Post)
HPost <- mcmc(kk); summary(HPost)
theta2Post <- mcmc(theta2[keep]); summary(theta2Post)
plot(theta2Post, density = FALSE, main = "Trace plot", ylab = TeX("$\\theta_{2}$"))
autocorr.plot(theta2Post, main = "Autocorrelation plot")
raftery.diag(theta2Post, q = 0.025, r = 0.005, s = 0.95)
geweke.diag(theta2Post, frac1 = 0.1, frac2 = 0.5)
heidel.diag(theta2Post, eps = 0.1, pvalue = 0.05)
effectiveSize(theta2Post)
```

FIGURE 4.11
Mining disaster change point: Trace plot of λ_2.

FIGURE 4.12
Mining disaster change point: Autocorrelation plot of λ_2.

R code. Posterior diagnostics: The mining disaster changepoint

```
# Marginal-conditional simulator
Theta1Prior <- rgamma(MCMC,a10,b10); Theta2Prior <- rgamma(MCMC,a20,b20)
kPrior <- sample(1:T, MCMC, replace = TRUE, prob = rep(1/T,T))
ytPrior <- function(par){
  y1t <- rpois(par[3], par[1])
  if(par[3] == T){y2t <- NULL
  }else{y2t <- rpois(T-par[3], par[2])
  }
  yt <- c(y1t, y2t)
  return(yt)
}
pars1 <- cbind(Theta1Prior, Theta2Prior, kPrior); Yt <- apply(pars1, 1, ytPrior)
parsmcmc1 <- coda::mcmc(pars1); Summ1 <- summary(parsmcmc1)
# Successive-conditional simulator
SucConSim <- function(a10, b10, a20, b20, par){
  y <- ytPrior(par)
  theta1 <- par[1]; theta2 <- par[2]; H <- par[3]
  a1 <- a10 + sum(y[1:H]);  b1 <- b10+H
  theta11 <- rgamma(1,a1,b1)
  if(H == T){ a2 <- a20
  }else{ a2 <- a20 + sum(y[(1+H):T])
  }
  b2 <- b20 + T-H; theta22 <- rgamma(1,a2,b2); pp<-NULL
  for(l in 1:T){
    p <- l*(theta22-theta11) + (sum(y[1:l]))*log(theta11/theta22)
    pp <- c(pp,p)
  }
  pps <- exp(pp - max(pp)); prob <- pps/sum(pps)
  H <- sample(1:T, 1, prob=prob)
  parNew <- list(y = y, pars = c(theta11, theta22, H))
  return(parNew)
}
a10 <- 0.5; b10 <- 1; a20 <- 0.5; b20 <- 1
# a10 <- 1; b10 <- 0.5; a20 <- 1; b20 <- 0.5
par1 <- rgamma(1,a10,b10); par2 <- rgamma(1,a20,b20)
par3 <- sample(1:T, 1, replace = TRUE, prob = rep(1/T,T))
pars2 <- matrix(NA, MCMC, 3); pars2[1,] <- c(par1, par2, par3)
for(s in 2:MCMC){
  Res <- SucConSim(a10 = a10, b10 = b10, a20 = a20, b20 = b20, par = pars2[s-1,])
  pars2[s, ] <- Res$pars
}
parsmcmc2 <- coda::mcmc(pars2); Summ2 <- summary(parsmcmc2)
TestGeweke <- function(j){
  Test <- (Summ1[["statistics"]][j,1] - Summ2[["statistics"]][j,1])/(Summ1[["statistics"]][j,4]+Summ2[["statistics"]][j,4])^0.5
  Reject <- abs(Test) > qnorm(0.975)
  return(list(Test = Test, Reject = Reject))
}
TestGeweke(1); TestGeweke(2); TestGeweke(3)
```

4.5 Summary

In this chapter, we present the most popular methods for obtaining posterior draws when the posterior distribution does not have a standard closed-form solution. In particular, Markov chain Monte Carlo (MCMC) methods, such as Gibbs sampling and the Metropolis-Hastings algorithm, are the most commonly used approaches in this book. However, Hamiltonian Monte Carlo is gaining particular relevance in high-dimensional problems, while particle filtering (Sequential Monte Carlo) is widely applied in time series models.

Each problem requires careful consideration to determine the most appropriate method, and in many cases, a combination of methods is necessary. For instance, estimating fixed parameters in *state-space* models typically requires MCMC methods, while recursion of the state vector requires particle filtering. Additionally, convergence diagnostics are crucial because MCMC methods rely on technical assumptions that must be verified.

4.6 Exercises

1. **Example: The normal model with independent priors**

   Let's recap the math test exercise in Chapter 3, this time assuming independent priors. Specifically, let $Y_i \sim N(\mu, \sigma^2)$, where $\mu \sim N(\mu_0, \sigma_0^2)$ and $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$. The sample size is 50, and the mean and standard deviation of the math scores are 102 and 10, respectively. We set $\mu_0 = 100$, $\sigma_0^2 = 100$, and $\alpha_0 = \delta_0 = 0.001$.

   - Find the posterior distribution of μ and σ^2.
   - Program a Gibbs sampler algorithm and plot the histogram of the posterior draws of μ

2. Show that the Gibbs sampler is a particular case of the Metropolis-Hastings where the acceptance probability is equal to 1.

3. Implement a Metropolis-Hastings to sample from the Cauchy distribution, $C(0, 1)$, using as proposals a standard normal distribution and a Student's t distribution with 5 degrees of freedom.

4. This exercise was proposed by Professor Hedibert Freitas Lopes, who cites [396] as a useful reference for an introduction to Hamiltonian Monte Carlo in **R** and the *hmclearn* package. The task is to obtain posterior draws using the Metropolis-Hastings and Hamiltonian Monte Carlo algorithms for the posterior distribution given

by

$$\pi(\theta_1, \theta_2 \mid \boldsymbol{y}) \propto \exp\left\{-\frac{1}{2}(\theta_1^2\theta_2^2 + \theta_1^2 + \theta_2^2 - 8\theta_1 - 8\theta_2)\right\}.$$

5. **Ph.D. students sleeping hours continues**

   (a) Use importance sampling based on a $U(0,1)$ proposal to obtain draws of $\boldsymbol{\theta} \mid \boldsymbol{y} \sim B(16.55, 39.57)$ in the Ph.D. students' sleeping hours example in Chapter 3. Note that, based on Exercise 15 in Chapter 3, $\alpha_0 = 1.44$ and $\beta_0 = 2.57$.

   (b) Compute the marginal likelihood in this context (Bernoulli-Beta model) and compare it to the result obtained using the Gelfand-Dey method.

6. Example 4.1 in [170] is

$$\theta_t = 0.5\theta_{t-1} + 25\frac{\theta_{t-1}}{1+\theta_{t-1}^2} + 8cos(1.2t) + w_t$$
$$y_t = \frac{\theta_t^2}{20} + \mu_t,$$

   where $\theta_0 \sim N(0, \sqrt{10})$, $w_t \sim \mathcal{N}(0, \sqrt{10})$ and $\mu_t \sim N(0, \sqrt{1})$.

   - Perform sequential importance sampling in this example.
   - Perform particle (Bootstrap) filtering in this example.
   - Estimate the marginal likelihood in this example.

7. **Ph.D. students sleeping hours continues**

   - Perform the diagnostics of Section 4.4 in this example.
   - Check if there are errors in the posterior simulator of the Metropolis-Hastings algorithm in this example using the Geweke approach using as test functions the first moments of p and p^2. Remember from Exercise 15 in Chapter 3 that the sample size is 52, and $\alpha_0 = 1.22$ and $\beta_0 = 2.57$.
   - Run the Geweke test using $\alpha_0 = 2.57$ and $\beta_0 = 1.22$, and check the results.

Part II

Regression models: A GUIded toolkit

5

Graphical user interface

This chapter introduces our graphical user interface (GUI) for conducting Bayesian regression analysis in a user-friendly environment, requiring no programming skills (drag and drop). Our GUI is built as an interactive web application using *shiny* [73] and incorporates packages such as *MCMCpack* [272] and *bayesm* [348] from the **R** software [319]. It is designed for teaching and applied purposes at an introductory level. In the following chapters of the second part of this book, we present several applications that demonstrate the potential of our GUI for applied researchers and practitioners.

5.1 Introduction

Our GUI enables users to perform inference using Bayesian regression analysis without requiring programming skills. The latter is often a significant impediment to the widespread adoption of the Bayesian framework [419, 223].

Several other graphical user interfaces are available for Bayesian regression analysis. *ShinyStan* [388] is a highly flexible, open-source program; however, users must have some programming skills. It is based on the *Stan* software for Bayesian data analysis [65]. *BugsXLA* [419] is also open source but less flexible, though it does not require programming skills. *Bayesian Regression: Nonparametric and Parametric Models* [223] is a user-friendly and flexible GUI based on the *MATLAB Compiler* for 64-bit Windows systems. It primarily focuses on Bayesian nonparametric regression and is designed for users already familiar with basic parametric models, such as those implemented in our GUI. Additionally, there are tools such as the *MATLAB Toolkit*, *Stata*, and *BayES*, but these are not open source.

We developed our GUI as an interactive web application using *shiny* [73] and various libraries in **R** [318]. The specific libraries and commands used in our GUI are listed in Table 14.1. It includes ten univariate models, four multivariate models, four time series models, three hierarchical longitudinal models, and seven Bayesian model averaging frameworks. Additionally, it provides basic summaries and diagnostics of the posterior chains, as well as visualizations such as trace plots, autocorrelation plots, and density plots.

In terms of flexibility and functionality, our GUI falls between *ShinyS-*

tan and *BugsXLA*: users do not need programming skills, but it is not as advanced as the software in [223]. However, our GUI runs on any operating system. We call our GUI BEsmarter,[1] and it is freely available at **https://github.com/besmarter/BSTApp**, where users can access all source code and datasets.

Simulated and applied datasets are stored in the **DataSim** and **DataApp** folders of our **GitHub** repository (see Tables 14.2 and 14.3 for details). The **DataSim** folder includes the files used to simulate different processes, providing access to population parameters. As a result, these files serve as a valuable pedagogical tool for illustrating statistical properties of the inferential frameworks available in our GUI. The **DataApp** folder contains the datasets used in this book, which users can use as templates for structuring their own datasets.

There are three ways to install our GUI. The easiest method, which requires installing **R** and potentially an **R** code editor, is to type:

R code. How to display our graphical user interface

```
shiny::runGitHub("besmarter/BSTApp", launch.browser = T)
```

in the **R** package console or any **R** code editor, we strongly recommend typing this command directly rather than copying and pasting it, as quotation marks may cause potential issues.

The second option is to visit **https://posit.cloud/content/4328505**, log in or sign up for **Posit Cloud**, and access the project titled **GUIded Bayesian Regression App BSTApp**. In the bottom-right window, navigate to the **BSTApp-master** folder under **Files**, open the **app.R** file, and click the **Run App** button. However, prolonged inactivity may cause the session to close.

The third approach, and our recommendation, is using a **Docker** image by running:

1. docker pull magralo95/besmartergui:latest
2. docker run –rm -p 3838:3838 magralo95/besmartergui

in your **Command Prompt**, this command creates an isolated environment for our GUI, ensuring consistent performance across different systems. Note that **Docker** must be installed to deploy our GUI using this method. Users can then access the app by navigating to *127.0.0.1:3838* or *http://localhost:3838/*.

[1] Bayesian Econometrics: Simulations, Models, and Applications to Research, Teaching, and Encoding with Responsibility.

FIGURE 5.1
Display of graphical user interface.

FIGURE 5.2
Univariate models: Specification.

After using any of the three methods to run our GUI, users will see a new window displaying a presentation of our research team (see Figure 5.1). Additionally, the top panel in Figure 5.1 shows the categories of models that can be estimated in our GUI.

5.2 Univariate models

After deploying our GUI (see Figure 5.1), the user should select *Univariate Models* from the top panel. Then, Figure 5.2 is displayed, showing a radio button on the left-hand side that lists the specific models within this category. In particular, users can see that the normal model is selected from the univariate models class.

FIGURE 5.3
Univariate models: Results.

Then, the right-hand panel displays a widget for uploading the input dataset, which must be a *csv* file with headers in the first row. Users must also select the separator type used in the input file: comma, semicolon, or tab (use the **DataSim** and **DataApp** folders for input file templates). Once the dataset is uploaded, users can preview the data. Range sliders allow users to set the number of iterations for the Markov Chain Monte Carlo algorithm, specify the burn-in period, and adjust the thinning parameter (see the following chapters in this section for technical details).

Next, users must specify the equation. This can be done using the formula builder, where they select the dependent variable and independent variables, then click on the *Build Formula* tab. The equation appears in the *Main Equation* space, formatted according to **R** syntax (see the main equation box in Figure 5.2, e.g., $y \sim x1 + x2 + x3$). Users can modify this as needed, including higher-order terms, interaction effects, or other transformations. These modifications must follow the standard formula syntax.[2]

By default, univariate models include an intercept, except for ordered probit models, where the specification must explicitly exclude it due to *identification* constraints (see details below).[3] Thus, users should specify this explicitly as follows: $y \sim x1 + x2 + x3 - 1$.

Finally, users must define the prior hyperparameters. For example, in the normal-inverse gamma model, these include the mean vector, covariance matrix, shape parameter, and scale parameter (see Figure 5.3). However, our GUI uses *non-informative* hyperparameters by default across all modeling frameworks, so this step is optional.

After completing the specification process, users should click the *Go!* button to initiate the estimation. Once the process is finished, our GUI displays

[2]See **https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/formula**
[3]An *identification* issue arises when multiple sets of model parameters yield the same likelihood function value.

the summary statistics and convergence diagnostics (see Figure 5.3). Additionally, widgets allow users to download the posterior chains (*csv* file) and graphs (*pdf* and *eps* files). Note that in the results—summary, posterior chains, and graphs—the coefficients are ordered with location parameters appearing first, followed by scale parameters.

For multinomial models (probit and logit), the dataset must be structured as follows: the first column should contain the dependent variable, followed by alternative-specific regressors (e.g., alternatives' prices), and finally, non-alternative-specific regressors (e.g., income). The formula builder allows users to specify the dependent variable as well as both types of independent variables (see technical details in the next chapter). Additionally, users must define the base category, the number of alternatives (which is also required for ordered probit), the number of alternative-specific regressors, and the number of non-alternative-specific regressors (see Figure 5.4).

For multinomial logit models, users can also specify a tuning parameter—the degrees of freedom for the Metropolis–Hastings algorithm (see technical details in the next chapter). This tuning option is available in our GUI when estimation relies on the Metropolis–Hastings algorithm.

In the results of these models, coefficients are ordered as follows:

1. Intercepts (cte_l in the summary display, where l represents the alternative).
2. Non-alternative-specific regressors (NAS_{jl} in the summary display, where l represents the alternative and j the non-alternative regressor).
3. Alternative-specific regressors (AS_j in the summary display, where j represents the alternative-specific regressor).

Note that the non-alternative-specific regressors associated with the base category are set to zero and do not appear in the results. Additionally, due to identification constraints in multinomial and multivariate probit models, some coefficients in the main diagonal of the covariance matrix remain constant.

For the negative binomial model, users must specify a dispersion parameter (see the next chapter for details). Similarly, for Tobit and quantile models, users need to define the censorship points and quantiles, respectively.

The Bayesian bootstrap method only requires uploading a dataset, specifying the number of MCMC iterations, setting the resampling size, and defining the equation (see Figure 5.5). The input file should follow the same structure as the one used for the univariate normal model.

FIGURE 5.4
Univariate models: Multinomial.

FIGURE 5.5
Univariate models: Bootstrap.

FIGURE 5.6
Multivariate models: Simple multivariate.

5.3 Multivariate models

After our GUI is deployed (see Figure 5.1), the user should select *Multivariate Models* from the top panel. Figure 5.6 will then be displayed, showing a radio button on the left-hand side that lists the specific models within this category.

Figure 5.6 illustrates the multivariate regression setup. The input file should first contain the dependent variables, followed by the regressors. If each equation includes an intercept, a column of 1s should be added after the dependent variables in the input file. Users can preview the data after uploading the file.

The user must specify the number of dependent variables and regressors, indicate whether an intercept should be included, and define the hyperparameter values (see Figure 5.6).

In seemingly unrelated regressions, the input file should first contain the dependent variables, followed by the regressors for each equation, including the intercept (a column of 1s) if necessary. Users must define the number of dependent variables (equations), the total number of regressors (the sum of all regressors associated with the equations), and the number of regressors per equation (including the intercept if necessary). Users can also specify the values of the hyperparameters if prior information is available.

The results of the simple multivariate and seemingly unrelated regressions first display the posterior location parameters by equation, followed by the posterior covariance matrix.

In the instrumental variable setting, users should specify the main equation and the instrumental equation. This setting includes intercepts by default. The first variable on the right-hand side of the main equation must be the variable with endogeneity issues. In the instrumental equation, the dependent variable is the one with endogeneity issues, modeled as a function of the in-

FIGURE 5.7
Multivariate models: Multivariate probit.

struments. Users can also specify the values of the hyperparameters if they have prior information. The input file should include the dependent variable, the endogenous regressor, the instruments, and the exogenous regressors. The results first list the posterior estimates of the endogenous regressor, followed by the location parameters of the auxiliary regression (instrumental equation), the location parameters of the exogenous regressors, and finally, the posterior covariance matrix.

The multivariate probit model requires the input dataset to be ordered by unit. For example, three choices imply repeating each unit three times. The first column must contain the identification for each unit, using ordered integers. Next, the dependent variable should be a single vector of 0s and 1s, followed by the regressors, which must include a column of 1s for the intercepts. Users should specify the number of units, the number of regressors, and the number of choices (see Figure 5.7). The results first display the posterior location parameters by equation, followed by the posterior covariance matrix.

5.4 Time series models

After our GUI is deployed (see Figure 5.8), the user should select *Time Series Models* from the top panel. Then, Figure 5.8 will be displayed, and the user will see the radio button on the left-hand side, which shows the specific models within this general class.

Users can perform inference using dynamic linear models (DLM), autoregressive moving average (ARMA) models, stochastic volatility models (SVM), and vector autoregressive (VAR) models. Users should upload a dataset, which must be a *csv* file with headers in the first row. The files for DLMs and SVMs

FIGURE 5.8
Time series models.

have the same structure: the first column contains the dependent variable, followed by the independent variables. For ARMA models, there is only one column with the modeled variable, while VAR models have each modeled variable in a separate column. Note that this version of the GUI does not allow for exogenous variables in VAR models. Users should specify the separator used in the input file: comma, semicolon, or tab. A dataset preview is displayed once the file is uploaded. Dataset templates can be found in the folders **DataSim** (see Table 14.2 for details) and **DataApp** (see Table 14.3 for details) in our *GitHub* repository.

Next, users should set the MCMC and burn-in iterations using the range sliders and the thinning parameter using the input box.

To estimate DLMs, users should set the hyperparameters for the precision of the observation equation and the state equations (means and variances) if prior information is available. Otherwise, users can click the *Pre Calculate Prior* button, where these hyperparameters are estimated based on a recursive model estimation using ordinary least squares (OLS). The sample size is progressively increased, and the location parameters are saved. The GUI then computes the covariance matrix of this sequence and uses it to set the prior mean for the precision of the state vector, which is equal to the inverse of the maximum element on the main diagonal of the covariance matrix ($a.theta$). The prior variance is set to ten times this value ($b.theta$). For the observation equation, the prior mean of the precision is set to the inverse of the OLS variance estimate ($a.y$), and the prior variance is set to ten times this value ($b.y$). This is a rudimentary approach to setting these hyperparameters, and users are encouraged to use a more thoughtful process.

Next, users should click the *Go!* button to start estimating the model. This may take a few minutes, as DLMs are complex to estimate. Users should be patient. Once the estimation is complete, the GUI will display graphs of the states (mean and 95% credible intervals), summary statistics of the posterior chains for the observation and state variances, and convergence diagnostics.

FIGURE 5.9
Time series models: ARMA specification

Users can download the mean and the lower and upper limits of the 95% credible intervals of the states, as well as the posterior chains for the variances.

For ARMA models, users need to set the frequency (annual -1-, quarterly -4-, and monthly -12-), as well as the AR and MA orders (see Figure 5.9). Then, users should set the location and scale hyperparameters for the intercept, autoregressive (AR), moving average (MA), and standard deviation terms. Note that there is only one set of hyperparameters for the AR and MA coefficients. This step is optional, as the GUI uses non-informative priors by default.

Then, users should click the *Go!* button, and the GUI will start estimating the model. The GUI will display the summary statistics of the posterior draws and the convergence diagnostics. The order is AR coefficients (if any), MA coefficients (if any), intercept, and standard deviation. Users can download the posterior chains and figures (density, autocorrelation, and trace plots).

Estimation of the SVMs requires setting the coefficients of the mean and standard deviation of the Gaussian prior for the regression parameters, the mean and standard deviation for the Gaussian prior distribution of the level of the log-volatility, shape parameters for the Beta prior distribution of the transformed persistence parameter, and a positive real number representing the scaling of the transformed volatility of log-volatility. However, this step is not necessary, as by default our GUI uses the default values in the *stochvol* package.

Then, click the *Go!* button, wait for the estimation to be completed, and the GUI will display the stochastic volatility plot (mean and 95% credible interval). Users can also view the summary and diagnostics of the posterior chains (see Figure 5.10). In addition, users can download the mean and the lower and upper limits of the 95% credible intervals of the stochastic volatility, as well as the posterior chains of the variances.

To estimate VAR models, users should set the number of lags, the impulse

FIGURE 5.10
Time series models: Stochastic volatility results.

response and forecast periods, the three coefficients of the Minnesota prior, and the type of impulse response (*forecast error impulse response* -feir- or *orthogonalized impulse response* -other-, both cumulative or non-cumulative). See Chapter 8 for details, Section 8.4.

Click the *Go!* button, and after a few minutes, users will be able to see the plots of the impulse responses and forecasts (means and 95% credible intervals). Click the *Download Results* button, and a zip file with *.csv* files containing the impulse responses and forecasts, along with their plots, will be downloaded.

5.5 Longitudinal/panel models

After our GUI is deployed (see Figure 5.1), the user should select *Hierarchical Longitudinal Models* in the top panel. Then, Figure 5.11 will be displayed, and the user can see the radio button on the left-hand side that shows the specific models inside this generic class.

The hierarchical longitudinal models tab allows for estimating models that account for within-subject correlation when the dependent variable is continuous (Normal), binary (Logit), or a count (Poisson).

The input files for hierarchical longitudinal models should first include the dependent variable, followed by the regressors and a cross-sectional identifier ($i = 1, 2, \ldots, N$). It is not a requirement to have a balanced dataset: T_i can be different for each i (see Chapter 9 for technical details). Users can see templates of datasets in the folders **DataSim** (see Table 14.2 for details) and **DataApp** (see Table 14.3 for details) in our *GitHub* repository. When the dataset is uploaded, users will have a preview of it.

FIGURE 5.11
Hierarchical longitudinal models: Specification.

Users should also specify the fixed part equation and the random part equation, both in **R** format. If only random intercepts are required, do not enter anything in the latter part (see Figure 5.11). Users should also type the name of the cross-sectional identifier variable. The results displayed and the posterior graphs are associated with the fixed effects and covariance matrix. However, users can download the posterior chains of all posterior estimates: fixed and random effects, and the covariance matrix.

5.6 Bayesian model average

After our GUI is deployed (see Figure 5.1), the user should select *Bayesian Model Averaging* in the top panel. Then, Figure 5.12 will be displayed, and the user can see the radio button on the left-hand side that shows the specific models inside this generic class.

Bayesian model averaging (BMA) based on a Gaussian distribution can be carried out using the Bayesian information criterion (BIC) approximation, Markov chain Monte Carlo model composition (MC3), instrumental variables (see Figure 5.12), and dynamic BMA. The first two approaches require an input dataset where the first column is the dependent variable, followed by the potentially important regressors.

Users should set the bandwidth model selection parameter (O_R) and the number of iterations for BIC and MC3, respectively (see Chapter 10 for technical details). The results include the posterior inclusion probability ($p \neq 0$), expected value (EV), and standard deviation (SD) of the coefficients associated with each regressor. The BIC framework also displays the most relevant models with their posterior model probabilities (PMP). Users can download

FIGURE 5.12
Bayesian model averaging: Specification and results.

FIGURE 5.13
Bayesian model averaging: Instrumental variable specification.

two *csv* files: *Best models* and *Descriptive statistics coefficients*. The former is a 0-1 matrix such that the columns are the regressors and the rows are the models; a 1 indicates the presence of a specific regressor in a specific model, and 0 indicates its absence. Note that the last column of this file is the posterior model probability for each model (row). The latter file shows the posterior inclusion probabilities, expected values, and standard deviations associated with each regressor, taking into account the BMA procedure based on the best models.

Bayesian model averaging with endogeneity issues requires two input files. The first file should have the dependent variable in the first column, followed by the regressors with endogeneity issues, and then the exogenous regressors. The user should include a column of 1's if an intercept is required. The second input file contains all the instruments. Users should also specify the number of regressors with endogeneity issues (see Figure 5.13).

The results include the posterior inclusion probabilities and expected val-

ues for each regressor. The user can find the results of the main equation, and then of the auxiliary equations. Users can download *csv* files of BMA results for both the second stage (main equation) and the first stage (auxiliary equations). In addition, users can download the posterior chains of the location parameters of the main equation, β_l, $l = 1, 2, \ldots, dim\{\boldsymbol{\beta}\}$, the location parameters of the auxiliary equations, $\gamma_{j,i}$, $j = 1, 2, \ldots, dim\{\boldsymbol{\beta}_s\}$ where $dim\{\boldsymbol{\beta}_s\}$ is the number of regressors with endogeneity issues, $i = 1, 2, \ldots, dim\{\boldsymbol{\gamma}\}$, where $dim\{\boldsymbol{\gamma}\}$ is the number of regressors in the auxiliary regressors (exogeneous regressors + instruments), and the elements of the covariance matrix $\sigma_{j,k}$ (see Chapter 10 for technical details).

Dynamic BMA also requires two files. The first is the dataset with the dependent variable and potential regressors, and the second file describes the competing models. There is one column for each regressor and one row for each competing model; 0 indicates that the regressor is not in the model, and 1 indicates that it is in the model. Users can see templates of this file in the folders **DataSim** (see Table 14.2 for details) and **DataApp** (see Table 14.3 for details) of our *GitHub* repository.

Then, the users should set the *forgetting parameters* of the covariance and transition matrices and click the *Go!* button. A plot of the PMPs of the competing models is displayed, and users can click the *Download the results for DBMA*. Two files are downloaded: the first contains the dynamic Bayesian average filtering recursions for each state, and the second contains the PMP of each model and the dynamic Bayesian model averaging prediction.

Bayesian model averaging based on BIC approximation for non-linear models (Logit, Gamma, and Poisson) requires an input dataset where the first column is the dependent variable, and the other columns are the potentially relevant regressors. Users should specify the bandwidth model selection parameters, also referred to as Occam's window parameters (O_R and O_L). Our GUI displays the posterior inclusion probabilities ($p! = 0$), the expected value of the posterior coefficients (EV), and the standard deviation (SD). In addition, users can view the results associated with the models with the highest posterior model probabilities and download *csv* files with the results of specifications of the best models and descriptive statistics of the posterior coefficients from the BMA procedure. These files are similar to the results of the BIC approximation for the Gaussian model.

5.7 Help

The last tab in our GUI is *Help*. There, you can find the *PDF* version of this book and the link to the *HTML* online version. Users can also send me an email at *aramir21@gmail.com* for any questions, comments, or suggestions.

5.8 Warning

Users should also note that sometimes our GUI shuts down. In our experience, this is due to computational issues arising from the implicit commands we call when estimating certain models. These issues may include computationally singular systems, missing values where TRUE/FALSE are needed, L-BFGS-B requiring finite values for "fn", NA/NaN/Inf values, or errors in `backsolve`. These issues can sometimes be resolved by adjusting the dataset, such as avoiding high levels of multicollinearity.

It should also be noted that when warning messages are displayed in our GUI, there is a high likelihood of convergence issues with the posterior chains. Therefore, the results may not be trustworthy. Users can identify these problems by checking the console in their *RStudio* sections, where the specific folder/file where the issue occurred will be specified. In any case, we would appreciate your feedback to improve and enhance our GUI.

We should also mention that there are many ways to improve the codes presented in this book, and particularly, the following five chapters. For instance, the *MCMCpack* and *bayesm* packages perform most of the matrix operations in C++ using the *Rcpp* package. This substantially speeds up the algorithms compared to the codes presented in the next chapters when we program from scratch the samplers. We could further improve the computational times of our codes using parallel computing and the *Rcpp* package, but this requires more advanced skills that are not covered in this book.

6

Univariate models

We describe how to perform Bayesian inference in some of the most common univariate models: normal-inverse gamma, logit, probit, multinomial probit and logit, ordered probit, negative binomial, tobit, quantile regression, and Bayesian bootstrap in linear models. The point of departure is assuming a random sample of cross-sectional units. We then show the posterior distributions of the parameters and some applications. In addition, we show how to perform inference in various models using three levels of programming skills: our graphical user interface (GUI), packages from **R**, and programming the posterior distributions. The first requires no programming skills, the second requires an intermediate level, and the third demands more advanced skills. We also include mathematical and computational exercises.

We can run our GUI typing

R code. How to display our graphical user interface

```
shiny::runGitHub("besmarter/BSTApp", launch.browser = T)
```

in the **R** package console or any **R** code editor. However, users should see Chapter 5 for details.

6.1 The Gaussian linear model

The Gaussian linear model specifies $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\mu}$ such that $\boldsymbol{\mu} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_N)$ is an stochastic error, $\boldsymbol{X}$ is a $N \times K$ matrix of regressors, $\boldsymbol{\beta}$ is a K-dimensional vector of location coefficients, σ^2 is the variance of the model (scale parameter), $\boldsymbol{y}$ is a N-dimensional vector of a dependent variable, and N is the sample size. We describe this model using the conjugate family in Section 3.3, that is,

$\pi(\boldsymbol{\beta}, \sigma^2) = \pi(\boldsymbol{\beta} \mid \sigma^2) \times \pi(\sigma^2)$, and this allowed to get the posterior marginal distribution for $\boldsymbol{\beta}$ and σ^2.

We assume independent prior in this section, that is, $\pi(\boldsymbol{\beta}, \sigma^2) = \pi(\boldsymbol{\beta}) \times \pi(\sigma^2)$, where $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$, $\alpha_0/2$ and $\delta_0/2$ are the shape and rate parameters. This setting allows getting the posterior conditional distributions, that is, $\pi(\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y}, \boldsymbol{X})$ and $\pi(\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{X})$, which in turn allows to use the Gibbs sampler algorithm to perform posterior inference of $\boldsymbol{\beta}$ and σ^2.

The likelihood function in this model is

$$p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X}) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\}.$$

Then, the conditional posterior distributions are

$$\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

and

$$\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{X})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{y})$, $\alpha_n = \alpha_0 + N$ and $\delta_n = \delta_0 + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$ (see Exercise 1 in this chapter).[1]

Example: The market value of soccer players in Europe

Let's analyze the determinants of the market value of soccer players in Europe. In particular, we use the dataset *1ValueFootballPlayers.csv* which is in folder **DataApp** in our github repository **https://github.com/besmarter/BSTApp**. This dataset was used by [372] to finding the determinants of high performance soccer players in the five most important national leagues in Europe.

The specification of the model is

$$\begin{aligned}\log(\text{Value}_i) = {} & \beta_1 + \beta_2\text{Perf}_i + \beta_3\text{Age}_i + \beta_4\text{Age}_i^2 + \beta_5\text{NatTeam}_i \\ & + \beta_6\text{Goals}_i + \beta_7\text{Exp}_i + \beta_8\text{Exp}_i^2 + \mu_i,\end{aligned}$$

where *Value* is the market value in Euros (2017), *Perf* is a measure of performance, *Age* is the players' age in years, *NatTem* is an indicator variable that takes the value of 1 if the player has been on the national team, *Goals* is the number of goals scored by the player during his career, and *Exp* is his experience in years.

We assume that the dependent variable distributes normal, then we use a normal-inverse gamma model using vague conjugate priors where $\boldsymbol{B}_0 = 1000\boldsymbol{I}_8$, $\boldsymbol{\beta}_0 = \boldsymbol{0}_8$, $\alpha_0 = 0.001$ and $\delta_0 = 0.001$. We perform a Gibbs sampler

[1] This model can be extended to consider heteroskedasticity such that $y_i \sim N(\boldsymbol{x}_i^\top\boldsymbol{\beta}, \sigma^2/\tau_i)$, where $\tau_i \sim G(v/2, v/2)$. See Exercise 2 for details.

with 5,000 MCMC iterations plus a burn-in equal to 5,000, and a thinning parameter equal to 1.

Once our GUI is displayed (see beginning of this chapter), we should follow Algorithm A6 to run linear Gaussian models in our GUI (see Chapter 5 for details):

Algorithm A6 Gaussian linear model

1: Select *Univariate Models* on the top panel
2: Select *Normal* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Set the hyperparameters: mean vector, covariance matrix, shape and scale parameters. This step is not necessary as by default our GUI uses non-informative priors
8: Click the *Go!* button
9: Analyze results
10: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

We can see in the following **R** code examples how to perform the linear Gaussian model using the *MCMCregress* command from the *MCMCpack* package, as well as how to program the Gibbs sampler ourselves. We should obtain similar results using all three approaches: GUI, package, and our function. In fact, our GUI relies on the *MCMCregress* command. For instance, the value of a top soccer player in Europe increases by 134% ($\exp(0.85) - 1$) on average when he has played for the national team, with a 95% credible interval of (86%, 197%).

R code. The value of soccer players, using R packages

```
rm(list = ls())
set.seed(010101)
########################## Linear regression: Value of soccer players ##########################
Data <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/1ValueFootballPlayers.csv", sep = ",", header = TRUE, quote = "")
attach(Data)
y <- log(Value)
# Value: Market value in Euros (2017) of soccer players
# Regressors quantity including intercept
X <- cbind(1, Perf, Age, Age2, NatTeam, Goals, Exp, Exp2)
# Perf: Performance. Perf2: Performance squared. Age: Age; Age: Age squared.
# NatTeam: Indicator of national team. Goals: Scored goals. Goals2: Scored goals squared
# Exp: Years of experience. Exp2: Years of experience squared. Assists: Number of assists
k <- dim(X)[2]
N <- dim(X)[1]
# Hyperparameters
d0 <- 0.001/2
a0 <- 0.001/2
b0 <- rep(0, k)
c0 <- 1000
B0 <- c0*diag(k)
B0i <- solve(B0)
# MCMC parameters
mcmc <- 5000
burnin <- 5000
tot <- mcmc + burnin
thin <- 1
# Posterior distributions using packages: MCMCpack sets the model in terms of the precision matrix
posterior  <- MCMCpack::MCMCregress(y~X-1, b0=b0, B0 = B0i, c0 = a0, d0 = d0, burnin = burnin, mcmc = mcmc, thin = thin)
summary(coda::mcmc(posterior))
Iterations = 1:5000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 5000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
          Mean       SD  Naive SE Time-series SE
X         3.695499 2.228060 3.151e-02      3.151e-02
XPerf     0.035445 0.004299 6.079e-05      6.079e-05
XAge      0.778410 0.181362 2.565e-03      2.565e-03
XAge2    -0.016617 0.003380 4.781e-05      4.781e-05
XNatTeam  0.850362 0.116861 1.653e-03      1.689e-03
XGoals    0.009097 0.001603 2.266e-05      2.266e-05
XExp      0.206208 0.062713 8.869e-04      8.428e-04
XExp2    -0.006992 0.002718 3.844e-05      3.719e-05
sigma2    0.969590 0.076091 1.076e-03      1.076e-03
```

R. code. The value of soccer players, programming our Gibbs sampler

```
# Posterior distributions programming the Gibbs sampling
# Auxiliary parameters
XtX <- t(X)%*%X
bhat <- solve(XtX)%*%t(X)%*%y
an <- a0 + N
# Gibbs sampling functions
PostSig2 <- function(Beta){
  dn <- d0 + t(y - X%*%Beta)%*%(y - X%*%Beta)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2){
  Bn <- solve(B0i + sig2^(-1)*XtX)
  bn <- Bn%*%(B0i%*%b0 + sig2^(-1)*XtX%*%bhat)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostBetas <- matrix(0, mcmc+burnin, k)
PostSigma2 <- rep(0, mcmc+burnin)
Beta <- rep(0, k)
for(s in 1:tot){
  sig2 <- PostSig2(Beta = Beta)
  PostSigma2[s] <- sig2
  Beta <- PostBeta(sig2 = sig2)
  PostBetas[s,] <- Beta
}
keep <- seq((burnin+1), tot, thin)
PosteriorBetas <- PostBetas[keep,]
colnames(PosteriorBetas) <- c("Intercept", "Perf", "Age", "
    Age2", "NatTeam", "Goals", "Exp", "Exp2")
summary(coda::mcmc(PosteriorBetas))
Iterations = 1:5000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 5000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
          Mean       SD  Naive SE Time-series SE
Intercept  3.663230 2.194363 3.103e-02      3.103e-02
Perf       0.035361 0.004315 6.102e-05      6.102e-05
Age        0.780374 0.178530 2.525e-03      2.525e-03
Age2      -0.016641 0.003332 4.713e-05      4.713e-05
NatTeam    0.850094 0.119093 1.684e-03      1.684e-03
Goals      0.009164 0.001605 2.270e-05      2.270e-05
Exp        0.205965 0.062985 8.907e-04      8.596e-04
Exp2      -0.007006 0.002731 3.862e-05      3.701e-05
PosteriorSigma2 <- PostSigma2[keep]
summary(coda::mcmc(PosteriorSigma2))
Iterations = 1:5000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 5000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean             SD       Naive SE Time-series SE
0.973309       0.077316       0.001093       0.001116
```

6.2 The logit model

In the logit model the dependent variable is binary, $y_i = \{1, 0\}$, then it follows a Bernoulli distribution, $y_i \overset{ind}{\sim} B(\pi_i)$, that is, $p(y_i = 1) = \pi_i$, such that $\pi_i = \frac{\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}}{1+\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}}$, where $\boldsymbol{x}_i$ is a K-dimensional vector of regressors.

The likelihood function of the logit model is

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{X}) &= \prod_{i=1}^{N} \pi_i^{y_i} (1-\pi_i)^{1-y_i} \\ &= \prod_{i=1}^{N} \left(\frac{\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}}{1+\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}} \right)^{y_i} \left(\frac{1}{1+\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}} \right)^{1-y_i}. \end{aligned}$$

We can specify a Normal distribution as prior, $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$. Then, the posterior distribution is

$$\begin{aligned} \pi(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) \propto \prod_{i=1}^{N} \left(\frac{\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}}{1+\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}} \right)^{y_i} \left(\frac{1}{1+\exp\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\}} \right)^{1-y_i} \\ \times \exp\left\{ -\frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_0) \right\}. \end{aligned}$$

The logit model does not have a standard posterior distribution. Therefore, a random walk Metropolis–Hastings algorithm can be used to obtain draws from the posterior distribution. A potential proposal distribution is a multivariate normal, centered at the current value, with covariance matrix $\tau^2(\boldsymbol{B}_0^{-1} + \widehat{\boldsymbol{\Sigma}}^{-1})^{-1}$, where $\tau > 0$ is a tuning parameter and $\widehat{\boldsymbol{\Sigma}}$ is the sample covariance matrix obtained from the maximum likelihood estimation [271].[2]

Observe that

$$\log(p(\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{X})) = \sum_{i=1}^{N} y_i \boldsymbol{x}_i^\top \boldsymbol{\beta} - \log(1 + \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta})).$$

We can use this expression when calculating the acceptance parameter in the computational implementation of the Metropolis-Hastings algorithm. In particular, the acceptance parameter is

$$\alpha = \min\left\{1, \exp\left(\log(p(\boldsymbol{y} \mid \boldsymbol{\beta}^c, \boldsymbol{X})) + \log(\pi(\boldsymbol{\beta}^c)) - \left(\log(p(\boldsymbol{y} \mid \boldsymbol{\beta}^{(s-1)}, \boldsymbol{X})) + \log(\pi(\boldsymbol{\beta}^{(s-1)}))\right)\right)\right\},$$

where $\boldsymbol{\beta}^c$ and $\boldsymbol{\beta}^{(s-1)}$ are the draws from the proposal distribution and the previous iteration of the Markov chain, respectively.[3]

[2] Tuning parameters should be set in a way that ensures reasonable diagnostic criteria and acceptance rates.

[3] Formulating the acceptance rate using log helps mitigate computational problems.

Example: Simulation exercise

Let's do a simulation exercise to check the performance of the algorithm. Set $\boldsymbol{\beta} = \begin{bmatrix} 0.5 & 0.8 & -1.2 \end{bmatrix}^\top$, $x_{ik} \sim N(0,1)$, $k = 2,3$ and $i = 1,2,\ldots,10000$.

We set as hyperparameters $\boldsymbol{\beta}_0 = [0\ 0\ 0]^\top$ and $\boldsymbol{B}_0 = 1000\boldsymbol{I}_3$. The tune parameter for the Metropolis-Hastings algorithm is equal to 1.

Once our GUI is displayed (see beginning of this chapter), we should follow Algorithm A7 to run logit models in our GUI (see Chapter 5 for details):

Algorithm A7 Logit model

1: Select *Univariate Models* on the top panel
2: Select *Logit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Set the hyperparameters: mean vector and covariance matrix. This step is not necessary as by default our GUI uses non-informative priors
8: Select the tuning parameter for the Metropolis-Hastings algorithm. This step is not necessary as by default our GUI sets the tuning parameter at 1.1
9: Click the *Go!* button
10: Analyze results
11: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

We can see in the following **R** code how to perform the logit model using the *MCMClogit* command from the *MCMCpack* package, as well as by programming the Metropolis-Hastings algorithm ourselves.

We should obtain similar results using the three approaches: GUI, package, and our function. Our GUI relies on the *MCMClogit* command. In particular, we achieve an acceptance rate of 0.46, and the diagnostics suggest that the posterior chains behave well. In general, the 95% credible intervals encompass the population values, and both the mean and median are very close to these values.

R. code. Simulation of the logit model estimation using R packages

```
########################## Logit: Simulation ##########################
# Simulate data
rm(list = ls())
set.seed(010101)
N <- 10000 # Sample size
B <- c(0.5, 0.8, -1.2) # Population location parameters
x2 <- rnorm(N) # Regressor
x3 <- rnorm(N) # Regressor
X <- cbind(1, x2, x3) # Regressors
XB <- X%*%B
PY <- exp(XB)/(1 + exp(XB)) # Probability of Y = 1
Y <- rbinom(N, 1, PY) # Draw Y's
table(Y) # Frequency
# write.csv(cbind(Y, x2, x3), file = "DataSimulations/LogitSim.csv") # Export data
# MCMC parameters
iter <- 5000; burnin <- 1000; thin <- 5; tune <- 1
# Hyperparameters
K <- dim(X)[2]
b0 <- rep(0, K)
c0 <- 1000
B0 <- c0*diag(K)
B0i <- solve(B0)
# Posterior distributions using packages: MCMCpack sets the model in terms of the precision matrix
RegLog <- MCMCpack::MCMClogit(Y~X-1, mcmc = iter, burnin = burnin, thin = thin, b0 = b0, B0 = B0i, tune = tune)
summary(RegLog)
Iterations = 1001:5996
Thinning interval = 5
Number of chains = 1
Sample size per chain = 1000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
      Mean      SD  Naive SE Time-series SE
X    0.4896 0.02550 0.0008064       0.001246
Xx2  0.8330 0.02730 0.0008632       0.001406
Xx3 -1.2104 0.03049 0.0009643       0.001536
2. Quantiles for each variable:
      2.5%     25%     50%     75%   97.5%
X    0.4424  0.4728  0.4894  0.5072  0.5405
Xx2  0.7787  0.8159  0.8327  0.8505  0.8852
Xx3 -1.2758 -1.2296 -1.2088 -1.1902 -1.1513
```

R. code. Simulation of the logit model estimation programming our M-H algorithm

```
# Posterior distributions programming the Metropolis-Hastings algorithm
MHfunc <- function(y, X, b0 = rep(0, dim(X)[2] + 1), B0 = 1000*diag(dim(X)[2] + 1), tau = 1, iter = 6000, burnin = 1000, thin = 5){
  Xm <- cbind(1, X) # Regressors
  K <- dim(Xm)[2] # Number of location parameters
  BETAS <- matrix(0, iter + burnin, K) # Space for posterior chains
  Reg <- glm(y ~ Xm - 1, family = binomial(link = "logit")) # Maximum likelihood estimation
  BETA <- Reg$coefficients # Maximum likelihood parameter estimates
  tot <- iter + burnin # Total iterations M-H algorithm
  COV <- vcov(Reg) # Maximum likelihood covariance matrix
  COVt <- tau^2*solve(solve(B0) + solve(COV)) # Covariance matrix for the proposal distribution
  Accep <- rep(0, tot) # Space for calculating the acceptance rate
  # Create progress bar in case that you want to see iterations progress
  pb <- winProgressBar(title = "progress bar", min = 0,
  max = tot, width = 300)
  for(it in 1:tot){
    BETAc <- BETA + MASS::mvrnorm(n = 1, mu = rep(0, K), Sigma = COVt) # Candidate location parameter
    likecand <- sum((Xm%*%BETAc) * Y - apply(Xm%*%BETAc, 1, function(x) log(1 + exp(x)))) # Log likelihood for the candidate
    likepast <- sum((Xm%*%BETA) * Y - apply((Xm%*%BETA), 1, function(x) log(1 + exp(x)))) # Log likelihood for the actual draw
    priorcand <- (-1/2)*crossprod((BETAc - b0), solve(B0))%*%(BETAc - b0) # Log prior for candidate
    priorpast <- (-1/2)*crossprod((BETA - b0), solve(B0))%*%(BETA - b0) # Log prior for actual draw
    alpha <- min(1, exp((likecand + priorcand) - (likepast + priorpast))) #Probability of selecting candidate
    u <- runif(1) # Decision rule for selecting candidate
    if(u < alpha){
      BETA <- BETAc # Changing reference for candidate if selected
      Accep[it] <- 1 # Indicator if the candidate is accepted
    }
    BETAS[it, ] <- BETA # Saving draws
    setWinProgressBar(pb, it, title=paste( round(it/tot*100, 0),
    "% done"))
  }
  close(pb)
  keep <- seq(burnin, tot, thin)
  return(list(Bs = BETAS[keep[-1], ], AceptRate = mean(Accep[keep[-1]])))
}
```

R. code. Simulation of the logit model programming our M-H algorithm, results

```
Posterior <- MHfunc(y = Y, X = cbind(x2, x3), iter = iter,
    burnin = burnin, thin = thin) # Running our M-H function
     changing some default parameters.
paste("Acceptance rate equal to", round(Posterior$AceptRate,
     2), sep = " ")
"Acceptance rate equal to 0.46"
PostPar <- coda::mcmc(Posterior$Bs)
# Names
colnames(PostPar) <- c("Cte", "x1", "x2")
# Summary posterior draws
summary(PostPar)
Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean       SD  Naive SE Time-series SE
Cte   0.4893 0.02427 0.0007674       0.001223
x1    0.8309 0.02699 0.0008536       0.001440
x2   -1.2107 0.02943 0.0009308       0.001423
2. Quantiles for each variable:
    2.5%     25%     50%     75%   97.5%
Cte   0.4431  0.4721  0.4899  0.5059  0.5344
x1    0.7817  0.8123  0.8305  0.8505  0.8833
x2   -1.2665 -1.2309 -1.2107 -1.1911 -1.1538
# Trace and density plots
plot(PostPar)
# Autocorrelation plots
coda::autocorr.plot(PostPar)
# Convergence diagnostics
coda::geweke.diag(PostPar)
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5
Cte     x1     x2
-0.975 -3.112  1.326
coda::raftery.diag(PostPar,q=0.5,r=0.05,s = 0.95)
Quantile (q) = 0.5
Accuracy (r) = +/- 0.05
Probability (s) = 0.95
Burn-in  Total Lower bound  Dependence
(M)      (N)   (Nmin)       factor (I)
Cte 6        731   385          1.90
x1  6        703   385          1.83
x2  6        725   385          1.88
coda::heidel.diag(PostPar)
Stationarity start     p-value
test         iteration
Cte passed         1       0.4436
x1  passed       101       0.3470
x2  passed         1       0.0872
Halfwidth Mean   Halfwidth
test
Cte passed     0.489 0.00240
x1  passed     0.832 0.00268
x2  passed    -1.211 0.00279
```

6.3 The probit model

The probit model also has a binary dependent variable. In this case, there is a latent variable (y_i^*, which is unobserved) that defines the structure of the estimation problem.

In particular,

$$y_i = \begin{Bmatrix} 0, & y_i^* \leq 0 \\ 1, & y_i^* > 0 \end{Bmatrix},$$

such that $y_i^* = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \mu_i$, $\mu_i \overset{i.i.d.}{\sim} N(0,1)$.[4] This implies $P(y_i = 1) = \pi_i = \Phi(\boldsymbol{x}_i^\top \boldsymbol{\beta})$, $\boldsymbol{x}_i$ is a K-dimensional vector of regressors.

[4] implemented data augmentation [393] to apply a Gibbs sampling algorithm to this model. Augmenting this model with y_i^*, we can express the likelihood contribution from observation i as:

$$p(y_i \mid y_i^*) = \mathbb{1}(y_i = 0)\mathbb{1}(y_i^* \leq 0) + \mathbb{1}(y_i = 1)\mathbb{1}(y_i^* > 0),$$

where $\mathbb{1}(A)$ is an indicator function that takes the value of 1 when the condition A is satisfied.

The posterior distribution is:

$$\pi(\boldsymbol{\beta}, \boldsymbol{y^*} \mid \boldsymbol{y}, \boldsymbol{X}) \propto \prod_{i=1}^{N} [\mathbb{1}(y_i = 0)\mathbb{1}(y_i^* \leq 0) + \mathbb{1}(y_i = 1)\mathbb{1}(y_i^* > 0)] \\ \times N_N(\boldsymbol{y^*} \mid \boldsymbol{X\beta}, \boldsymbol{I}_n) \times N_K(\boldsymbol{\beta} \mid \boldsymbol{\beta}_0, \boldsymbol{B}_0),$$

where we assume a Gaussian prior for $\boldsymbol{\beta}$: $\boldsymbol{\beta} \sim N_K(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$. This implies

$$y_i^* \mid \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{X} \sim \begin{Bmatrix} TN_{(-\infty,0]}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, 1) & , \ y_i = 0 \\ TN_{(0,\infty)}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, 1) & , \ y_i = 1 \end{Bmatrix},^{5}$$

$$\boldsymbol{\beta} \mid \boldsymbol{y}^*, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top \boldsymbol{X})^{-1}$, and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top \boldsymbol{y}^*)$.

Example: Determinants of hospitalization

We use the dataset named **2HealthMed.csv**, which is located in the **DataApp** folder of our GitHub repository **(https://github.com/besmarter/BSTApp)**, and was used by [327]. The dependent variable is a binary indicator, taking the value 1 if an individual was hospitalized in 2007, and 0 otherwise.

[4] The variance in this model is set to 1 due to identification restrictions. Observe that $P(y_i = 1 \mid \boldsymbol{x}_i) = P(y_i^* > 0 \mid \boldsymbol{x}_i) = P(\boldsymbol{x}_i^\top \boldsymbol{\beta} + \mu_i > 0 \mid \boldsymbol{x}_i) = P(\mu_i > -\boldsymbol{x}_i^\top \boldsymbol{\beta} \mid \boldsymbol{x}_i) = P(c \times \mu_i > -c \times \boldsymbol{x}_i^\top \boldsymbol{\beta} \mid \boldsymbol{x}_i)$ $\forall c > 0$. Multiplying for a positive constant does not affect the probability of $y_i = 1$.

[5] TN denotes a truncated normal density.

The specification of the model is

$$\begin{aligned}\text{Hosp}_i = \beta_1 + \beta_2\text{SHI}_i + \beta_3\text{Female}_i + \beta_4\text{Age}_i + \beta_5\text{Age}_i^2 + \beta_6\text{Est2}_i + \beta_7\text{Est3}_i \\ + \beta_8\text{Fair}_i + \beta_9\text{Good}_i + \beta_{10}\text{Excellent}_i,\end{aligned}$$

where *SHI* is a binary variable equal to 1 if the individual is enrolled in a subsidized health care program and 0 otherwise, *Female* is an indicator of gender, *Age* is in years, *Est2* and *Est3* are indicators of socioeconomic status, with *Est1* being the reference category (the lowest status), and *HealthStatus* is a self-perception of health status, where *bad* is the reference category.

We set $\boldsymbol{\beta}_0 = \mathbf{0}_{10}$, $\boldsymbol{B}_0 = \boldsymbol{I}_{10}$, with iterations, burn-in, and thinning parameters equal to 10000, 1000, and 1, respectively. We can use Algorithm A8 to run the probit model in our GUI. Our GUI relies on the *rbprobitGibbs* command from the *bayesm* package to perform inference in the probit model. The following **R** code shows how to run this example using the *rbprobitGibbs* command. We also asked you to implement a Gibbs sampler algorithm to perform inference in the probit model in the exercises.

Algorithm A8 Probit model

1: Select *Univariate Models* on the top panel
2: Select *Probit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Set the hyperparameters: mean vector and covariance matrix. This step is not necessary as by default our GUI uses non-informative priors
8: Click the *Go!* button
9: Analyze results
10: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

Our analysis finds evidence that gender and self-perceived health status significantly affect the probability of hospitalization. Women have a higher probability of being hospitalized than men, and individuals with a better perception of their health status have a lower probability of hospitalization.

R. code. Determinants of hospitalization

```
mydata <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/2HealthMed.csv", sep = ",", header = TRUE, quote = "")
attach(mydata)
str(mydata)
K <- 10 # Number of regressors
b0 <- rep(0, K) # Prio mean
B0i <- diag(K) # Prior precision (inverse of covariance)
Prior <- list(betabar = b0, A = B0i) # Prior list
y <- Hosp # Dependent variables
X <- cbind(1, SHI, Female, Age, Age2, Est2, Est3, Fair, Good, Excellent) # Regressors
Data <- list(y = y, X = X) # Data list
Mcmc <- list(R = 10000, keep = 1, nprint = 0) # MCMC parameters
RegProb <- bayesm::rbprobitGibbs(Data = Data, Prior = Prior, Mcmc = Mcmc) # Inference using bayesm package
PostPar <- coda::mcmc(RegProb$betadraw) # Posterior draws
colnames(PostPar) <- c("Cte", "SHI", "Female", "Age", "Age2", "Est2", "Est3", "Fair", "Good", "Excellent") # Names
summary(PostPar) # Posterior summary
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000
2. Quantiles for each variable:
         2.5%        25%        50%        75%      97.5%
Cte       -1.22e+00 -1.03e+00 -9.43e-01 -8.50e-01 -0.671744
SHI       -1.24e-01 -4.63e-02 -6.30e-03  3.26e-02  0.104703
Female     2.80e-02  9.65e-02  1.28e-01  1.60e-01  0.223123
Age       -7.55e-03 -2.50e-03  1.25e-04  2.80e-03  0.007646
Age2      -4.98e-05  9.05e-06  4.02e-05  7.07e-05  0.000128
Est2      -1.89e-01 -1.23e-01 -8.84e-02 -5.32e-02  0.012714
Est3      -2.13e-01 -1.03e-01 -4.73e-02  1.01e-02  0.109527
Fair      -7.09e-01 -5.69e-01 -4.93e-01 -4.16e-01 -0.269494
Good      -1.42e+00 -1.28e+00 -1.20e+00 -1.12e+00 -0.982533
Excellent -1.33e+00 -1.15e+00 -1.06e+00 -9.74e-01 -0.795881
```

6.4 The multinomial probit model

The multinomial probit model is used to model the choice of the l-th alternative over a set of L mutually exclusive options. We observe the following:

$$y_{il} = \begin{cases} 1, & \text{if } y_{il}^* \geq \max\{\boldsymbol{y}_i^*\}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{y}_i^* = \boldsymbol{X}_i\boldsymbol{\delta}+\boldsymbol{\mu}_i$, with $\boldsymbol{\mu}_i \overset{i.i.d.}{\sim} N(\boldsymbol{0}, \boldsymbol{\Sigma})$. The vector $\boldsymbol{y}_i^*$ is an unobserved latent vector of dimension L. The matrix $\boldsymbol{X}_i = \left[(1\ \boldsymbol{c}_i^\top) \otimes \boldsymbol{I}_L\ \boldsymbol{A}_i\right]$ is an $L \times j$ matrix of regressors for each alternative, where $l = 1, 2, \ldots, L$, and $j = L \times (1 + \dim(\boldsymbol{c}_i)) + a$. Here, $\boldsymbol{c}_i$ is a vector of individual-specific characteristics, $\boldsymbol{A}_i$ is an $L \times a$ matrix of alternative-varying regressors, a is the number of alternative-varying regressors, and $\boldsymbol{\delta}$ is a j-dimensional vector of parameters.

We take into account simultaneously the alternative-varying regressors (alternative attributes) and alternative-invariant regressors (individual characteristics).[6] The vector $\boldsymbol{y}_i^*$ can be stacked into a multiple regression model with correlated stochastic errors, i.e., $\boldsymbol{y}^* = \boldsymbol{X}\boldsymbol{\delta} + \boldsymbol{\mu}$, where $\boldsymbol{y}^* = \left[\boldsymbol{y}_1^{*\top}\ \boldsymbol{y}_2^{*\top}\ \ldots\ \boldsymbol{y}_N^{*\top}\right]$, $\boldsymbol{X} = \left[\boldsymbol{X}_1^\top\ \boldsymbol{X}_2^\top\ \ldots\ \boldsymbol{X}_N^\top\right]^\top$, and $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^\top\ \boldsymbol{\mu}_2^\top\ \ldots\ \boldsymbol{\mu}_N^\top\right]^\top$.

Following the practice of expressing y_{il}^* relative to y_{iL}^* by letting $\boldsymbol{w}_i = [w_{i1}\ w_{i2}\ \ldots\ w_{iL-1}]^\top$, where $w_{il} = y_{il}^* - y_{iL}^*$, we can write $\boldsymbol{w}_i = \boldsymbol{R}_i\boldsymbol{\beta} + \boldsymbol{\epsilon}_i$, with $\boldsymbol{\epsilon}_i \sim N(\boldsymbol{0}, \boldsymbol{\Omega})$, where $\boldsymbol{R}_i = \left[(1\ \boldsymbol{c}_i^\top) \otimes \boldsymbol{I}_{L-1}\ \boldsymbol{\Delta}\boldsymbol{A}_i\right]$ is an $(L-1) \times K$ matrix, with $\Delta\boldsymbol{A}_i = \boldsymbol{A}_{li} - \boldsymbol{A}_{Li}$, for $l = 1, 2, \ldots, L-1$. That is, the last row of $\boldsymbol{A}_i$ is subtracted from each row of $\boldsymbol{A}_i$, and $\boldsymbol{\beta}$ is a K-dimensional vector, where $K = (L-1) \times (1 + \dim(\boldsymbol{c}_i)) + a$.

Observe that $\boldsymbol{\beta}$ contains the same last a elements as $\boldsymbol{\delta}$, that is, the alternative-specific attribute coefficients. However, the first $(L-1) \times (1 + \dim(\boldsymbol{c}_i))$ elements of $\boldsymbol{\beta}$ are the differences $\delta_{jl} - \delta_{jL}$, for $j = 1, \ldots, \dim(\boldsymbol{c}_i)$ and $l = 1, 2, \ldots, L-1$. That is, these elements represent the difference between the coefficients of each qualitative response and the L-th alternative for the individuals' characteristics. This makes it difficult to interpret the multinomial probit coefficients.

Note that in multinomial models, for each alternative-specific attribute, it is only necessary to estimate one coefficient for all alternatives. However, for individuals' characteristics (non-alternative-specific regressors), it is required to estimate $L-1$ coefficients, since the coefficient for the base alternative is set equal to 0.

The likelihood function in this model is given by

$$p(\boldsymbol{\beta}, \boldsymbol{\Omega} \mid \boldsymbol{y}, \boldsymbol{R}) = \prod_{i=1}^{N} \prod_{l=1}^{L} p_{il}^{y_{il}},$$

where $p_{il} = p(y_{il}^* \geq \max(\boldsymbol{y}_i^*))$.

We assume independent priors for the parameters:

$$\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0) \quad \text{and} \quad \boldsymbol{\Omega}^{-1} \sim W(\alpha_0, \boldsymbol{\Sigma}_0).^7$$

[6] Note that this model is not identified if $\boldsymbol{\Sigma}$ is unrestricted. The likelihood function remains the same if a scalar random variable is added to each of the L latent regressions.

[7] The W denotes the Wishart density.

We can employ Gibbs sampling in this model, as it is a standard Bayesian linear regression model when data augmentation is used for $\boldsymbol{w}$. The posterior conditional distributions are given by

$$\boldsymbol{\beta} \mid \boldsymbol{\Omega}, \boldsymbol{w} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$
$$\boldsymbol{\Omega}^{-1} \mid \boldsymbol{\beta}, \boldsymbol{w} \sim W(\alpha_n, \boldsymbol{\Sigma}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^{*\top}\boldsymbol{X}^*)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^{*\top}\boldsymbol{w}^*)$, $\boldsymbol{\Omega}^{-1} = \boldsymbol{C}^\top\boldsymbol{C}$, $\boldsymbol{X}_i^{*\top} = \boldsymbol{C}^\top\boldsymbol{R}_i$, $\boldsymbol{w}_i^* = \boldsymbol{C}^\top\boldsymbol{w}_i$, $\boldsymbol{X}^* = \begin{bmatrix} \boldsymbol{X}_1^* \\ \boldsymbol{X}_2^* \\ \vdots \\ \boldsymbol{X}_N^* \end{bmatrix}$, $\alpha_n = \alpha_0 + N$, $\boldsymbol{\Sigma}_n = (\boldsymbol{\Sigma}_0 + \sum_{i=1}^N (\boldsymbol{w}_i - \boldsymbol{R}_i\boldsymbol{\beta})^\top(\boldsymbol{w}_i - \boldsymbol{R}_i\boldsymbol{\beta}))^{-1}$.

We can collapse the multinomial vector $\boldsymbol{y}_i$ into the indicator variable $d_i = \sum_{l=1}^{L-1} l \times \mathbb{1}(\max(\boldsymbol{w}_l) = w_{il})$.[8] Then the distribution of $\boldsymbol{w}_i \mid \boldsymbol{\beta}, \boldsymbol{\Omega}^{-1}, d_i$ is an $L-1$ dimensional Gaussian distribution truncated over the appropriate cone in $\mathcal{R}^{L-1}$. [280] propose drawing from the univariate conditional distributions $w_{il} \mid \boldsymbol{w}_{i,-l}, \boldsymbol{\beta}, \boldsymbol{\Omega}^{-1}, d_i \sim TN_{I_{il}}(m_{il}, \tau_{ll}^2)$, where

$$I_{il} = \begin{Bmatrix} w_{il} > \max(\boldsymbol{w}_{i,-l}, 0), & d_i = l \\ w_{il} < \max(\boldsymbol{w}_{i,-l}, 0), & d_i \neq l \end{Bmatrix},$$

and permuting the columns and rows of $\boldsymbol{\Omega}^{-1}$ so that the l-th column and row is the last,

$$\boldsymbol{\Omega}^{-1} = \begin{bmatrix} \boldsymbol{\Omega}_{-l,-l} & \boldsymbol{\omega}_{-l,l} \\ \boldsymbol{\omega}_{l,-1} & \omega_{l,l} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{\Omega}_{-l,-l}^{-1} + \tau_{ll}^{-2}\boldsymbol{f}_l\boldsymbol{f}_l^\top & -\boldsymbol{f}_l\tau_{ll}^{-2} \\ -\tau_{ll}^{-2}\boldsymbol{f}_l^\top & \tau_{ll}^{-2} \end{bmatrix}$$

where $\boldsymbol{f}_l = \boldsymbol{\Omega}_{-l,-l}^{-1}\boldsymbol{\omega}_{-l,l}$, $\tau_{ll}^2 = \omega_{ll} - \boldsymbol{\omega}_{l,-l}\boldsymbol{\Omega}_{-l,-1}^{-1}\boldsymbol{\omega}_{-l,l}$, $m_{il} = \boldsymbol{r}_{il}^\top\boldsymbol{\beta} + \boldsymbol{f}_l^\top(\boldsymbol{w}_{i,-l} - \boldsymbol{R}_{i,-l}\boldsymbol{\beta})$, $\boldsymbol{w}_{i,-l}$ is an $L-2$ dimensional vector of all components of $\boldsymbol{w}_i$ excluding w_{il}, $\boldsymbol{r}_{il}$ is the l-th row of $\boldsymbol{R}_i$, $l = 1, 2, \ldots, L-1$.

The identified parameters are obtained by normalizing with respect to one of the diagonal elements $\frac{1}{\omega_{1,1}^{0.5}}\boldsymbol{\beta}$ and $\frac{1}{\omega_{1,1}}\boldsymbol{\Omega}$.[9]

Warning: This model is an example where decisions must be made about setting the model in an identified parameter space versus an unidentified parameter space. The mixing properties of the posterior draws can be better in the latter case [281], which typically results in less computational burden. However, it is important to recover the identified space in a final stage. Additionally, defining priors in the unidentified space may have unintended consequences on the posterior distributions in the identified space [298]. The multinomial probit model presented in this section is set in the unidentified

[8] Observe that the identification issue in this model is due to scaling w_{il} by a positive constant does not change the value of d_i.

[9] Our GUI is based on the *bayesm* package that takes into account this identification restriction to display the outcomes of the posterior chains.

space [280], while a version of the multinomial probit in the identified space is presented by [281].

Example: Choice of fishing mode

We used in this application the dataset *3Fishing.csv* from [61, p. 491]. The dependent variable is mutually exclusive alternatives regarding fishing modes (mode), where beach is equal to 1, pier is equal to 2, private boat is equal to 3, and chartered boat (baseline alternative) is equal to 4. In this model, we have

$$\boldsymbol{X}_i = \begin{bmatrix} 1 & 0 & 0 & 0 & \text{Income}_i & 0 & 0 & 0 & \text{Price}_{i,1} & \text{Catch rate}_{i,1} \\ 0 & 1 & 0 & 0 & 0 & \text{Income}_i & 0 & 0 & \text{Price}_{i,2} & \text{Catch rate}_{i,2} \\ 0 & 0 & 1 & 0 & 0 & 0 & \text{Income}_i & 0 & \text{Price}_{i,3} & \text{Catch rate}_{i,3} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \text{Income}_i & \text{Price}_{i,4} & \text{Catch rate}_{i,4} \end{bmatrix}.$$

In this example, chartered boat is the base category, the number of choice categories is four, there are two alternative-specific regressors (price and catch rate), and one non-alternative-specific regressor (income). This setting involves the estimation of eight location parameters ($\boldsymbol{\beta}$): three intercepts, three for income, one for price, and one for catch rate. This is the order of the posterior chains in our GUI. Note that the location coefficients are set equal to 0 for the baseline category. For multinomial models, we strongly recommend using the last category as the baseline.

We also get posterior estimates for a 3×3 covariance matrix (four alternatives minus one), where the element (1,1) is equal to 1 due to identification restrictions, and elements 2 and 4 are the same, as well as 3 and 7, and 6 and 8, due to symmetry.[10] Observe that this identification restriction implies *NaN* values in [160] and [188] tests for element (1,1) of the covariance matrix, and just eight dependence factors associated with the remaining elements of the covariance matrix.

Once our GUI is displayed (see beginning of this chapter), we should follow Algorithm A9 to run multinomial probit models in our GUI (see Chapter 5 for details), which in turn uses the command *rmnpGibbs* from the *bayesm* package.

We ran 100,000 MCMC iterations plus 10,000 as burn-in with a thinning parameter equal to 5, where all priors use default values for the hyperparameters in our GUI. We found that the 95% credible intervals of the coefficient associated with income for beach and private boat alternatives are equal to (8.58e-06, 8.88e-05) and (3.36e-05, 1.45e-04). This suggests that the probability of choosing these alternatives increases compared to a chartered boat when income increases. In addition, an increase in the price or a decrease in the catch rate for specific fishing alternatives imply lower probabilities of choosing them as the 95% credible intervals are (-9.91e-03, -3.83e-03) and (1.40e-01, 4.62e-01), respectively. However, the posterior chain diagnostics suggest there are convergence issues with the posterior draws (see Exercise 5).

[10] This is the order in the pdf, eps and csv files that can be downloaded from our GUI.

Algorithm A9 Multinomial probit models

1: Select *Univariate Models* on the top panel
2: Select *Multinomial Probit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Select the number of the **Base Alternative**
7: Select the **Number of choice categorical alternatives**
8: Select the **Number of alternative specific variables**
9: Select the **Number of Non-alternative specific variables**
10: Click the *Build formula* button to generate the formula in **R** syntax.
11: Set the hyperparameters: mean vector, covariance matrix, scale matrix and degrees of freedom. This step is not necessary as by default our GUI uses non-informative priors
12: Click the *Go!* button
13: Analyze results
14: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

Results. Choice of fishing mode

```
Iterations = 10005:110000
Thinning interval = 5
Number of chains = 1
Sample size per chain = 20000
Quantiles for each variable:
         2.5%         25%         50%         75%       97.5%
cte_1   -5.83e-01 -4.08e-01 -3.22e-01 -2.37e-01 -7.93e-02
cte_2   -1.93e-01 -4.14e-02  2.16e-02  7.93e-02  1.93e-01
cte_3   -8.15e-01 -5.43e-01 -4.29e-01 -3.33e-01 -1.70e-01
NAS_1_1  8.58e-06  3.61e-05  4.95e-05  6.27e-05  8.88e-05
NAS_1_2 -3.24e-05 -7.04e-06  5.52e-06  1.93e-05  5.17e-05
NAS_1_3  3.36e-05  6.38e-05  8.08e-05  9.99e-05  1.45e-04
AS_1    -9.91e-03 -7.90e-03 -6.86e-03 -5.93e-03 -3.83e-03
AS_2     1.40e-01  2.25e-01  2.72e-01  3.28e-01  4.62e-01
```

6.5 The multinomial logit model

The multinomial logit model is used to model mutually exclusive discrete outcomes or qualitative response variables. However, this model assumes the independence of irrelevant alternatives (IIA), meaning that the choice between two alternatives does not depend on a third alternative. We consider the multinomial mixed logit model (not to be confused with the random parameters logit model), which accounts for both alternative-varying regressors (conditional) and alternative-invariant regressors (multinomial) simultaneously.[11]

In this setting, there are L mutually exclusive alternatives, and the dependent variable y_{il} is equal to 1 if the lth alternative is chosen by individual i, and 0 otherwise, where $l = \{1, 2, \ldots, L\}$. The likelihood function is

$$p(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) = \prod_{i=1}^{N} \prod_{l=1}^{L} p_{il}^{y_{il}},$$

where the probability that individual i chooses the alternative l is given by

$$p_{il} := p(y_i = l \mid \boldsymbol{\beta}, \boldsymbol{X}) = \frac{\exp\left\{\boldsymbol{x}_{il}^{\top}\boldsymbol{\beta}_l\right\}}{\sum_{j=1}^{L} \exp\left\{\boldsymbol{x}_{ij}^{\top}\boldsymbol{\beta}_j\right\}},$$

$\boldsymbol{y}$ and $\boldsymbol{X}$ are the vector and matrix of the dependent variable and regressors, respectively, and $\boldsymbol{\beta}$ is the vector containing all the coefficients.

Remember that coefficients associated with alternative-invariant regressors are set to 0 for the baseline category, and the coefficients associated with the alternative-varying regressors are the same for all the categories. In addition, we assume $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ as the prior distribution. Thus, the posterior distribution is

$$\pi(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{\beta} \mid \boldsymbol{y}, \boldsymbol{X}) \times \pi(\boldsymbol{\beta}).$$

As the multinomial logit model does not have a standard posterior distribution, [350] propose a "tailored" independent Metropolis–Hastings algorithm where the proposal distribution is a multivariate Student's t distribution with v degrees of freedom (tuning parameter), mean equal to the maximum likelihood estimator, and scale equal to the inverse of the Hessian matrix.

Example: Simulation exercise

Let's conduct a simulation exercise to evaluate the performance of the Metropolis-Hastings algorithm for inference in the multinomial logit model. We consider a scenario with three alternatives, one alternative-invariant regressor (plus the intercept), and three alternative-varying regressors. The population parameters are given by $\boldsymbol{\beta}_1 = [1 \;\; -2.5 \;\; 0.5 \;\; 0.8 \;\; -3]^{\top}$, $\boldsymbol{\beta}_2 =$

[11] The multinomial mixed logit model can be implemented as a conditional logit model.

$[1\ \ -3.5\ 0.5\ 0.8\ \ -3]^\top$, and $\boldsymbol{\beta}_3 = [0\ 0\ 0.5\ 0.8\ \ -3]^\top$, where the first two elements of each vector correspond to the intercept and the alternative-invariant regressor, while the last three elements correspond to the alternative-varying regressors. The sample size is 1000, and all regressors are simulated from standard normal distributions.

We can deploy our GUI using the command line at the beginning of this chapter. We should follow Algorithm A10 to run multinomial logit models in our GUI (see Chapter 5 for details):

Algorithm A10 Multinomial logit models

1: Select *Univariate Models* on the top panel
2: Select *Multinomial Logit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Select the **Base Alternative**
7: Select the **Number of choice categorical alternatives**
8: Select the **Number of alternative specific variables**
9: Select the **Number of Non-alternative specific variables**
10: Click the *Build formula* button to generate the formula in **R** syntax.
11: Set the hyperparameters: mean vector and covariance matrix. This step is not necessary as by default our GUI uses non-informative priors
12: Select the tuning parameter for the Metropolis-Hastings algorithm, that is, the **Degrees of freedom: Multivariate Student's t distribution**
13: Click the *Go!* button
14: Analyze results
15: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

The following code in **R** demonstrates how to implement the M-H algorithm from scratch. The first part simulates the dataset, the second part constructs the log-likelihood function, and the third part implements the M-H algorithm. We use vague priors centered at zero, with a covariance matrix of $1000\boldsymbol{I}_7$. We observe that the posterior estimates closely match the population parameters, and all 95% credible intervals contain the population parameters.

R. code. Simulation of the multinomial logit model

```
remove(list = ls())
set.seed(12345)
# Simulation of data
N<-1000  # Sample Size
B<-c(0.5,0.8,-3); B1<-c(-2.5,-3.5,0); B2<-c(1,1,0)
# Alternative specific attributes of choice 1, for instance,
    price, quality and duration of choice 1
X1<-matrix(cbind(rnorm(N,0,1),rnorm(N,0,1),rnorm(N,0,1)),N,
    length(B))
# Alternative specific attributes of choice 2, for instance,
    price, quality and duration of choice 2
X2<-matrix(cbind(rnorm(N,0,1),rnorm(N,0,1),rnorm(N,0,1)),N,
    length(B))
# Alternative specific attributes of choice 3, for instance,
    price, quality and duration of choice 3
X3<-matrix(cbind(rnorm(N,0,1),rnorm(N,0,1),rnorm(N,0,1)),N,
    length(B))
X4<-matrix(rnorm(N,1,1),N,1)
V1<-B2[1]+X1%*%B+B1[1]*X4; V2<-B2[2]+X2%*%B+B1[2]*X4; V3<-B2
    [3]+X3%*%B+B1[3]*X4
suma<-exp(V1)+exp(V2)+exp(V3)
p1<-exp(V1)/suma; p2<-exp(V2)/suma; p3<-exp(V3)/suma
p<-cbind(p1,p2,p3)
y<- apply(p,1, function(x)sample(1:3, 1, prob = x, replace =
    TRUE))
y1<-y==1; y2<-y==2; y3<-y==3
```

R. code. Simulation of the multinomial logit model

```
# Log likelihood
log.L<- function(Beta){
  V1<-Beta[1]+Beta[3]*X4+X1%*%Beta[5:7]
  V2<-Beta[2]+Beta[4]*X4+X2%*%Beta[5:7]
  V3<- X3%*%Beta[5:7]
  suma<-exp(V1)+exp(V2)+exp(V3)
  p11<-exp(V1)/suma;  p22<-exp(V2)/suma;  p33<-exp(V3)/suma
  suma2<-NULL
  for(i in 1:N){
    suma1<-y1[i]*log(p11[i])+y2[i]*log(p22[i])+y3[i]*log(p33[i])
    suma2<-c(suma2,suma1)}
  logL<-sum(suma2)
  return(-logL)
}
# Parameters: Proposal
k <- 7
res.optim<-optim(rep(0, k), log.L, method="BFGS", hessian=TRUE)
MeanT <- res.optim$par
ScaleT <- as.matrix(Matrix::forceSymmetric(solve(res.optim$hessian))) # Force this matrix to be symmetric
# Hyperparameters: Priors
B0 <- 1000*diag(k); b0 <- rep(0, k)
MHfunction <- function(iter, tuning){
  Beta <- rep(0, k); Acept <- NULL
  BetasPost <- matrix(NA, iter, k)
  pb <- winProgressBar(title = "progress bar", min = 0, max = iter, width = 300)
  for(s in 1:iter){
    LogPostBeta <- -log.L(Beta) + mvtnorm::dmvnorm(Beta, mean = b0, sigma = B0, log = TRUE)
    BetaC <- c(LaplacesDemon::rmvt(n=1, mu = MeanT, S = ScaleT, df = tuning))
    LogPostBetaC <- -log.L(BetaC) + mvtnorm::dmvnorm(BetaC, mean = b0, sigma = B0, log = TRUE)
    alpha <- min(exp((LogPostBetaC-mvtnorm::dmvt(BetaC, delta = MeanT, sigma = ScaleT, df = tuning, log = TRUE))-(LogPostBeta-mvtnorm::dmvt(Beta, delta = MeanT, sigma = ScaleT, df = tuning, log = TRUE))) ,1)
    u <- runif(1)
    if(u <= alpha){
      Acepti <- 1; Beta <- BetaC
    }else{
      Acepti <- 0; Beta <- Beta
    }
    BetasPost[s, ] <- Beta; Acept <- c(Acept, Acepti)
    setWinProgressBar(pb, s, title=paste( round(s/iter*100, 0),"% done"))
  }
  close(pb); AcepRate <- mean(Acept)
  Results <- list(AcepRate = AcepRate, BetasPost = BetasPost)
  return(Results)
}
```

R. code. Simulation of the multinomial logit model

```
# MCMC parameters
mcmc <- 10000; burnin <- 1000; thin <- 5; iter <- mcmc +
    burnin; keep <- seq(burnin, iter, thin); tuning <- 6 #
    Degrees of freedom
ResultsPost <- MHfunction(iter = iter, tuning = tuning)
summary(coda::mcmc(ResultsPost$BetasPost[keep[-1], ]))
Iterations = 1:2000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 2000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
      Mean      SD Naive SE Time-series SE
[1,]  0.9711 0.20162 0.004508       0.004508
[2,]  0.9742 0.20934 0.004681       0.004681
[3,] -2.4350 0.18950 0.004237       0.004137
[4,] -3.4195 0.24656 0.005513       0.005513
[5,]  0.5253 0.07396 0.001654       0.001654
[6,]  0.8061 0.08007 0.001790       0.001790
[7,] -3.0853 0.17689 0.003955       0.003955
2. Quantiles for each variable:
      2.5%     25%     50%     75%   97.5%
var1  0.5862  0.8367  0.9650  1.1017  1.3683
var2  0.5679  0.8310  0.9681  1.1151  1.3761
var3 -2.8239 -2.5607 -2.4291 -2.3050 -2.0812
var4 -3.9176 -3.5806 -3.4074 -3.2496 -2.9423
var5  0.3840  0.4761  0.5250  0.5759  0.6647
var6  0.6555  0.7494  0.8064  0.8616  0.9604
var7 -3.4476 -3.1991 -3.0777 -2.9641 -2.7500
```

6.6 Ordered probit model

The ordered probit model is used when there is a natural order in the categorical response variable. In this case, there is a latent variable $y_i^* = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \mu_i$, where $\boldsymbol{x}_i$ is a K-dimensional vector of regressors, $\mu_i \overset{i.i.d.}{\sim} N(0,1)$, such that $y_i = l$ if and only if $\alpha_{l-1} < y_i^* \leq \alpha_l$, for $l \in \{1, 2, \ldots, L\}$, where $\alpha_0 = -\infty$,

$\alpha_1 = 0$, and $\alpha_L = \infty$.[12] Then, the probability of observing $y_i = l$ is given by:

$$p(y_i = l) = \Phi(\alpha_l - \boldsymbol{x}_i^\top \boldsymbol{\beta}) - \Phi(\alpha_{l-1} - \boldsymbol{x}_i^\top \boldsymbol{\beta}),$$

and the likelihood function is:

$$p(\boldsymbol{\beta}, \boldsymbol{\alpha} \mid \boldsymbol{y}, \boldsymbol{X}) = \prod_{i=1}^{N} p(y_i = l \mid \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{X}).$$

The priors in this model are independent, i.e., $\pi(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \pi(\boldsymbol{\beta}) \times \pi(\boldsymbol{\gamma})$, where $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\boldsymbol{\gamma} \sim N(\boldsymbol{\gamma}_0, \boldsymbol{\Gamma}_0)$, and $\boldsymbol{\gamma} = [\gamma_2 \ \gamma_3 \ \ldots \ \gamma_{L-1}]^\top$, such that:

$$\boldsymbol{\alpha} = \left[\exp\{\gamma_2\} \ \sum_{l=2}^{3} \exp\{\gamma_l\} \ \ldots \ \sum_{l=2}^{L-1} \exp\{\gamma_l\} \right]^\top.$$

This structure imposes the ordinal condition on the cut-offs.

This model does not have a standard conditional posterior distribution for $\boldsymbol{\gamma}$ ($\boldsymbol{\alpha}$), but it does have a standard conditional distribution for $\boldsymbol{\beta}$ once data augmentation is used. We can then use a Metropolis-within-Gibbs sampling algorithm. In particular, we use Gibbs sampling to draw $\boldsymbol{\beta}$ and $\boldsymbol{y}^*$, where:

$$\boldsymbol{\beta} \mid \boldsymbol{y}^*, \boldsymbol{\alpha}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

with $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top \boldsymbol{X})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top \boldsymbol{y}^*)$, and:

$$y_i^* \mid \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{y}, \boldsymbol{X} \sim TN_{(\alpha_{y_i-1}, \alpha_{y_i})}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, 1).$$

We use a random-walk Metropolis–Hastings algorithm for $\boldsymbol{\gamma}$ with a proposal distribution that is Gaussian with mean equal to the current value and covariance matrix $s^2(\boldsymbol{\Gamma}_0^{-1} + \hat{\boldsymbol{\Sigma}}_\gamma^{-1})^{-1}$, where $s > 0$ is a tuning parameter and $\hat{\boldsymbol{\Sigma}}_\gamma$ is the sample covariance matrix associated with γ from the maximum likelihood estimation.

Example: Determinants of preventive health care visits

We used the file named *2HealthMed.csv* in this application. In particular, the dependent variable is *MedVisPrevOr*, which is an ordered variable equal to 1 if the individual did not visit a physician for preventive reasons, 2 if the individual visited once in that year, and so on, until it is equal to 6 for visiting five or more times. The latter category represents 1.6% of the sample. Observe that the dependent variable has six categories.

In this example, the set of regressors is given by *SHI*, which is an indicator of being in the subsidized health care system (1 means being in the

[12] Identification issues necessitate setting the variance in this model equal to 1 and $\alpha_1 = 0$. Observe that multiplying y_i^* by a positive constant or adding a constant to all of the cut-offs and subtracting the same constant from the intercept does not affect y_i.

system), sex (*Female*), age (both linear and squared), socioeconomic conditions indicator (*Est2* and *Est3*), with the lowest being the baseline category, self-perception of health status (*Fair*, *Good*, and *Excellent*), where *Bad* is the baseline, and education level (*PriEd*, *HighEd*, *VocEd*, *UnivEd*), with *no education* as the baseline category.

We ran this application with 50,000 MCMC iterations plus 10,000 as burn-in, and a thinning parameter equal to 5. This setting means 10,000 effective posterior draws. We set $\boldsymbol{\beta}_0 = \mathbf{0}_{11}$, $\boldsymbol{B}_0 = 1000\boldsymbol{I}_{11}$, $\boldsymbol{\gamma}_0 = \mathbf{0}_4$, $\boldsymbol{\Gamma}_0 = \boldsymbol{I}_4$, and the tuning parameter is 1.

We can run the ordered probit models in our GUI following the steps in Algorithm A11.

Algorithm A11 Ordered probit models

1: Select *Univariate Models* on the top panel
2: Select *Ordered Probit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. Remember that this formula must have -1 to omit the intercept in the specification.
7: Set the hyperparameters: mean vector and covariance matrix. This step is not necessary as by default our GUI uses non-informative priors
8: Select the number of ordered alternatives
9: Set the hyperparameters: mean and covariance matrix of the cutoffs. This step is not necessary as by default our GUI uses non-informative prior
10: Select the tuning parameter for the Metropolis-Hastings algorithm
11: Click the *Go!* button
12: Analyze results
13: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

The following **R** code shows how to perform inference in this model using the command *rordprobitGibbs* from the *bayesm* library, which is the command that our GUI uses.

R. code. Determinants of preventive health care visits

```
rm(list = ls())
set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/2HealthMed.csv", sep = ",", header = TRUE, quote = "")
attach(Data)
y <- MedVisPrevOr
# MedVisPrevOr: Oredered variable for preventive visits to doctors in one year: 1 (none), 2 (once), ... 6 (five or more)
X <- cbind(SHI, Female, Age, Age2, Est2, Est3, Fair, Good, Excellent, PriEd, HighEd, VocEd, UnivEd)
k <- dim(X)[2]
L <- length(table(y))
# Hyperparameters
b0 <- rep(0, k); c0 <- 1000; B0 <- c0*diag(k)
gamma0 <- rep(0, L-2); Gamma0 <- diag(L-2)
# MCMC parameters
mcmc <- 60000+1; thin <- 5; tuningPar <- 1/(L-2)^0.5
DataApp <- list(y = y, X = X, k = L)
Prior <- list(betabar = b0, A = solve(B0), dstarbar = gamma0, Ad = Gamma0)
mcmcpar <- list(R = mcmc, keep = 5, s = tuningPar)
PostBeta <- bayesm::rordprobitGibbs(Data = DataApp, Prior = Prior, Mcmc = mcmcpar)
```

R. code. Determinants of preventive health care visits, results

```
BetasPost <- coda::mcmc(PostBeta[["betadraw"]])
colnames(BetasPost) <- c("SHI", "Female", "Age", "Age2", "
    Est2", "Est3", "Fair", "Good", "Excellent", "PriEd", "
    HighEd", "VocEd", "UnivEd")
summary(BetasPost)
Iterations = 1:12000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 12000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean         SD  Naive SE Time-series SE
SHI        0.0654824 2.281e-02 2.082e-04      3.357e-04
Female    -0.0374788 1.908e-02 1.742e-04      1.742e-04
Age        0.0190336 1.869e-03 1.706e-05      4.576e-05
Age2      -0.0002328 2.438e-05 2.225e-07      6.690e-07
Est2       0.0949445 2.226e-02 2.032e-04      4.659e-04
Est3      -0.1383965 3.411e-02 3.114e-04      3.459e-04
Fair       0.6451828 5.375e-02 4.907e-04      3.924e-03
Good       0.7343932 4.955e-02 4.523e-04      4.491e-03
Excellent  0.9826531 6.393e-02 5.836e-04      5.261e-03
PriEd      0.0309418 2.376e-02 2.169e-04      2.221e-04
HighEd    -0.1805753 2.910e-02 2.656e-04      3.456e-04
VocEd      0.1395760 9.640e-02 8.800e-04      9.291e-04
UnivEd    -0.2218120 1.189e-01 1.086e-03      1.086e-03
2. Quantiles for each variable:
        2.5%        25%        50%        75%      97.5%
SHI        0.02090  0.04995  0.06540  0.08085  0.11021
Female    -0.07463 -0.05042 -0.03777 -0.02456  0.00023
Age        0.01550  0.01781  0.01902  0.02023  0.02268
Age2      -0.00028 -0.00024 -0.00023 -0.00021 -0.00018
Est2       0.05149  0.08004  0.09482  0.10968  0.13933
Est3      -0.20559 -0.16144 -0.13815 -0.11563 -0.07179
Fair       0.55799  0.61295  0.64148  0.67268  0.74395
Good       0.66690  0.70808  0.73032  0.75406  0.81064
Excellent  0.88919  0.94770  0.97836  1.01026  1.08460
PriEd     -0.01584  0.01493  0.03101  0.04718  0.07732
HighEd    -0.23782 -0.20035 -0.18021 -0.16073 -0.12435
VocEd     -0.04911  0.07474  0.13811  0.20414  0.33331
UnivEd    -0.45381 -0.30239 -0.22193 -0.14148  0.00863
# Convergence diagnostics
coda::geweke.diag(BetasPost)
coda::raftery.diag(BetasPost,q=0.5,r=0.05,s = 0.95)
coda::heidel.diag(BetasPost)
# Cut offs
Cutoffs <- PostBeta[["cutdraw"]]
summary(Cutoffs)
coda::geweke.diag(Cutoffs)
coda::heidel.diag(Cutoffs)
coda::raftery.diag(Cutoffs[,-1],q=0.5,r=0.05,s = 0.95)
```

The results suggest that older individuals (at a decreasing rate) in the subsidized health program, characterized by being in the second socioeconomic status, with an increasing self-perception of good health, and not having high school as their highest education degree, have a higher probability of visiting a physician for preventive health purposes. Convergence diagnostics look well, except for the self-health perception draws.

We also obtained the posterior estimates of the cutoffs in the ordered probit model. These estimates are necessary to calculate the probability that an individual is in a specific category of physician visits. Due to identification restrictions, the first cutoff is set equal to 0. This is why we observe *NaN* values in [160] and [188] tests, and only four values in the [325] test, which correspond to the remaining free cutoffs. It seems that these cutoff estimates have some convergence issues when using the [325] test as a diagnostic tool. Furthermore, their dependence factors are also very high.

6.7 Negative binomial model

The dependent variable in the negative binomial model is a nonnegative integer or count. In contrast to the Poisson model, the negative binomial model accounts for over-dispersion. The Poisson model assumes equi-dispersion, meaning the mean and variance are equal.

We assume that $y_i \overset{i.i.d.}{\sim} \mathrm{NB}(\gamma, \theta_i)$, where the density function for individual i is

$$\frac{\Gamma(y_i + \gamma)}{\Gamma(\gamma) y_i!} (1 - \theta_i)^{y_i} \theta_i^{\gamma},$$

with the success probability $\theta_i = \frac{\gamma}{\lambda_i + \gamma}$, where $\lambda_i = \exp\left\{\boldsymbol{x}_i^\top \boldsymbol{\beta}\right\}$ is the mean, and $\gamma = \exp\{\alpha\}$ is the target for the number of successful trials, or the dispersion parameter, and $\boldsymbol{x}_i$ is a K-dimensional vector of regressors.

We assume independent priors for this model: $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\alpha \sim G(\alpha_0, \delta_0)$.

This model does not have standard conditional posterior distributions. Therefore, [350] propose using a random-walk Metropolis–Hastings algorithm where the proposal distribution for $\boldsymbol{\beta}$ is Gaussian, centered at the current stage, with covariance matrix $s_{\boldsymbol{\beta}}^2 \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}$, where $s_{\boldsymbol{\beta}}$ is a tuning parameter and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}$ is the maximum likelihood covariance estimator. Additionally, the proposal for α is normal, centered at the current value, with variance $s_\alpha^2 \hat{\sigma}_\alpha^2$, where s_α is a tuning parameter and $\hat{\sigma}_\alpha^2$ is the maximum likelihood variance estimator.

Example: Simulation exercise

Let's do a simulation exercise to check the performance of the M-H algorithms in the negative binomial model. There are two regressors, $x_{i1} \sim$

$U(0,1)$ and $x_{i1} \sim N(0,1)$, and the intercept. The dispersion parameter is $\gamma = \exp\{1.2\}$, and $\boldsymbol{\beta} = [1\ 1\ 1]^\top$. The sample size is 1,000.

We run this simulation using 10,000 MCMC iterations, a burn-in equal to 1,000, and a thinning parameter equal to 5. We set vague priors for the location parameters, particularly, $\boldsymbol{\beta}_0 = \mathbf{0}_3$ and $\boldsymbol{B}_0 = 1000\boldsymbol{I}_3$, and $\alpha_0 = 0.5$ and $\delta_0 = 0.1$, which are the default values in the *rnegbinRw* command from *bayesm* package in **R**. In addition, the tuning parameters of the Metropolis–Hastings algorithms are $s_\beta = 2.93/k^{1/2}$ and $s_\alpha = 2.93$, which are also the default parameters in *rnegbinRw*, k is the number of location parameters.

We can run the negative binomial models in our GUI following the steps in the Algorithm A12.

Algorithm A12 Negative binomial models

1: Select *Univariate Models* on the top panel
2: Select *Negative Binomial (Poisson)* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Set the hyperparameters: mean vector, covariance matrix, shape and scale parameters. This step is not necessary as by default our GUI uses non-informative priors
8: Select the tuning parameters for the Metropolis-Hastings algorithms
9: Click the *Go!* button
10: Analyze results
11: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

The following **R** code shows how to perform inference in the negative binomial model programming the M-H algorithms from scratch. We ask to estimate this example using the *rnegbinRw* command in Exercise 8.

We observe from the results that all 95% credible intervals encompass the population parameters, and the posterior means are very close to the population parameters.

R. code. Simulation of the negative binomial model

```
rm(list = ls())
set.seed(010101)
N <- 2000 # Sample size
x1 <- runif(N); x2 <- rnorm(N)
X <- cbind(1, x1, x2); k <- dim(X)[2]; B <- rep(1, k)
alpha <- 1.2; gamma <- exp(alpha); lambda <- exp(X%*%B)
y <- rnbinom(N, mu = lambda, size = gamma)
# log likelihood
logLik <- function(par){
  alpha <- par[1]; beta <- par[2:(k+1)]
  gamma <- exp(alpha)
  lambda <- exp(X%*%beta)
  logLikNB <- sum(sapply(1:N, function(i){dnbinom(y[i], size
     = gamma, mu = lambda[i], log = TRUE)}))
  return(-logLikNB)
}
# Parameters: Proposal
par0 <- rep(0.5, k+1)
res.optim <- optim(par0, logLik, method="BFGS", hessian=TRUE
    )
res.optim$par
res.optim$convergence
Covar <- solve(res.optim$hessian)
CovarBetas <- Covar[2:(k+1),2:(k+1)]
VarAlpha <- Covar[1:1]
# Hyperparameters: Priors
B0 <- 1000*diag(k); b0 <- rep(0, k)
alpha0 <- 0.5; delta0 <- 0.1
```

R. code. Simulation of the negative binomial model, M-H algorithm

```
# Metropolis-Hastings function
MHfunction <- function(iter, sbeta, salpha){
  Beta <- rep(0, k);  Acept1 <- NULL; Acept2 <- NULL
  BetasPost <- matrix(NA, iter, k); alpha <- 1
  alphaPost <- rep(NA, iter); par <- c(alpha, Beta)
  pb <- winProgressBar(title = "progress bar", min = 0, max = iter, width = 300)
  for(s in 1:iter){
    LogPostBeta <- -logLik(par) + dgamma(alpha, shape = alpha0, scale = delta0, log = TRUE) + mvtnorm::dmvnorm(Beta, mean = b0, sigma = B0, log = TRUE)
    BetaC <- c(MASS::mvrnorm(1, mu = Beta, Sigma = sbeta^2*CovarBetas))
    parC <- c(alpha, BetaC)
    LogPostBetaC <- -logLik(parC) + dgamma(alpha, shape = alpha0, scale = delta0, log = TRUE) +  mvtnorm::dmvnorm(BetaC, mean = b0, sigma = B0, log = TRUE)
    alpha1 <- min(exp((LogPostBetaC - mvtnorm::dmvnorm(BetaC, mean = Beta, sigma = sbeta^2*CovarBetas, log = TRUE))-(LogPostBeta - mvtnorm::dmvnorm(Beta, mean = Beta, sigma = sbeta^2*CovarBetas, log = TRUE))),1)
    u1 <- runif(1)
    if(u1 <= alpha1){Acept1i <- 1; Beta <- BetaC}else{
      Acept1i <- 0; Beta <- Beta
    }
    par <- c(alpha, Beta)
    LogPostBeta <- -logLik(par) + dgamma(alpha, shape = alpha0, scale = delta0, log = TRUE) + mvtnorm::dmvnorm(Beta, mean = b0, sigma = B0, log = TRUE)
    alphaC <- rnorm(1, mean = alpha, sd = salpha*VarAlpha^0.5)
    parC <- c(alphaC, Beta)
    LogPostBetaC <- -logLik(parC) + dgamma(alphaC, shape = alpha0, scale = delta0, log = TRUE) +  mvtnorm::dmvnorm(Beta, mean = b0, sigma = B0, log = TRUE)
    alpha2 <- min(exp((LogPostBetaC - dnorm(alphaC, mean = alpha, sd = salpha*VarAlpha^0.5, log = TRUE))-(LogPostBeta - dnorm(alpha, mean = alpha, sd = salpha*VarAlpha^0.5, log = TRUE))),1)
    u2 <- runif(1)
    if(u2 <= alpha2){Acept2i <- 1; alpha <- alphaC}else{
      Acept2i <- 0; alpha <- alpha
    }

    BetasPost[s, ] <- Beta; alphaPost[s] <- alpha
    Acept1 <- c(Acept1, Acept1i); Acept2 <- c(Acept2, Acept2i)
    setWinProgressBar(pb, s, title=paste( round(s/iter*100, 0),"% done"))
  }
  close(pb)
  AcepRateBeta <- mean(Acept1); AcepRateAlpha <- mean(Acept2)
  Results <- list(AcepRateBeta = AcepRateBeta, AcepRateAlpha = AcepRateAlpha, BetasPost = BetasPost, alphaPost = alphaPost)
  return(Results)
}
```

R. code. Simulation of the negative binomial model, results

```
# MCMC parameters
mcmc <- 10000
burnin <- 1000
thin <- 5
iter <- mcmc + burnin
keep <- seq(burnin, iter, thin)
sbeta <- 2.93/sqrt(k); salpha <- 2.93
# Run M-H
ResultsPost <- MHfunction(iter = iter, sbeta = sbeta, salpha
     = salpha)
ResultsPost$AcepRateBeta
ResultsPost$AcepRateAlpha
summary(coda::mcmc(ResultsPost$BetasPost[keep[-1], ]))
Iterations = 1:2000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 2000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
      Mean      SD  Naive SE Time-series SE
[1,] 1.0270 0.04799 0.0010730      0.0014727
[2,] 0.9981 0.07752 0.0017333      0.0024262
[3,] 0.9677 0.02343 0.0005239      0.0007182
2. Quantiles for each variable:
      2.5%    25%    50%    75% 97.5%
var1 0.9343 0.9943 1.0255 1.0592 1.122
var2 0.8445 0.9448 0.9980 1.0520 1.144
var3 0.9242 0.9512 0.9678 0.9839 1.013
summary(coda::mcmc(ResultsPost$alphaPost[keep[-1]]))
Iterations = 1:2000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 2000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
      Mean             SD       Naive SE Time-series SE
1.282664       0.058769       0.001314       0.001427
2. Quantiles for each variable:
2.5%   25%   50%   75% 97.5%
1.173 1.242 1.282 1.320 1.407
```

6.8 Tobit model

The dependent variable is partially observed in Tobit models due to sampling schemes, whereas the regressors are completely observed. In particular,

$$y_i = \begin{Bmatrix} L, & y_i^* < L, \\ y_i^*, & L \leq y_i^* < U, \\ U, & y_i^* \geq U, \end{Bmatrix}$$

where $y_i^* \overset{i.i.d.}{\sim} N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2)$, $\boldsymbol{x}_i$ is a K-dimensional vector of regressors.[13]

We use conjugate independent priors $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$, and data augmentation using $\boldsymbol{y}_C^*$ such that $y_{C_i}^* \overset{i.n.d.}{\sim} N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2)$, $y_{C_i} = \left\{ y_{C_i^L}^* \cup y_{C_i^U}^* \right\}$ are lower and upper censored data. This allows implementing the Gibbs sampling algorithm [81]. Then,

$$\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{y^*} \mid \boldsymbol{y}, \boldsymbol{X}) \propto \prod_{i=1}^{N} \left[\mathbb{1}(y_i = L)\mathbb{1}(y_{C_i^L}^* < L) + \mathbb{1}(L \leq y_i < U) + \mathbb{1}(y_i = U)\mathbb{1}(y_{C_i^U}^* \geq U) \right]$$
$$\times N(y_i^* \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \times N(\boldsymbol{\beta} \mid \boldsymbol{\beta}_0, \boldsymbol{B}_0) \times IG(\sigma^2 \mid \alpha_0/2, \delta_0/2)$$

The posterior distributions are

$$y_{C_i}^* \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim \begin{Bmatrix} TN_{(-\infty, L)}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \ , \ y_i = L \\ TN_{[U, \infty)}(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \ \ , \ y_i = U \end{Bmatrix},$$

$$\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \sigma^2 \boldsymbol{B}_n),$$
$$\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sigma^{-2}\boldsymbol{X}^\top \boldsymbol{X})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top \boldsymbol{y}^*)$, $\alpha_n = \alpha_0 + N$ and $\delta_n = \delta_0 + (\boldsymbol{y}^* - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{y}^* - \boldsymbol{X}\boldsymbol{\beta})$.

Example: The market value of soccer players in Europe continues

We continue with the example of the market value of soccer players from Section 6.1. We specify the same equation but assume that the sample is censored from below, such that we only have information for soccer players whose market value is higher than one million euros. The dependent variable is *log(ValueCens)*, and the left censoring point is 13.82.

Algorithm A13 illustrates how to estimate Tobit models in our GUI. Our GUI utilizes the *MCMCtobit* command from the *MCMCpack* package.

[13] We can set L or U equal to $-\infty$ or ∞ to model data censored on just one side.

Algorithm A13 Tobit models

1: Select *Univariate Models* on the top panel
2: Select *Tobit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Set the left and right censoring points. To censor above only, specify *-Inf* in the left censoring box, and to censor below only, specify *Inf* in the right censoring box
8: Set the hyperparameters: mean vector, covariance matrix, shape and scale parameters. This step is not necessary as by default our GUI uses non-informative priors
9: Click the *Go!* button
10: Analyze results
11: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

We run this application using the same hyperparameters that we set in the example of Section 6.1. All results seem similar to those in the example of linear models. In addition, the posterior chains seem to achieve good diagnostics.

R. code. The value of soccer player with left censoring

```
rm(list = ls()); set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/1ValueFootballPlayers.csv", sep = ",", header = TRUE, quote = "")
attach(Data)
y <- log(ValueCens)
X <- cbind(1, Perf, Age, Age2, NatTeam, Goals, Exp, Exp2)
k <- dim(X)[2]
N <- dim(X)[1]
# Hyperparameters
d0 <- 0.001; a0 <- 0.001
b0 <- rep(0, k); c0 <- 1000; B0 <- c0*diag(k)
B0i <- solve(B0)
# MCMC parameters
mcmc <- 50000
burnin <- 10000
tot <- mcmc + burnin
thin <- 1
# Posterior distributions using packages: MCMCpack sets the model in terms of the precision matrix
posterior  <- MCMCpack::MCMCtobit(y~X-1, b0=b0, B0 = B0i, c0 = a0, d0 = d0, burnin = burnin, mcmc = mcmc, thin = thin, below = 13.82, above = Inf)
summary(coda::mcmc(posterior))
Iterations = 1:50000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 50000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
        Mean       SD  Naive SE Time-series SE
X         1.045830 2.641038 1.181e-02      1.673e-02
XPerf     0.033990 0.004515 2.019e-05      2.247e-05
XAge      1.025099 0.213368 9.542e-04      1.335e-03
XAge2    -0.021871 0.004004 1.791e-05      2.542e-05
XNatTeam  0.847495 0.125924 5.631e-04      6.393e-04
XGoals    0.010088 0.001649 7.377e-06      7.691e-06
XExp      0.174383 0.069948 3.128e-04      3.846e-04
XExp2    -0.005652 0.002970 1.328e-05      1.561e-05
sigma2    0.982906 0.095965 4.292e-04      6.727e-04
2. Quantiles for each variable:
        2.5%       25%       50%       75%      97.5%
X        -4.174794 -0.725691  1.076420  2.840533  6.1935618
XPerf     0.025110  0.030949  0.033980  0.037003  0.0428650
XAge      0.608620  0.880648  1.023043  1.168486  1.4480001
XAge2    -0.029801 -0.024556 -0.021822 -0.019164 -0.0140990
XNatTeam  0.603953  0.762394  0.846461  0.932056  1.0960274
XGoals    0.006875  0.008977  0.010091  0.011197  0.0133323
XExp      0.038752  0.127167  0.173880  0.221355  0.3122043
XExp2    -0.011483 -0.007623 -0.005654 -0.003662  0.0001615
sigma2    0.811953  0.915246  0.977257  1.043158  1.1879232
```

R. code. The value of soccer player with left censoring, Gibbs sampler

```
# Gibbs sampling functions
XtX <- t(X)%*%X
PostBeta <- function(Yl, sig2){
  Bn <- solve(B0i + sig2^(-1)*XtX)
  bn <- Bn%*%(B0i%*%b0 + sig2^(-1)*t(X)%*%Yl)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostYl <- function(Beta, sig2, L, U, i){
  Ylmean <- X[i,]%*%Beta
  if(y[i] == L){
    Yli <- truncnorm::rtruncnorm(1, a = -Inf, b = L, mean =
    Ylmean, sd = sig2^0.5)
  }else{
    if(y[i] == U){
      Yli <- truncnorm::rtruncnorm(1, a = U, b = Inf, mean =
     Ylmean, sd = sig2^0.5)
    }else{
      Yli <- y[i]
    }
  }
  return(Yli)
}
PostSig2 <- function(Beta, Yl){
  an <- a0 + length(y)
  dn <- d0 + t(Yl - X%*%Beta)%*%(Yl - X%*%Beta)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBetas <- matrix(0, mcmc+burnin, k); Beta <- rep(0, k)
PostSigma2 <- rep(0, mcmc+burnin); sig2 <- 1
L <- log(1000000); U <- Inf
# create progress bar in case that you want to see
    iterations progress
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  Yl <- sapply(1:N, function(i){PostYl(Beta = Beta, sig2 =
    sig2, L = L, U = U, i)})
  Beta <- PostBeta(Yl = Yl, sig2 = sig2)
  sig2 <- PostSig2(Beta = Beta, Yl = Yl)
  PostBetas[s,] <- Beta; PostSigma2[s] <- sig2
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
     "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorBetas <- PostBetas[keep,]
colnames(PosteriorBetas) <- c("Intercept", "Perf", "Age", "
    Age2", "NatTeam", "Goals", "Exp", "Exp2")
summary(coda::mcmc(PosteriorBetas))
summary(coda::mcmc(PostSigma2[keep]))
```

6.9 Quantile regression

In quantile regression, the location parameters vary according to the quantile of the dependent variable. Let $q_\tau(\boldsymbol{x}_i) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau$ denote the τ-th quantile regression function of y_i given $\boldsymbol{x}_i$, where $\boldsymbol{x}_i$ is a K-dimensional vector of regressors, and $0 < \tau < 1$. Specifically, we have the model $y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau + \mu_i$, with the condition $\int_{-\infty}^{0} f_\tau(\mu_i)\, d\mu_i = \tau$, meaning that the τ-th quantile of μ_i is 0.

In particular, [236] propose the asymmetric Laplace distribution for $f_\tau(\mu_i)$, given by

$$f_\tau(\mu_i) = \tau(1-\tau)\exp\left\{-\mu_i(\tau - \mathbb{1}(\mu_i < 0))\right\},$$

where $\mu_i(\tau - \mathbb{1}(\mu_i < 0))$ is the check (loss) function. These authors also propose a location-scale mixture of normals representation, given by

$$\mu_i = \theta e_i + \psi\sqrt{e_i} z_i,$$

where $\theta = \frac{1-2\tau}{\tau(1-\tau)}$, $\psi^2 = \frac{2}{\tau(1-\tau)}$, $e_i \sim E(1)$, and $z_i \sim N(0,1)$, with $e_i \perp z_i$.[14] As a result of this representation and the fact that the sample is i.i.d., the likelihood function is

$$p(\boldsymbol{y} \mid \boldsymbol{\beta}_\tau, \boldsymbol{e}, \boldsymbol{X}) \propto \left(\prod_{i=1}^{N} e_i^{-1/2}\right) \exp\left\{-\sum_{i=1}^{N} \frac{(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau - \theta e_i)^2}{2\psi^2 e_i}\right\}.$$

Assuming a normal prior for $\boldsymbol{\beta}_\tau$, i.e., $\boldsymbol{\beta}_\tau \sim N(\boldsymbol{\beta}_{\tau 0}, \boldsymbol{B}_{\tau 0})$, and using data augmentation for $\boldsymbol{e}$, we can implement a Gibbs sampling algorithm for this model. The posterior distributions are as follows:

$$\boldsymbol{\beta}_\tau \mid \boldsymbol{e}, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_{n\tau}, \boldsymbol{B}_{n\tau}),$$

$$e_i \mid \boldsymbol{\beta}_\tau, \boldsymbol{y}, \boldsymbol{X} \sim \text{GIG}\left(\frac{1}{2}, \alpha_{ni}, \delta_{ni}\right),^{15}$$

where

$$\boldsymbol{B}_{n\tau} = \left(\boldsymbol{B}_{\tau 0}^{-1} + \sum_{i=1}^{N} \frac{\boldsymbol{x}_i \boldsymbol{x}_i^\top}{\psi^2 e_i}\right)^{-1},$$

$$\boldsymbol{\beta}_{n\tau} = \boldsymbol{B}_{n\tau}\left(\boldsymbol{B}_{\tau 0}^{-1}\boldsymbol{\beta}_{\tau 0} + \sum_{i=1}^{N} \frac{\boldsymbol{x}_i(y_i - \theta e_i)}{\psi^2 e_i}\right),$$

$$\alpha_{ni} = \frac{(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau)^2}{\psi^2}, \quad \delta_{ni} = 2 + \frac{\theta^2}{\psi^2}.$$

[14] E denotes an exponential density.
[15] GIG denotes a generalized inverse Gaussian density.

Example: The market value of soccer players in Europe continues

We continue the example of the market value of soccer players from Section 6.1. Now, we want to examine whether the marginal effect of having been on the national team varies with the quantile of the market value of top soccer players in Europe. Thus, we use the same regressors as in the previous example, but analyze the effects at the 0.5-th and 0.9-th quantiles of *NatTeam*.

Algorithm A14 shows how to estimate quantile regression models in our GUI. Our GUI uses the command *MCMCquantreg* from the package *MCMCpack*. The following code demonstrates how to perform this analysis using the package.

The results show that at the median market value (0.5-th quantile), the 95% credible interval for the coefficient associated with *national team* is (0.34, 1.02), with a posterior mean of 0.69. At the 0.9-th quantile, these values are (0.44, 1.59) and 1.03, respectively. It appears that being on the national team increases the market value of more expensive players more significantly on average, although there is some overlap in the credible intervals.

Algorithm A14 Quantile regression

1: Select *Univariate Models* on the top panel
2: Select *Quantile* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Set the quantile to be analyzed, by default is 0.5
8: Set the hyperparameters: mean vector and covariance matrix. This step is not necessary as by default our GUI uses non-informative priors
9: Click the *Go!* button
10: Analyze results
11: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

R. code. The value of soccer player, quantile regression

```
rm(list = ls()); set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/1
    ValueFootballPlayers.csv", sep = ",", header = TRUE,
    quote = "")
attach(Data)
y <- log(ValueCens)
X <- cbind(1, Perf, Age, Age2, NatTeam, Goals, Exp, Exp2)
k <- dim(X)[2]; N <- dim(X)[1]
# Hyperparameters
b0 <- rep(0, k); c0 <- 1000; B0 <- c0*diag(k); B0i <- solve(
    B0)
# MCMC parameters
mcmc <- 50000; burnin <- 10000
tot <- mcmc + burnin; thin <- 1
# Quantile
q <- 0.5
posterior05  <- MCMCpack::MCMCquantreg(y~X-1, tau = q, b0=b0
    , B0 = B0i, burnin = burnin, mcmc = mcmc, thin = thin,
    below = 13.82, above = Inf)
summary(coda::mcmc(posterior05))
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean        SD  Naive SE Time-series SE
X          7.374348 2.916758 1.304e-02      2.291e-02
XPerf      0.029325 0.005938 2.655e-05      5.241e-05
XAge       0.550633 0.241596 1.080e-03      1.903e-03
XAge2     -0.012027 0.004597 2.056e-05      3.643e-05
XNatTeam   0.685275 0.170768 7.637e-04      1.587e-03
XGoals     0.010608 0.002425 1.085e-05      1.951e-05
XExp       0.092561 0.085499 3.824e-04      6.799e-04
XExp2     -0.002979 0.003877 1.734e-05      2.941e-05
2. Quantiles for each variable:
2.5%        25%        50%         75%      97.5%
X          1.74594   5.405772   7.351090   9.2994982 13.216024
XPerf      0.01753   0.025340   0.029354   0.0333155  0.040906
XAge       0.06845   0.390780   0.553187   0.7139430  1.016664
XAge2     -0.02087  -0.015141  -0.012095  -0.0089849 -0.002813
XNatTeam   0.34645   0.572081   0.686735   0.7996086  1.016189
XGoals     0.00578   0.009055   0.010562   0.0121751  0.015403
XExp      -0.06761   0.034149   0.089632   0.1482128  0.267536
XExp2     -0.01094  -0.005456  -0.002891  -0.0004099  0.004466
```

R. code. The value of soccer player, quantile regression

```
q <- 0.9
posterior09  <- MCMCpack::MCMCquantreg(y~X-1, tau = q, b0=b0
    , B0 = B0i, burnin = burnin, mcmc = mcmc, thin = thin,
    below = 13.82, above = Inf)
summary(coda::mcmc(posterior09))
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
           Mean       SD  Naive SE Time-series SE
X          8.860043 5.933902 2.654e-02      6.686e-02
XPerf      0.019663 0.010241 4.580e-05      1.140e-04
XAge       0.532823 0.483213 2.161e-03      5.397e-03
XAge2     -0.012328 0.008864 3.964e-05      9.620e-05
XNatTeam   1.033384 0.294271 1.316e-03      3.389e-03
XGoals     0.008781 0.004340 1.941e-05      4.991e-05
XExp       0.132760 0.177677 7.946e-04      2.125e-03
XExp2     -0.002713 0.007639 3.416e-05      8.531e-05
2. Quantiles for each variable:
           2.5%       25%       50%       75%    97.5%
X         -2.7084122  4.829341  8.821031 12.850002 20.66191
XPerf     -0.0001863  0.012782  0.019605  0.026495  0.03991
XAge      -0.4180422  0.207000  0.532486  0.858221  1.48632
XAge2     -0.0300400 -0.018216 -0.012235 -0.006345  0.00497
XNatTeam   0.4384014  0.840123  1.038986  1.234456  1.59482
XGoals     0.0019513  0.005661  0.008176  0.011327  0.01881
XExp      -0.2320608  0.014760  0.139452  0.256663  0.46053
XExp2     -0.0162717 -0.007954 -0.003198  0.002031  0.01385
```

6.10 Bayesian bootstrap regression

We implement the Bayesian bootstrap [360] for linear regression models. In particular, the Bayesian bootstrap simulates the posterior distributions by assuming that the sample cumulative distribution function (CDF) is the population CDF (this assumption is also implicit in the frequentist bootstrap [122]).

Given $y_i \overset{i.i.d.}{\sim} \mathcal{F}$, where $\mathcal{F}$ does not specify a particular parametric family of distributions, but instead sets $\mathbb{E}(y_i \mid \boldsymbol{x}_i) = \boldsymbol{x}_i^\top \boldsymbol{\beta}$, with $\boldsymbol{x}_i$ being a K-dimensional vector of regressors and $\boldsymbol{\beta}$ a K-dimensional vector of parameters,

the Bayesian bootstrap generates posterior probabilities for each y_i, where the values of $\boldsymbol{y}$ that are not observed have zero posterior probability.

The algorithm to implement the Bayesian bootstrap is the following:

Algorithm A15 Bayesian bootstrap from scratch in linear regression

1: Draw $\boldsymbol{g} \sim Dir(\alpha_1, \alpha_2, \ldots, \alpha_N)$ such that $\alpha_i = 1\ \forall i$.
2: $\boldsymbol{g} = (g_1, g_2, \ldots, g_N)$ is the vector of probabilities to attach to $(y_1, \boldsymbol{x}_1^\top), (y_2, \boldsymbol{x}_2^\top), \ldots, (y_n, \boldsymbol{x}_N^\top)$ for each Bayesian bootstrap replication.
3: Sample $(y_i, \boldsymbol{x}_i^\top)$ N times with replacement and probabilities g_i, $i = 1, 2, \ldots, N$.
4: Estimate $\boldsymbol{\beta}$ using ordinary least squares in the model $\mathbb{E}(\boldsymbol{y} \mid \boldsymbol{X}) = \boldsymbol{X}\boldsymbol{\beta}$, $\boldsymbol{y}$ being a S_1 dimensional vector, and $\boldsymbol{X}$ a $S_1 \times K$ matrix from the previous stage.*
5: Repeat this process S times.
6: The distribution of $\boldsymbol{\beta}^{(s_2)}$ is the Bayesian distribution of $\boldsymbol{\beta}$.

*Ordinary least squares is the posterior mean of $\boldsymbol{\beta}$ using Jeffrey's prior in a linear regression.

Example: Simulation exercise

Let's perform a simulation exercise to evaluate the performance of the Algorithm A15 for inference using the Bayesian bootstrap. The data-generating process is defined by two regressors, each distributed as standard normal. The location vector is $\boldsymbol{\beta} = [1\ 1\ 1]^\top$, with a variance of $\sigma^2 = 1$, and the sample size is 1,000.

Algorithm A16 illustrates how to use our GUI to run the Bayesian bootstrap. Our GUI is based on the *bayesboot* command from the *bayesboot* package in **R**. Exercise 11 asks about using this package to perform inference in this simulation and compares the results with those obtained using our GUI with $S = 10000$.

The following **R** code shows how to program the Bayesian bootstrap from scratch. We observe from the results that all 95% credible intervals encompass the population parameters, and the posterior means are close to the population parameters.

Algorithm A16 Bayesian bootstrap in linear regression

1: Select *Univariate Models* on the top panel
2: Select *Bootstrap* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend. You should see a preview of the dataset
4: Select number of bootstrap replications using the *Range sliders*
5: Select dependent and independent variables using the *Formula builder* table
6: Click the *Build formula* button to generate the formula in **R** syntax. You can modify the formula in the **Main equation** box using valid arguments of the *formula* command structure in **R**
7: Click the *Go!* button
8: Analyze results
9: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

R. code. Bayesian bootstrap

```
rm(list = ls()); set.seed(010101)
N <- 1000; x1 <- runif(N); x2 <- rnorm(N)
X <- cbind(x1, x2); k <- dim(X)[2]
B <- rep(1, k+1); sig2 <- 1
u <- rnorm(N, 0, sig2); y <- cbind(1, X)%*%B + u
data <- as.data.frame(cbind(y, X))
names(data) <- c("y", "x1", "x2")
Reg <- function(d){
  Reg <- lm(y ~ x1 + x2, data = d)
  Bhat <- Reg$coef
  return(Bhat)
}
S <- 10000; alpha <- 1
BB <- function(S, df, alpha){
  Betas <- matrix(NA, S, dim(df)[2])
  N <- dim(df)[1]
  pb <- winProgressBar(title = "progress bar", min = 0, max
    = S, width = 300)
  for(s in 1:S){
    g <- LaplacesDemon::rdirichlet(N, alpha)
    ids <- sample(1:N, size = N, replace = TRUE, prob = g)
    datas <- df[ids,]
    names(datas) <- names(df)
    Bs <- Reg(d = datas)
    Betas[s, ] <- Bs
    setWinProgressBar(pb, s, title=paste( round(s/S*100, 0),
     "% done"))
  }
  close(pb)
  return(Betas)
}
BBs <- BB(S = S, df = data, alpha = alpha)
summary(coda::mcmc(BBs))
```

R. code. Bayesian bootstrap, results

```
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
      Mean      SD  Naive SE Time-series SE
[1,] 0.9172 0.06386 0.0006386      0.0006291
[2,] 1.1733 0.10888 0.0010888      0.0010201
[3,] 1.0137 0.03386 0.0003386      0.0003386
2. Quantiles for each variable:
      2.5%    25%    50%    75% 97.5%
var1 0.7926 0.8743 0.9169 0.9599 1.043
var2 0.9608 1.0984 1.1739 1.2468 1.389
var3 0.9473 0.9910 1.0136 1.0365 1.079
```

6.11 Summary

In this chapter, we present the core univariate regression models and demonstrate how to perform Bayesian inference using Markov Chain Monte Carlo (MCMC) methods. Specifically, we cover a range of algorithms: Gibbs sampling, Metropolis-Hastings, nested Metropolis-Hastings, and Metropolis-Hastings-within-Gibbs. These algorithms form the foundation for performing Bayesian inference in more complex settings using cross-sectional datasets.

6.12 Exercises

1. Get the posterior conditional distributions of the Gaussian linear model assuming independent priors $\pi(\boldsymbol{\beta}, \sigma^2) = \pi(\boldsymbol{\beta}) \times \pi(\sigma^2)$, where $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$.
2. Given the model $y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2/\tau_i)$ (Gaussian linear model with heteroskedasticity) with independent priors, $\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\tau}) = \pi(\boldsymbol{\beta}) \times \pi(\sigma^2) \times \prod_{i=1}^N \pi(\tau_i)$, where $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$, $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$ and $\tau_i \sim G(v/2, v/2)$. Show that $\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n)$,

$\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2)$ and $\tau_i \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim G(v_{1n}/2, v_{2in}/2)$, where $\boldsymbol{\tau} = [\tau_1 \ \ldots \ \tau_n]^\top$, $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sigma^{-2}\boldsymbol{X}^\top \Psi \boldsymbol{X})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top \Psi \boldsymbol{y})$, $\alpha_n = \alpha_0 + N$, $\delta_n = \delta_0 + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top \Psi (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$, $v_{1n} = v + 1$, $v_{2in} = v + \sigma^{-2}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2$, and $\Psi = \text{diagonal}\{\tau_i\}$.

3. **The market value of soccer players in Europe continues**

   Use the setting of the previous exercise to perform inference using a Gibbs sampling algorithm of the the market value of soccer players in Europe setting $v = 5$ and same other hyperparameters as the homoscedastic case. Is there any meaningful difference for the coefficient associated with the national team compared to the application in the homoscedastic case?

4. **Example: Determinants of hospitalization continues**

   Program a Gibbs sampling algorithm in the application of determinants of hospitalization.

5. **Choice of the fishing mode continues**

   - Run the Algorithm A9 of the book to show the results of the Geweke [160], Raftery [325] and Heidelberger [188] tests using our GUI.
   - Use the command *rmnpGibbs* to do the example of the choice of the fishing mode.

6. **Simulation exercise of the multinomial logit model continues**

   Perform inference in the simulation of the multinomial logit model using the command *rmnlIndepMetrop* from the *bayesm* package of **R** and using our GUI.

7. **Simulation of the ordered probit model**

   Simulate an ordered probit model where the first regressor distributes $N(6, 5)$ and the second distributes $G(1, 1)$, the location vector is $\boldsymbol{\beta} = [0.5 \ -0.25 \ 0.5]^\top$, and the cutoffs are in the vector $\boldsymbol{\alpha} = [0 \ 1 \ 2.5]^\top$. Program from scratch a Metropolis-within-Gibbs sampling algorithm to perform inference in this simulation.

8. **Simulation of the negative binomial model continues**

   Perform inference in the simulation of the negative binomial model using the *bayesm* package in **R** software.

9. **The market value of soccer players in Europe continues**

   Perform the application of the value of soccer players with left censuring at one million Euros in our GUI using the Algorithm A13, and the hyperparameters of the example.

10. **The market value of soccer players in Europe continues**

    Program from scratch the Gibbs sampling algorithm in the example of the market value of soccer players at the 0.75 quantile.

11. Use the *bayesboot* package to perform inference in the simulation exercise of Section 6.10, and compared the results with the ones that we get using our GUI setting $S = 10000$.

7

Multivariate models

We describe how to perform Bayesian inference in multivariate response models, including multivariate regression, seemingly unrelated regression, instrumental variables, and the multivariate probit model. In particular, we present the posterior distributions of the parameters and demonstrate several applications and simulations. Additionally, we show how to perform inference in these models using three levels of programming skills: GUI, packages, and programming the algorithms from scratch. Finally, we provide some mathematical and computational exercises.

Remember that we can run our GUI typing

R code. How to display our graphical user interface

```
shiny::runGitHub("besmarter/BSTApp", launch.browser = T)
```

in the **R** package console or any **R** code editor. However, users should see Chapter 5 for seeing other options.

7.1 Multivariate regression

A complete presentation of this model is given in Section 3.4. We show here the setting, and the posterior distributions for facility in exposition. In particular, there are M multiply dependent variables which share the same set of regressors, and their stochastic errors are contemporaneously correlated. In particular, $\boldsymbol{Y} = [\boldsymbol{y_1}\ \boldsymbol{y_2}\ \ldots\ \boldsymbol{y_M}]$ is a $N \times M$ matrix that is generated by $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{U}$ where $\boldsymbol{X}$ is an $N \times K$ matrix of regressors, $\boldsymbol{B} = [\boldsymbol{\beta}_1\ \boldsymbol{\beta}_2 \ldots \boldsymbol{\beta}_M]$ is a $K \times M$ matrix of parameters, and $\boldsymbol{U} = [\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_M]$ is a matrix of stochastic random errors such that $\boldsymbol{\mu}_i \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$, $i = 1, 2, \ldots, N$ is each row of $\boldsymbol{U}$.

The prior is given by $\boldsymbol{B} \mid \boldsymbol{\Sigma} \sim N(\boldsymbol{B}_0, \boldsymbol{V}_0, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \sim IW(\boldsymbol{\Psi}_0, \alpha_0)$. Therefore, the conditional posterior distributions are

$$\boldsymbol{B} \mid \boldsymbol{\Sigma}, \boldsymbol{Y}, \boldsymbol{X} \sim N(\boldsymbol{B}_n, \boldsymbol{V}_n, \boldsymbol{\Sigma}),$$

$$\boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X} \sim IW(\boldsymbol{\Psi}_n, \alpha_n),$$

where $\boldsymbol{V}_n = (\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{V}_0^{-1})^{-1}$, $\boldsymbol{B}_n = \boldsymbol{V}_n(\boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{B}})$, $\hat{\boldsymbol{B}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{Y}$, $\boldsymbol{S} = (\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{B}})^\top(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{B}})$, $\boldsymbol{\Psi}_n = \boldsymbol{\Psi}_0 + \boldsymbol{S} + \boldsymbol{B}_0^\top \boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \widehat{\boldsymbol{B}}^\top \boldsymbol{X}^\top \boldsymbol{X}\widehat{\boldsymbol{B}} - \boldsymbol{B}_n^\top \boldsymbol{V}_n^{-1}\boldsymbol{B}_n$, and $\alpha_n = \alpha_0 + N$. We can use a Gibbs sampling algorithm in this model since the conditional posterior distributions are standard.

Example: The effect of institutions on per capita gross domestic product

To illustrate multivariate regression models, we use the dataset provided by [3], who analyzed the effect of property rights on economic growth.

We begin with the following *simultaneous structural* economic model:[1]

$$\log(\text{pcGDP95}_i) = \beta_1 + \beta_2 \text{PAER}_i + \beta_3 \text{Africa} + \beta_4 \text{Asia} + \beta_5 \text{Other} + u_{1i}, \tag{7.1}$$

$$\text{PAER}_i = \alpha_1 + \alpha_2 \log(\text{pcGDP95}_i) + \alpha_3 \log(\text{Mort}_i) + u_{2i}, \tag{7.2}$$

where *pcGDP95*, *PAER*, and *Mort* represent the per capita gross domestic product (GDP) in 1995, the average index of protection against expropriation between 1985 and 1995, and the settler mortality rate during the period of colonization, respectively. *Africa*, *Asia*, and *Other* are indicator variables for continents, with *America* serving as the baseline group.

In this model, there is *reverse (simultaneous) causality* due to the contemporaneous effect of *GDP* on *PAER*, and vice versa. Therefore, estimating Equations 7.1 and 7.2 without accounting for this phenomenon results in posterior mean estimates that are *biased* and *inconsistent* from a sampling (frequentist) perspective.[2] A potential strategy to address this issue is to estimate the *reduced-form* model, i.e., a model without *simultaneous causality*, where all *endogenous variables* are functions of *exogenous variables*. The former are determined within the model (e.g., $\log(\text{pcGDP95}_i)$ and *PAER* in this example), while the latter are determined outside the model (e.g., $\log(\text{Mort}_i)$, *Africa*, *Asia*, and *Other* in this example).

Replacing Equation 7.2 into Equation 7.1, and solving for $\log(pcGDP95)$,

$$\log(\text{pcGDP95}_i) = \pi_1 + \pi_2 \log(\text{Mort}_i) + \pi_3 \text{Africa} + \pi_4 \text{Asia} + \pi_5 \text{Other} + e_{1i}. \tag{7.3}$$

[1] This model captures the potential underlying economic relationship between the variables.

[2] Note that $\mathbb{E}[u_1 \text{PAER}] \neq 0$, which means failing to meet a necessary condition for obtaining *unbiased* and *consistent* estimators of $\boldsymbol{\beta}$. See Exercise 1.

Then, by substituting Equation 7.3 into Equation 7.2, and solving for *PAER*, we obtain

$$\text{PAER}_i = \gamma_1 + \gamma_2 \log(\text{Mort}_i) + \gamma_3 \text{Africa} + \gamma_4 \text{Asia} + \gamma_5 \text{Other} + e_{2i}, \quad (7.4)$$

where $\pi_2 = \frac{\beta_2\alpha_3}{1-\beta_2\alpha_2}$ and $\gamma_2 = \frac{\alpha_3}{1-\beta_2\alpha_2}$, given that $\beta_2\alpha_2 \neq 1$, i.e., independent equations (see Exercise 2).

Observe that Equations 7.3 and 7.4 have the form of a multivariate regression model, where the common set of regressors is $\boldsymbol{X} =$ [log(Mort) Africa Asia Other] and the common set of dependent variables is $\boldsymbol{Y} =$ [log(pcGDP95) PAER]. Therefore, we can estimate this model using the setup outlined in this section.

In the first stage, we estimate the parameters of the *reduced-form* model (Equations 7.3 and 7.4), but the main interest lies in estimating the parameters of the *structural* model (Equations 7.1 and 7.2). A valid question is whether we can recover (identify) the *structural* parameters from the *reduced-form* parameters. There are two criteria to answer this question: the order condition, which is necessary, and the rank condition, which is both necessary and sufficient.

The order condition

Given a system of equations with M endogenous variables, and K exogenous variables (including the intercept), there are two ways to assess the order condition:

- The parameters of an equation in the system are identified if there are at least $M-1$ variables excluded from the equation (*exclusion restrictions*). The equation is *exactly identified* if the number of excluded variables is $M-1$, and is *over identified* if the number of excluded variables is greater than $M-1$.

- The parameters of equation m in the system are identified if $K - K_m \geq M_m - 1$, where K_m and M_m are the number of exogenous and endogenous variables in equation m, respectively. The m-th equation is *exactly identified* if $K - K_m = M_m - 1$, and *over identified* if if $K - K_m > M_m - 1$.

We can see from Equations 7.1 and 7.2 in this example that $K = 5$, $M = 2$, $K_1 = 4$, $K_2 = 2$, $M_1 = 2$ and $M_2 = 2$. This means that $K - K_1 = 1 = M - 1$ and $K - K_2 = 3 > M - 1 = 1$, that is, the order condition says that both equations satisfy the necessary condition of identification, the first equation would be *exactly identified*, and the second equation would be *over identified*. Observe that there is one excluded variable from the first equation, and there are three excluded variables from the second equation.

The rank condition

The rank condition (necessary and sufficient) says that given a *structural*

model with M equations (M endogenous variables), an equation is identified if and only if there is at least one determinant different from zero from a $(M-1) \times (M-1)$ matrix built using the excluded variables in the analyzed equation, but included in any other equation of the system.

It is useful to build the *identification matrix* to implement the *rank* condition. Table 7.1 shows this matrix in this example.

TABLE 7.1
Identification matrix.

log(pcGDP95)	PAER	Constant	log(Mort)	Africa	Asia	Other
1	-β_2	-β_1	0	-β_3	-β_4	-β_5
-α_2	1	$-\alpha_1$	-α_3	0	0	0

The only excluded variable in the log(pcGDP95) equation is log(Mort). Therefore, there is only one matrix that can be constructed using the excluded variables from this equation, which is $[-\alpha_3]$ (see column 4 in Table 7.1). The determinant of this matrix is $-\alpha_3$, and as long as this coefficient is nonzero (i.e., $\alpha_3 \neq 0$), meaning that the mortality rate is relevant in the PAER equation, the coefficients in the log(pcGDP95) equation are *exactly identified*. For example, $\beta_2 = \frac{\pi_2}{\gamma_2}$, which represents the effect of property rights on GDP, is exactly identified.

It is crucial to observe the importance of excluding log(Mort) from the log(pcGDP95) equation, while including log(Mort) in the PAER equation. This is known as the *exclusion restriction*, which requires the presence of an exogenous source of variability in the PAER equation to help identify the log(pcGDP95) equation. The presence of relevant exogenous sources of variability is an essential factor in the identification, estimation, and inference of *structural* parameters.

As for the identification of the *structural* parameters in the PAER equation, there are three potential matrices that can be constructed: $[-\beta_3]$, $[-\beta_4]$, and $[-\beta_5]$ (see columns 5, 6, and 7 in Table 7.1). As long as any of these parameters are relevant in the log(pcGDP95) equation, the PAER equation is identified. In this case, the PAER equation is *over-identified*, meaning there are multiple ways to estimate the parameters in this equation. For example, $\alpha_2 = \gamma_3/\pi_3 = \gamma_4/\pi_4 = \gamma_5/\pi_5$ (see Exercise 2).

In general, recovering the *structural* parameters from the *reduced-form* parameters can be challenging due to the need for relevant identification restrictions, which can be difficult to find in some applications.[3]

For this example, we set non-informative priors: $\boldsymbol{B}_0 = [\mathbf{0}_5\ \mathbf{0}_5]$, $\boldsymbol{V}_0 = 100\boldsymbol{I}_K$, $\boldsymbol{\Psi}_0 = 5\boldsymbol{I}_2$, and $\alpha_0 = 5$.[4] Once our GUI is displayed (see the be-

[3]Good introductory-level textbooks on identification in linear systems include [174, Chap. 19] and [422, Chap. 16].

[4]Note that we are setting the priors in the *reduced-form* model. This may have unintended consequences for the posterior distributions of the *structural* parameters, which are ultimately the parameters of interest to researchers. For further discussion, see [234, p. 302].

ginning of this chapter), we should follow Algorithm A17 to run multivariate linear models in the GUI (see Chapter 5 for details, particularly on how to set the data set).

Algorithm A17 Multivariate linear model

1: Select *Multivariate Models* on the top panel
2: Select *Simple Multivariate* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select the number of dependent variables in the box **Number of endogenous variables: m**
6: Select the number of independent variables (including the intercept) in the box **Number of exogenous variables: k**
7: Set the hyperparameters: mean vectors, covariance matrix, degrees of freedom, and the scale matrix. This step is not necessary as by default our GUI uses non-informative priors
8: Click the *Go!* button
9: Analyze results
10: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

The following **R** code shows how to perform the Gibss sampling algorithm in this example using the dataset *4Institutions.csv*. We ask to run this example using the *rmultireg* command from the *bayesm* package as an exercise. We find that the posterior mean *structural* effect of property rights on GDP is 0.98, and the 95% credible interval is (0.56, 2.87). This means that there is evidence supporting a positive effect of property rights on gross domestic product.

R code. The effect of institutions on per capita GDP

```
rm(list = ls())
set.seed(12345)
DataInst <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/4Institutions.csv", sep = ",", header = TRUE, quote = "")
attach(DataInst)
Y <- cbind(logpcGDP95, PAER)
X <- cbind(1, logMort, Africa, Asia, Other)
M <- dim(Y)[2]
K <- dim(X)[2]
N <- dim(Y)[1]
# Hyperparameters
B0 <- matrix(0, K, M)
c0 <- 100
V0 <- c0*diag(K)
Psi0 <- 5*diag(M)
a0 <- 5
# Posterior parameters
Bhat <- solve(t(X)%*%X)%*%t(X)%*%Y
S <- t(Y - X%*%Bhat)%*%(Y - X%*%Bhat)
Vn <- solve(solve(V0) + t(X)%*%X)
Bn <- Vn%*%(solve(V0)%*%B0 + t(X)%*%X%*%Bhat)
Psin <- Psi0 + S + t(B0)%*%solve(V0)%*%B0 + t(Bhat)%*%t(X)%*%X%*%Bhat - t(Bn)%*%solve(Vn)%*%Bn
an <- a0 + N
#Posterior draws
s <- 10000 #Number of posterior draws
SIGs <- replicate(s, LaplacesDemon::rinvwishart(an, Psin))
BsCond <- sapply(1:s, function(s) {MixMatrix::rmatrixnorm(n = 1, mean=Bn, U = Vn,V = SIGs[,,s])})
summary(coda::mcmc(t(BsCond)))
SIGMs <- t(sapply(1:s, function(l) {gdata::lowerTriangle(SIGs[,,l], diag=TRUE, byrow=FALSE)}))
summary(coda::mcmc(SIGMs))
hdiBs <- HDInterval::hdi(t(BsCond), credMass = 0.95) # Highest posterior density credible interval
hdiBs
hdiSIG <- HDInterval::hdi(SIGMs, credMass = 0.95) # Highest posterior density credible interval
hdiSIG
beta2 <- BsCond[2,]/BsCond[7,]
summary(coda::mcmc(beta2)) # Effect of property rights on GDP
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean              SD       Naive SE Time-series SE
0.9796         16.8430         0.1684         0.1684
2. Quantiles for each variable:
2.5%    25%    50%    75%  97.5%
0.5604 0.7984 0.9677 1.2329 2.8709
```

7.2 Seemingly unrelated regression

In seemingly unrelated regression (SUR) models, there are M dependent variables, each with potentially different regressors, such that the stochastic errors are contemporaneously correlated. The model is given by:

$$\boldsymbol{y}_m = \boldsymbol{X}_m \boldsymbol{\beta}_m + \boldsymbol{\mu}_m,$$

where $\boldsymbol{y}_m$ is an N-dimensional vector of observations, $\boldsymbol{X}_m$ is an $N \times K_m$ matrix of regressors, $\boldsymbol{\beta}_m$ is a K_m-dimensional vector of location parameters, and $\boldsymbol{\mu}_m$ is an N-dimensional vector of stochastic errors, for $m = 1, 2, \ldots, M$.

Let $\boldsymbol{\mu}_i = [\mu_{i1}\ \mu_{i2}\ \ldots\ \mu_{iM}]^\top$, where $\boldsymbol{\mu}_i \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$. Stacking the M equations, we can write the model as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\mu},$$

where $\boldsymbol{y} = \left[\boldsymbol{y}_1^\top\ \boldsymbol{y}_2^\top\ \ldots\ \boldsymbol{y}_M^\top\right]^\top$ is an MN-dimensional vector, $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_1^\top\ \boldsymbol{\beta}_2^\top\ \ldots\ \boldsymbol{\beta}_M^\top\right]^\top$ is a K-dimensional vector with $K = \sum_{m=1}^M K_m$, and $\boldsymbol{X}$ is an $MN \times K$ block-diagonal matrix composed of the individual $\boldsymbol{X}_m$, i.e.,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_2 & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{X}_M \end{bmatrix}.$$

Similarly, the vector of errors is given by $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^\top\ \boldsymbol{\mu}_2^\top\ \ldots\ \boldsymbol{\mu}_M^\top\right]^\top$, which is an MN-dimensional vector of stochastic errors, with $\boldsymbol{\mu} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_N)$.

The likelihood function for the parameters is then:

$$p(\boldsymbol{\beta}, \boldsymbol{\Sigma} \mid \boldsymbol{y}, \boldsymbol{X}) \propto |\boldsymbol{\Sigma}|^{-N/2} \exp\left\{-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top (\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_N)(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\}.$$

Using independent priors $\pi(\boldsymbol{\beta}) \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\pi(\boldsymbol{\Sigma}^{-1}) \sim W(\alpha_0, \boldsymbol{\Psi}_0)$, the posterior distributions are

$$\boldsymbol{\beta} \mid \boldsymbol{\Sigma}, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

$$\boldsymbol{\Sigma}^{-1} \mid \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{X} \sim W(\alpha_n, \boldsymbol{\Psi}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{X}^\top(\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_N)\boldsymbol{X} + \boldsymbol{B}_0^{-1})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top(\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_N)\boldsymbol{y})$, $\alpha_n = \alpha_0 + N$ and $\boldsymbol{\Psi}_n = (\boldsymbol{\Psi}_0^{-1} + \boldsymbol{U}^\top\boldsymbol{U})^{-1}$, where $\boldsymbol{U}$ is an $N \times M$ matrix whose columns are $\boldsymbol{y}_m - \boldsymbol{X}_m\boldsymbol{\beta}_m$.

We can demonstrate, through straightforward yet tedious algebra, that by

defining $\boldsymbol{y}_i = [y_{i1}\ y_{i2}\ \dots\ y_{iM}]$ and

$$\boldsymbol{X}_i = \begin{bmatrix} x_{1i}^\top & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & x_{2i}^\top & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & x_{Mi}^\top \end{bmatrix},$$

we alternatively have $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{X}_i)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{y}_i)$ and $\boldsymbol{\Psi}_n = (\boldsymbol{\Psi}_0^{-1} + \sum_{i=1}^N (\boldsymbol{y}_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})(\boldsymbol{y}_i - \boldsymbol{X}_i^\top \boldsymbol{\beta})^\top)^{-1}$.

Observe that we have standard conditional posteriors, thus, we can employ a Gibbs sampling algorithm to get the posterior draws.

Example: Utility demand

Let's use the dataset *Utilities.csv* to estimate a seemingly unrelated regression (SUR) model for utilities. We adopt the same setting as in Exercise 14 of Chapter 3, where we estimate a multivariate regression model while omitting households with no consumption in any utility. In this exercise, we observe that not all regressors are relevant for the demand of electricity, water, and gas. Thus, we estimate the following model:

$$\begin{aligned}
\log(\text{electricity}_i) &= \beta_1 + \beta_2 \log(\text{electricity price}_i) + \beta_3 \log(\text{water price}_i) \\
&\quad + \beta_4 \log(\text{gas price}_i) + \beta_5 \text{IndSocio1}_i + \beta_6 \text{IndSocio2}_i + \beta_7 \text{Altitude}_i \\
&\quad + \beta_8 \text{Nrooms}_i + \beta_9 \text{HouseholdMem}_i + \beta_{10} \log(\text{Income}_i) + \mu_{i1} \\
\log(\text{water}_i) &= \alpha_1 + \alpha_2 \log(\text{electricity price}_i) + \alpha_3 \log(\text{water price}_i) \\
&\quad + \alpha_4 \log(\text{gas price}_i) + \alpha_5 \text{IndSocio1}_i + \alpha_6 \text{IndSocio2}_i \\
&\quad + \alpha_7 \text{Nrooms}_i + \alpha_8 \text{HouseholdMem}_i + \mu_{i2} \\
\log(\text{gas}_i) &= \gamma_1 + \gamma_2 \log(\text{electricity price}_i) + \gamma_3 \log(\text{water price}_i) \\
&\quad + \gamma_4 \log(\text{gas price}_i) + \gamma_5 \text{IndSocio1}_i + \gamma_6 \text{IndSocio2}_i + \gamma_7 \text{Altitude}_i \\
&\quad + \gamma_8 \text{Nrooms}_i + \gamma_9 \text{HouseholdMem}_i + \mu_{i3},
\end{aligned}$$

where electricity, water, and gas represent the monthly consumption of electricity (kWh), water (m^3), and gas (m^3) of Colombian households. The dataset includes information on 2,103 households, with details on the average prices of electricity (USD/kWh), water (USD/m^3), and gas (USD/m^3), as well as indicators of the socioeconomic conditions of the neighborhood where the household is located (IndSocio1 being the lowest and IndSocio3 the highest). Additionally, there is information on whether the household is located in a municipality situated at over 1,000 meters above sea level, the number of rooms in the house, the number of household members, and monthly income (USD).

Since each equation has a different set of regressors, and we suspect correlation between the stochastic errors of the three equations, we should estimate a seemingly unrelated regression (SUR) model. We expect unobserved corre-

lation across these equations because we are modeling utilities, and in some cases, a single provider handles all three services and issues one bill.

Algorithm A18 demonstrates how to estimate SUR models using our GUI. Our GUI utilizes the command *rsurGibbs* from the *bayesm* package in **R** software. See Chapter 5 for further details, including instructions on how to set up the dataset, and check the templates available in our GitHub repository (**https://github.com/besmarter/BSTApp**) in the **DataApp** and **DataSim** folders.

The following code shows how to program this application using this package. We use 10000 MCMC iterations, $\boldsymbol{\beta}_0 = \mathbf{0}_{27}$, $\boldsymbol{B}_0 = 100\boldsymbol{I}_{27}$, $\alpha_0 = 5$ and $\boldsymbol{\Psi} = 5\boldsymbol{I}_3$.

We find that the posterior median estimates of the own-price elasticities of demand for electricity, water, and gas are -1.88, -0.36, and -0.62, respectively, and none of the 95% credible intervals encompass 0. This means that a 1% increase in the prices of electricity, water, and gas results in a 1.88%, 0.36%, and 0.62% decrease in the monthly consumption of these utilities, respectively.[5] In general, there is evidence supporting the relevance of all regressors in these equations, with a few exceptions, and unobserved correlation in the demand for these services, which further supports the use of a SUR model in this application.

[5]This is an example where concerns about *biased* and *inconsistent* posterior mean estimates may arise, for instance, due to *reverse causality* between quantity and demand. These concerns are valid; however, we are using micro-level data, which implies no demand-supply simultaneity. Additionally, the utility providers operate in regulated natural monopoly markets, which mitigates endogeneity from searching provider strategies. Finally, we took prices directly from provider records, which avoids potential price measurement errors [332].

Algorithm A18 Seemingly unrelated regression

1: Select *Multivariate Models* on the top panel
2: Select *Seemingly Unrelated Regression* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Select the number of dependent variables in the box **Number of endogenous variables: m**
6: Select the number of independent variables in the box **TOTAL number Exogenous Variables: k**. This is the sum of all exogenous variables over all equations including intercepts. In the example of **Utility demand**, it is equal to 27
7: Set the hyperparameters: mean vectors, covariance matrix, degrees of freedom, and the scale matrix. This step is not necessary as by default our GUI uses non-informative priors
8: Click the *Go!* button
9: Analyze results
10: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

R code. Utility demand in Colombia

```
rm(list = ls())
set.seed(010101)
library(dplyr)
DataUt <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/Utilities.csv", sep = ",", header = TRUE, quote = "")
DataUtEst <- DataUt %>%
filter(Electricity != 0 & Water !=0 & Gas != 0)
attach(DataUtEst)
y1 <- log(Electricity); y2 <- log(Water); y3 <- log(Gas)
X1 <- cbind(1, LnPriceElect, LnPriceWater, LnPriceGas, IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem, Lnincome)
X2 <- cbind(1, LnPriceElect, LnPriceWater, LnPriceGas, IndSocio1, IndSocio2, Nrooms, HouseholdMem)
X3 <- cbind(1, LnPriceElect, LnPriceWater, LnPriceGas, IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem)
regdata <- NULL
regdata[[1]] <- list(y = y1, X = X1); regdata[[2]] <- list(y = y2, X = X2); regdata[[3]] <- list(y = y3, X = X3)
M <- length(regdata); K1 <- dim(X1)[2]; K2 <- dim(X2)[2]; K3 <- dim(X3)[2]
K <- K1 + K2 + K3
# Hyperparameters
b0 <- rep(0, K); c0 <- 100; B0 <- c0*diag(K); V <- 5*diag(M); a0 <- M
Prior <- list(betabar = b0, A = solve(B0), nu = a0, V = V)
#Posterior draws
S <- 10000; keep <- 1; Mcmc <- list(R = S, keep = keep)
PosteriorDraws <- bayesm::rsurGibbs(Data = list(regdata = regdata), Mcmc = Mcmc, Prior = Prior)
```

R code. Utility demand in Colombia, results

```
Bs <- PosteriorDraws[["betadraw"]]
Names <- c("Const", "LnPriceElect", "LnPriceWater", "
    LnPriceGas", "IndSocio1", "IndSocio2",
"Altitude", "Nrooms", "HouseholdMem", "Lnincome", "Const",
"LnPriceElect", "LnPriceWater", "LnPriceGas", "IndSocio1", "
    IndSocio2",
"Nrooms", "HouseholdMem","Const",
"LnPriceElect", "LnPriceWater", "LnPriceGas", "IndSocio1", "
    IndSocio2",
"Altitude", "Nrooms", "HouseholdMem")
colnames(Bs) <- Names
summary(coda::mcmc(Bs))
2. Quantiles for each variable:
          2.5%      25%       50%      75%    97.5%
Const         0.44452  1.03120  1.342407  1.65192  2.25376
LnPriceElect -2.39679 -2.06328 -1.882706 -1.70369 -1.36996
LnPriceWater -0.44221 -0.38678 -0.356850 -0.32669 -0.26969
LnPriceGas   -0.21655 -0.13777 -0.098191 -0.05902  0.01872
IndSocio1    -0.87630 -0.78653 -0.737701 -0.68840 -0.59675
IndSocio2    -0.24601 -0.18286 -0.151440 -0.11896 -0.05681
Altitude     -0.27080 -0.23838 -0.220742 -0.20385 -0.17259
Nrooms        0.04596  0.06178  0.070023  0.07835  0.09422
HouseholdMem  0.06600  0.07994  0.086857  0.09411  0.10785
Lnincome      0.03836  0.05421  0.062957  0.07165  0.08717
Const         0.88957  1.73496  2.169638  2.62170  3.47216
LnPriceElect -0.81956 -0.31624 -0.054075  0.21132  0.71842
LnPriceWater -0.49559 -0.40995 -0.364248 -0.32026 -0.23639
LnPriceGas    0.06075  0.16754  0.226690  0.28570  0.39476
IndSocio1    -0.64203 -0.50302 -0.427819 -0.35226 -0.21315
IndSocio2    -0.50401 -0.40949 -0.359821 -0.31063 -0.21199
Nrooms        0.05688  0.08023  0.093139  0.10555  0.12968
HouseholdMem  0.10041  0.12065  0.131506  0.14260  0.16314
Const        -2.28569 -1.58566 -1.220078 -0.84612 -0.14787
LnPriceElect -2.42484 -2.01228 -1.797269 -1.57889 -1.16396
LnPriceWater -0.10684 -0.03923 -0.004088  0.03153  0.09905
LnPriceGas   -0.76526 -0.67445 -0.625899 -0.57734 -0.48125
IndSocio1    -0.91381 -0.80243 -0.744909 -0.68577 -0.57341
IndSocio2    -0.31791 -0.24388 -0.203300 -0.16415 -0.09012
Altitude      0.24896  0.29099  0.311668  0.33256  0.37278
Nrooms        0.06050  0.07921  0.089386  0.09943  0.11793
HouseholdMem  0.14467  0.16144  0.170024  0.17843  0.19431
summary(coda::mcmc(PosteriorDraws[["Sigmadraw"]]))
2. Quantiles for each variable:
        2.5%     25%     50%     75%  97.5%
var1 0.19912 0.20822 0.21332 0.21863 0.2290
var2 0.08183 0.09284 0.09870 0.10475 0.1160
var3 0.05121 0.05973 0.06426 0.06882 0.0781
var4 0.08183 0.09284 0.09870 0.10475 0.1160
var5 0.47763 0.49934 0.51131 0.52387 0.5493
var6 0.07318 0.08653 0.09351 0.10079 0.1145
var7 0.05121 0.05973 0.06426 0.06882 0.0781
var8 0.07318 0.08653 0.09351 0.10079 0.1145
var9 0.29523 0.30900 0.31654 0.32428 0.3397
```

We ask in the Exercise 5 to run this application using our GUI and the information in the dataset *Utilities.csv*. Observe that this file should be modified to agree the structure that requires our GUI (see the dataset *5Institutions.csv* in the folder *DataApp* of our GitHub repository - **https://github.com/besmarter/BSTApp**- for a template). In addition, we ask to program from scratch the Gibbs sampler algorithm in this application.

7.3 Instrumental variable

This inferential approach is used when there are *endogeneity* issues, that is, when the stochastic error is not independent of the regressors. This, in turn, generates *bias* in posterior mean estimates when we use an inferential approach that does not account for this issue. *Endogeneity* can be caused by *reverse causality*, *omitting relevant correlated variables*, or *measurement error* in the regressors.[6]

Let's specify the dependent variable as a linear function of one endogenous regressor and some exogenous regressors. That is,

$$y_i = \boldsymbol{x}_{ei}^\top \boldsymbol{\beta}_1 + \beta_s x_{si} + \mu_i$$

where $x_{si} = \boldsymbol{x}_{ei}^\top \boldsymbol{\gamma}_1 + \boldsymbol{z}_i^\top \boldsymbol{\gamma}_2 + v_i$, x_s is the variable that generates the endogeneity issues ($\mathbb{E}[\mu \mid x_s] \neq 0$), $\boldsymbol{x}_e$ are K_1 exogenous regressors ($\mathbb{E}[\mu \mid \boldsymbol{x}_e] = \boldsymbol{0}$), and $\boldsymbol{z}$ are K_2 instruments. The instruments are regressors that drive x_s ($\mathbb{E}[x_s \boldsymbol{z}] \neq \boldsymbol{0}$), but do not have a direct effect on y ($\mathbb{E}[y\boldsymbol{z} \mid x_s] = \boldsymbol{0}$). The equation for y is called the *structural equation*, and it is the equation that the researcher is ultimately interested in.

Assuming $(u_i, v_i)^\top \overset{i.i.d.}{\sim} N(0, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} = [\sigma_{lm}]$, $l, m = 1, 2$, the likelihood function is

$$p(\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Sigma} \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\boldsymbol{\Sigma}|^{\frac{N}{2}}} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}, x_{si} - \boldsymbol{w}_i^\top \boldsymbol{\gamma}) \boldsymbol{\Sigma}^{-1} \begin{pmatrix} y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta} \\ x_{si} - \boldsymbol{w}_i^\top \boldsymbol{\gamma} \end{pmatrix} \right\},$$

where $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_1^\top \ \beta_s\right]^\top$, $\boldsymbol{\gamma} = \left[\boldsymbol{\gamma}_1^\top \ \boldsymbol{\gamma}_2^\top\right]^\top$, $\boldsymbol{x}_i = \left[\boldsymbol{x}_{ei}^\top \ x_{si}\right]^\top$ and $\boldsymbol{w}_i = \left[\boldsymbol{x}_{ei}^\top \ \boldsymbol{z}_i^\top\right]^\top$.

We obtain the standard conditional posterior densities by specifying the following independent priors:

$$\boldsymbol{\gamma} \sim N(\boldsymbol{\gamma}_0, \boldsymbol{G}_0), \quad \boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0), \quad \boldsymbol{\Sigma}^{-1} \sim W(\alpha_0, \boldsymbol{\Psi}_0).$$

In particular, the conditional distributions are:

$$\boldsymbol{\beta} \mid \boldsymbol{\gamma}, \boldsymbol{\Sigma}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

[6] See [422, Chap. 15] for an introductory treatment of instrumental variables in the Frequentist inferential approach.

$$\boldsymbol{\gamma} \mid \boldsymbol{\beta}, \boldsymbol{\Sigma}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z} \sim N(\boldsymbol{\gamma}_n, \boldsymbol{G}_n),$$
$$\boldsymbol{\Sigma}^{-1} \mid \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z} \sim W(\alpha_n, \boldsymbol{\Psi}_n),$$

where

$$\boldsymbol{\beta}_n = \boldsymbol{B}_n \left(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \omega_1^{-1} \sum_{i=1}^{N} \left[\boldsymbol{x}_i \left(y_i - \frac{\sigma_{12}(x_{si} - \boldsymbol{w}_i^\top \boldsymbol{\gamma})}{\sigma_{22}} \right) \right] \right),$$

$$\boldsymbol{B}_n = \left(\omega_1^{-1} \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^\top + \boldsymbol{B}_0^{-1} \right)^{-1}, \quad \omega_1 = \sigma_{11} - \frac{\sigma_{12}^2}{\sigma_{22}},$$

$$\boldsymbol{G}_n = \left(\omega_2^{-1} \sum_{i=1}^{N} \boldsymbol{w}_i \boldsymbol{w}_i^\top + \boldsymbol{G}_0^{-1} \right)^{-1}, \quad \boldsymbol{\gamma}_n = \boldsymbol{G}_n \left(\boldsymbol{G}_0^{-1}\boldsymbol{\gamma}_0 + \omega_2^{-1} \sum_{i=1}^{N} \left[\boldsymbol{w}_i \left(x_{si} - \frac{\sigma_{12}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})}{\sigma_{11}} \right) \right] \right),$$

$$\omega_2 = \sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}, \quad \boldsymbol{\Psi}_n = \left[\boldsymbol{\Psi}_0^{-1} + \sum_{i=1}^{N} \begin{pmatrix} y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta} \\ x_{si} - \boldsymbol{w}_i^\top \boldsymbol{\gamma} \end{pmatrix} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}, x_{si} - \boldsymbol{w}_i^\top \boldsymbol{\gamma}) \right]^{-1},$$

$$\alpha_n = \alpha_0 + N, \quad \sigma_{lj} \text{ are the elements of } \boldsymbol{\Sigma}.$$

We also use a Gibbs sampling algorithm in this model since we have standard conditional posterior distributions.

Example: Simulation exercise

Let's simulate the simple process $y_i = \beta_1 + \beta_2 x_{si} + \mu_i$ and $x_{si} = \gamma_1 + \gamma_2 z_i + v_i$ where $[\mu_i \ v_i]^\top \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} = [\sigma_{lj}]$ such that $\sigma_{12} \neq 0$, $i = 1, 2, \ldots, 100$.

Observe that $\mu \mid v \sim N\left(\frac{\sigma_{12}}{\sigma_{22}} v, \sigma_{11} - \frac{\sigma_{21}^2}{\sigma_{22}}\right)$, this implies that $\mathbb{E}[\mu \mid x_s] = \mathbb{E}[\mu \mid v] = \frac{\sigma_{12}}{\sigma_{22}} v \neq 0$ given $\sigma_{12} \neq 0$ and $\mathbb{E}[\mu \mid z] = 0$. Let's set all location parameters equal to 1, and $\sigma_{11} = \sigma_{22} = 1$, $\sigma_{12} = 0.8$, and $z \sim N(0, 1)$. We know from the large sampling properties of the posterior mean that this converge to the maximum likelihood estimator (see Section 1.1, and [243, 404]), which in this setting is $\hat{\beta}_2 = \frac{\widehat{Cov}(x_s, y)}{\widehat{Var}(x_s)}$ which converges in probability to $\beta_2 + \frac{\sigma_{12}}{\sigma_{22} Var(x_s)} = \beta_2 + \frac{\sigma_{12}}{\sigma_{22}(\gamma_2^2 Var(z) + \sigma_{22})} = 1.4$, that is, the asymptotic bias when using the posterior mean of a linear regression without taking into account endogeneity is 0.4 in this example.

We assess the sampling performance of the Bayesian estimators by simulating this setting 100 times. The following code demonstrates how to do this using a linear model that does not account for the *endogeneity* issue (see Section 6.1), as well as how to implement the instrumental variable model using the function *rivGibbs* from the package *bayesm*.[7] In this setup, we use $\boldsymbol{B}_0 = 1000\boldsymbol{I}_2$, $\boldsymbol{\beta}_0 = \mathbf{0}_2$, and the parameters of the inverse gamma distribution

[7] It seems that this function does not account for the effect of the exogenous regressors in the equation for the endogenous regressors.

are set to 0.0005. For the instrumental variable model, we additionally set $\boldsymbol{\gamma}_0 = \mathbf{0}_2$, $\boldsymbol{G}_0 = 1000\boldsymbol{I}_2$, $\alpha_0 = 3$, and $\boldsymbol{\Psi}_0 = 3\boldsymbol{I}_2$.

R code. Simulation exercise, sampling properties ordinary and instrumental models

```
rm(list = ls()); set.seed(010101)
N <- 100; k <- 2
B <- rep(1, k); G <- rep(1, 2); s12 <- 0.8
SIGMA <- matrix(c(1, s12, s12, 1), 2, 2)
z <- rnorm(N); Z <- cbind(1, z); w <- matrix(1,N,1); S <- 100
U <- replicate(S, MASS::mvrnorm(n = N, mu = rep(0, 2), SIGMA))
x <- G[1] + G[2]*z + U[,2,]; y <- B[1] + B[2]*x + U[,1,]
# Hyperparameters
d0 <- 0.001/2; a0 <- 0.001/2
b0 <- rep(0, k); c0 <- 1000; B0 <- c0*diag(k)
B0i <- solve(B0); g0 <- rep(0, 2)
G0 <- 1000*diag(2); G0i <- solve(G0)
nu <- 3; Psi0 <- nu*diag(2)
# MCMC parameters
mcmc <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 1
# Gibbs sampling
Gibbs <- function(x, y){
  Data <- list(y = y, x = x, w = w, z = Z)
  Mcmc <- list(R = mcmc, keep = thin, nprint = 0)
  Prior <- list(md = g0, Ad = G0i, mbg = b0, Abg = B0i, nu = nu, V = Psi0)
  RestIV <- bayesm::rivGibbs(Data = Data, Mcmc = Mcmc, Prior = Prior)
  PostBIV <- mean(RestIV[["betadraw"]])
  ResLM <- MCMCpack::MCMCregress(y ~ x + w - 1, b0 = b0, B0 = B0i, c0 = a0, d0 = d0)
  PostB <- mean(ResLM[,1]); Res <- c(PostB,PostBIV)
  return(Res)
}
PosteriorMeans <- sapply(1:S, function(s) {Gibbs(x = x[,s], y = y[,s])})
rowMeans(PosteriorMeans)
Model <- c(replicate(S, "Ordinary"), replicate(S, "Instrumental"))
postmeans <- c(t(PosteriorMeans))
df <- data.frame(postmeans, Model, stringsAsFactors = FALSE)
library(ggplot2); library(latex2exp)
histExo <- ggplot(df, aes(x = postmeans, fill = Model)) + geom_histogram(bins = 40, position = "identity", color = "black", alpha = 0.5) + labs(title = "Overlayed Histograms", x = "Value", y = "Count") + scale_fill_manual(values = c("blue", "red")) + geom_vline(aes(xintercept = mean(postmeans[1:S])), color = "black", linewidth = 1, linetype = "dashed") + geom_vline(aes(xintercept = mean(postmeans[101:200])), color = "black", linewidth = 1, linetype = "dashed") + geom_vline(aes(xintercept = B[2]), color = "green", linewidth = 1, linetype = "dashed") + xlab(TeX("$E[\\beta_2]$")) + ylab("Frequency") + ggtitle("Histogram: Posterior means simulating 100 samples")
histExo
```

FIGURE 7.1
Histogram of posterior means: Ordinary and instrumental models.

Figure 7.1 displays the histograms of the posterior means of β_2 using the ordinary model, which does not account for endogeneity, and the instrumental variable model. On one hand, the mean of the posterior means for the ordinary model is 1.41 (black dashed line in the red histogram), implying a bias of 0.41, which is very close to the population bias of 0.40. On the other hand, the mean of the posterior means for the instrumental variable model is 1.04 (black dashed line in the blue histogram), which is close to the population value of $\beta_2 = 1$ (green dashed line).

We also observe that the histogram of the posterior means for the ordinary model is less dispersed. That is, this estimator is more efficient, which is a well-known result in the Frequentist inferential approach when comparing ordinary least squares and two-stage least squares (see [420, Chap. 5]).

Two very important aspects in the instrumental variables literature are the *weakness* and *exogeneity* of the instruments. The former refers to how strong the relationship is between the instruments and the endogenous regressors, while the latter refers to the independence of the instruments from the stochastic error in the *structural equation*. In Exercise 6, we ask you to use the previous code as a baseline to study these two aspects. Observe the link between the *weakness* and *exogeneity* of the instrument, and the *exclusion restrictions* ($\mathbb{E}[x_s \boldsymbol{z}] \neq \mathbf{0}$ and $\mathbb{E}[y\boldsymbol{z} \mid x_s] = \mathbf{0}$). This is the point of departure of [96], who propose assessing the plausibility of the *exclusion restrictions* by defining *plausible exogeneity* as having prior information that the effect of the instrument in the *structural equation* is near zero, but perhaps not exactly zero.

Algorithm A19 can be used to estimate the instrumental variable model using our GUI. We ask in Exercise 8 to replicate the example of the effect of institutions on per capita GDP using our GUI.

Algorithm A19 Instrumental variable model

1: Select *Multivariate Models* on the top panel
2: Select *Variable instrumental (two equations)* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Write down the formula of the structural equation in the **Main Equation** box. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation
6: Write down the formula of the endogenous regressor in the **Instrumental Equation** box. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation
7: Set the hyperparameters: mean vectors, covariance matrices, degrees of freedom, and the scale matrix. This step is not necessary as by default our GUI uses non-informative priors
8: Click the *Go!* button
9: Analyze results
10: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

7.4 Multivariate probit model

In the multivariate probit model [121], the response variable $y_{il} = \{0, 1\}$ indicates that individual i makes binary choices among L mutually exclusive alternatives, where $l = 1, 2, \ldots, L$ and $i = 1, 2, \ldots, N$. Specifically,

$$y_{il} = \begin{cases} 0, & y_{il}^* \leq 0 \\ 1, & y_{il}^* > 0 \end{cases}$$

where $\boldsymbol{y}_i^* = \boldsymbol{X}_i \boldsymbol{\beta} + \boldsymbol{\mu}_i \sim \text{i.i.d.}\, N(\mathbf{0}, \boldsymbol{\Sigma})$. Here, $\boldsymbol{y}_i^*$ is an unobserved latent L-dimensional vector, $\boldsymbol{X}_i = \boldsymbol{x}_i^\top \otimes \boldsymbol{I}_L$ is an $L \times K$ design matrix of regressors, with $K = L \times k$, where k is the number of regressors (i.e., the length of $\boldsymbol{x}_i$).

In addition, $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_1^\top \ \boldsymbol{\beta}_2^\top \ldots \boldsymbol{\beta}_k^\top\right]^\top$, where $\boldsymbol{\beta}_j$ forms an L-dimensional vector of coefficients for $j = 1, 2, \ldots, k$.

The likelihood function for this model is given by

$$p(\boldsymbol{\beta}, \boldsymbol{\Sigma} \mid \boldsymbol{y}, \boldsymbol{X}) = \prod_{i=1}^{N} \prod_{l=1}^{L} p_{il}^{y_{il}},$$

where $p_{il} = p(y_{il}^* \geq 0)$.

Observe that $p(y_{il}^* \geq 0) = p(\lambda_{ll} y_{il}^* \geq 0)$, $\lambda_{ll} > 0$. This generates identification issues because just the correlation matrix can be identified, same case as the univariate probit model where the variance of the model is fixed to 1. We follow the post processing strategy proposed by [121] to get identified parameters, that is, $\tilde{\boldsymbol{\beta}} = vec\{\boldsymbol{\Lambda}\boldsymbol{B}\}$ and the correlation matrix $\boldsymbol{R} = \boldsymbol{\Lambda}\boldsymbol{\Sigma}\boldsymbol{\Lambda}$, where $\boldsymbol{\Lambda} = diag\{\sigma_{ll}\}^{-1/2}$ and $\boldsymbol{B} = [\boldsymbol{\beta}_1 \ \boldsymbol{\beta}_2 \ldots \boldsymbol{\beta}_k]$.[8]

We assume independent priors: $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\boldsymbol{\Sigma}^{-1} \sim W(\alpha_0, \boldsymbol{\Psi}_0)$. We can apply Gibbs sampling to this model, as it is a standard Bayesian linear regression model when data augmentation in $\boldsymbol{y}^*$ is used.

The posterior conditional distributions are

$$\boldsymbol{\beta} \mid \boldsymbol{\Sigma}, \boldsymbol{w} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

$$\boldsymbol{\Sigma}^{-1} \mid \boldsymbol{\beta}, \boldsymbol{w} \sim W(\alpha_n, \boldsymbol{\Psi}_n),$$

$$y_{il}^* \mid \boldsymbol{y}_{i,-l}^*, \boldsymbol{\beta}, \boldsymbol{\Sigma}^{-1}, \boldsymbol{y_i} \sim TN_{I_{il}}(m_{il}, \tau_{ll}^2)$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^{*\top}\boldsymbol{X}^*)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^{*\top}\boldsymbol{y}^{**})$, $\boldsymbol{\Sigma}^{-1} = \boldsymbol{C}^\top\boldsymbol{C}$, $\boldsymbol{X}_i^* = \boldsymbol{C}^\top\boldsymbol{X}_i$, $\boldsymbol{y}_i^{**} = \boldsymbol{C}^\top\boldsymbol{y}_i^*$, $\alpha_n = \alpha_0 + N$, $\boldsymbol{\Psi}_n = (\boldsymbol{\Psi}_0 + \sum_{i=1}^N(\boldsymbol{y}_i^* - \boldsymbol{X}_i\boldsymbol{\beta})(\boldsymbol{y}_i^* - \boldsymbol{X}_i\boldsymbol{\beta})^\top)^{-1}$, $m_{il} = \boldsymbol{x}_{il}^\top\boldsymbol{\beta} + \boldsymbol{f}_l^\top(\boldsymbol{y}_{i,-l}^* - \boldsymbol{X}_{i,-l}\boldsymbol{\beta})$, $\boldsymbol{y}_{i,-l}^*$ is an $L-1$ dimensional vector of all components of $\boldsymbol{y}_i^*$ excluding y_{il}^*, $\boldsymbol{x}_{il}^\top$ is the l-th row of $\boldsymbol{X}_i$, $\boldsymbol{X}_{i,-l}$ is $\boldsymbol{X}_i$ after deleting the l-th row, $\boldsymbol{f}_l^\top = \boldsymbol{\omega}_{l,-l}^\top\boldsymbol{\Sigma}_{-l,-l}^{-1}$, $\boldsymbol{\omega}_{l,-l}^\top$ and $\boldsymbol{\Sigma}_{-l,-l}$ are the l-th row of $\boldsymbol{\Sigma}$ extracting the l-th element, and the sub-matrix of $\boldsymbol{\Sigma}$ extracting the l, l element, and $\tau_{ll}^2 = \sigma_{l,l} - \boldsymbol{\omega}_{l,-l}^\top\boldsymbol{\Sigma}_{-l,-l}^{-1}\boldsymbol{\omega}_{-l,l}$, and

$$\boldsymbol{X}^* = \begin{bmatrix} \boldsymbol{X}_1^* \\ \boldsymbol{X}_2^* \\ \vdots \\ \boldsymbol{X}_N^* \end{bmatrix}, \ I_{il} = \begin{Bmatrix} y_{il}^* > 0, & y_{il} = 1 \\ y_{il}^* \leq 0, & y_{il} = 0 \end{Bmatrix}, \text{ and } \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\omega}_1^\top \\ \boldsymbol{\omega}_2^\top \\ \vdots \\ \boldsymbol{\omega}_L^\top \end{bmatrix}.$$

The setting in our GUI has same regressors in each binary decision. However, we can see that the multivariate probit model is similar to a SUR model in latent variables. We ask in Exercise 9 to implement a Gibbs sampling algorithm for a multivariate probit model with different regressors in each equation.

[8] In a Bayesian setting, a model can be non-identified; however, the posterior distribution of the model parameters exists as long as a proper prior distribution is specified [121].

Example: Self selection in hospitalization due to a subsidized health care program

We use the dataset *7HealthMed.csv*, where the dependent variable is $y = [\text{Hosp SHI}]^\top$, with Hosp = 1 if an individual was hospitalized in the year prior to the survey (0 otherwise), and SHI = 1 if the individual had subsidized health insurance (0 otherwise).

Recall that our application of binary response models aimed to uncover the determinants of hospitalization in Medellín (Colombia), where one of the regressors was a binary indicator of participation in a subsidized health care program (Section 6.3). We can use a bivariate probit model if we suspect there is dependence between the decisions regarding these two variables. A priori, we would expect that being in a subsidized health care program increases the probability of hospitalization *ceteris paribus*, due to reduced costs for the patient. However, if an individual expects to be hospitalized in the future, and the factors influencing this decision are unobserved by the modeler, a feedback effect may exist from hospitalization to enrollment in the subsidized health care program.

Algorithm A20 Multivariate probit model

1: Select *Multivariate Models* on the top panel
2: Select *Multivariate Probit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Write down the number of cross-sectional units in the **Number of individuals: n** box
6: Write down the number of exogenous variables in the **Number of exogenous variables: k** box
7: Write down the number of choices in the **Number of choices: l** box
8: Set the hyperparameters: mean vectors, covariance matrix, degrees of freedom, and the scale matrix. This step is not necessary as by default our GUI uses non-informative priors
9: Click the *Go!* button
10: Analyze results
11: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

We considered seven regressors: a constant, gender (female), age, self-perception of health status (with categories fair, good, and excellent, using bad as the reference category), and the proportion of the individual's age spent living in their neighborhood. The last variable attempts to account for social capital, which can affect enrollment in the subsidized health insurance program, as the target population is identified by the local government [328].

The dataset includes 12,975 individuals who can "choose" two options: hospitalization and enrollment in the subsidized health insurance regime.

Algorithm A20 shows how to run a multivariate probit model using our GUI.

R code. Self selection in hospitalization

```
rm(list = ls()); set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/7HealthMed.
    csv", sep = ",", header = TRUE, quote = "")
attach(Data); str(Data)
p <- 2; nd <- 7; N <- length(y)/p; y <- y
Xd <- as.matrix(Data[seq(1, p*N, 2),3:9])
XcreateMP<-function(p,nxs,nind,Data){
  pandterm = function(message) {
    stop(message, call. = FALSE)
  }
  if (missing(nxs))
  pandterm("requires number of regressors: include intercept
    if required")
  if (missing(nind))
  pandterm("requires number of units (individuals)")
  if (missing(Data))
  pandterm("requires dataset")
  if (nrow(Data)!=nind*2)
  pandterm("check dataset! number of units times number
    alternatives should be equal to dataset rows")
  XXDat<-array(0,c(p,1+nxs,nind))
  XX<-array(0,c(p,nxs*p,nind))
  YY<-array(0,c(p,1,nind))
  is<- seq(p,nind*p,p)
  cis<- seq(nxs,nxs*p+1,nxs)
  for(i in is){
    j<-which(i==is)
    XXDat[,,j]<-as.matrix(Data[c((i-(p-1)):i),-1])
    YY[,,j]<-XXDat[,1,j]
    for(l in 1:p){
      XX[l,((cis[l]-(nxs-1)):cis[l]),j]<-XXDat[l,-1,j]
    }
  }
  return(list(y=YY,X=XX))
}
Dat <- XcreateMP(p = p, nxs = nd, nind = N, Data = Data)
y<-NULL; X<-NULL
for(i in 1:dim(Dat$y)[3]){
  y<-c(y,Dat$y[,,i])
  X<-rbind(X,Dat$X[,,i])
}
DataMP = list(p=p, y=y, X=X)
# Hyperparameters
k <- dim(X)[2]; b0 <- rep(0, k); c0 <- 1000
B0 <- c0*diag(k); B0i <- solve(B0)
a0 <- p - 1 + 3; Psi0 <- a0*diag(p)
Prior <- list(betabar = b0, A = B0i, nu = a0, V = Psi0)
# MCMC parameters
mcmc <- 20000; thin <- 5;
Mcmc <- list(R = mcmc, keep = thin)
```

R code. Self selection in hospitalization, results

```
Results <- bayesm::rmvpGibbs(Data = DataMP, Mcmc = Mcmc,
    Prior = Prior)
betatilde1 <- Results$betadraw[,1:7] / sqrt(Results$
    sigmadraw[,1])
summary(coda::mcmc(betatilde1))
Quantiles for each variable:

        2.5%       25%       50%       75%     97.5%
var1 -1.2247602 -1.059257 -0.970870 -0.889741 -0.733411
var2  0.0269885  0.090098  0.121863  0.155585  0.220662
var3  0.0007652  0.002207  0.002925  0.003685  0.005049
var4 -0.7477149 -0.598898 -0.522549 -0.445173 -0.296718
var5 -1.4520842 -1.309922 -1.234633 -1.160468 -1.018396
var6 -1.3503717 -1.182381 -1.092511 -1.005472 -0.837939
var7 -0.1791758 -0.103506 -0.064418 -0.024499  0.051849
betatilde2 <- Results$betadraw[,8:14] / sqrt(Results$
    sigmadraw[,4])
summary(coda::mcmc(betatilde2))
Quantiles for each variable:
        2.5%       25%       50%       75%    97.5%
var1  0.306343  0.477265  0.564698  0.656616  0.82932
var2  0.258347  0.289819  0.305902  0.322116  0.35284
var3  0.007848  0.008656  0.009124  0.009591  0.01045
var4 -0.488810 -0.313532 -0.218459 -0.130373  0.04144
var5 -0.677686 -0.511529 -0.418139 -0.332415 -0.17322
var6 -0.703355 -0.527642 -0.433378 -0.341355 -0.16989
var7  0.164388  0.203623  0.224533  0.245306  0.28513
sigmadraw12 <-  Results$sigmadraw[,3] / (Results$sigmadraw
    [,1]*Results$sigmadraw[,4])^0.5
summary(coda::mcmc(sigmadraw12))
Quantiles for each variable:
2.5%       25%       50%       75%     97.5%
-0.070515 -0.025009 -0.002895  0.018432  0.060986
```

We set 20,000 MCMC iterations with a thinning parameter equal to 5. The hyperparameters are $\boldsymbol{\beta}_0 = \mathbf{0}_{14}$, $\boldsymbol{B}_0 = 100\boldsymbol{I}_{14}$, $\alpha_0 = 4$, and $\boldsymbol{\Psi}_0 = 4\boldsymbol{I}_2$.[9]

The previous **R** code demonstrates how to obtain the posterior draws using the *rmvpGibbs* command from the *bayesm* package. The results suggest that females, older individuals, and those who self-assess their health as poor are more likely to be hospitalized. Furthermore, females, older individuals, and those with a poor or fair self-perception of health, who have lived a larger

[9]Note that the order of the location coefficients in our GUI follows the equations, not the order of the regressors as in the theoretical setting presented in this section. This distinction is important for correctly setting the hyperparameters and interpreting the results of the location parameters.

proportion of their life in their current neighborhood, are more likely to be enrolled in the subsidized health care system. However, the results indicate that there is no unobserved correlation between the two equations, as the 95% credible interval for the correlation is (-0.07, 0.06).

7.5 Summary

In this chapter, we present the setting and posterior distributions of the most common multivariate models. The multivariate framework allows us to address *endogeneity* issues by using the conditional distribution of a multivariate normal vector. Moreover, we always obtain posterior conditional distributions that belong to standard families (multivariate normal, Wishart, and truncated normal) in these models. This property enables the implementation of the Gibbs sampling algorithm for all these models.

7.6 Exercises

1. Show that $\mathbb{E}[u_1 \text{PAER}] = \frac{\alpha_1}{1-\beta_1\alpha_1}\sigma_1^2$, assuming that $\mathbb{E}[u_1 u_2] = 0$, where $\text{Var}(u_1) = \sigma_1^2$, in the example of the effect of institutions on per capita GDP.
2. Show that $\beta_1 = \pi_1/\gamma_1$, in the example of the effect of institutions on per capita GDP.
3. **The effect of institutions on per capita gross domestic product continues I**

   Use the *rmultireg* command from the *bayesm* package to perform inference in the example of the effect of institutions on per capita GDP.
4. **Demand and supply simulation** Given the structural demand-supply model:

$$q_i^d = \beta_1 + \beta_2 p_i + \beta_3 y_i + \beta_4 pc_i + \beta_5 ps_i + u_{i1},$$
$$q_i^s = \alpha_1 + \alpha_2 p_i + \alpha_3 er_i + u_{i2},$$

   where q^d is demand, q^s is supply, p, y, pc, ps, and er are price, income, complementary price, substitute price, and exchange rate, respectively. Complementary and substitute prices refer to the prices of complementary and substitute goods for q. Assume that $\boldsymbol{\beta} = [5\ \ -0.5\ 0.8\ \ -0.4\ 0.7]^\top$, $\boldsymbol{\alpha} = [-2\ 0.5\ \ -0.4]^\top$, $u_1 \sim N(0, 0.5^2)$,

and $u_2 \sim N(0, 0.5^2)$. Additionally, assume that $y \sim N(10, 1)$, $pc \sim N(5, 1)$, $ps \sim N(5, 1)$, and $er \sim N(15, 1)$.

- Find the *reduced-form* model by using the condition that in equilibrium, demand and supply are equal, i.e., $q^d = q^s$. This condition defines the observable quantity, q.
- Simulate p and q from the *reduced-form* equations.
- Perform inference for the *reduced-form* model using the *rmultireg* command from the *bayesm* package.
- Use the posterior draws of the *reduced-form* parameters to perform inference for the *structural* parameters. Any issues? Hint: Are all *structural* parameters exactly identified?

5. **Utility demand continues**

   - Run the **Utility demand** application using our GUI and the information in the dataset *Utilities.csv*. Hint: This file should be modified to agree the structure that requires our GUI (see the dataset *5Institutions.csv* in the folder *DataApp* of our GitHub repository -**https://github.com/besmarter/BSTApp**- for a template).
   - Program from scratch the Gibbs sampler algorithm in this application.

6. **Simulation exercise of instrumental variables continues I**

   - Use the setting of the simulation exercise with instrumental variables to analyze the impact of a weak instrument. For instance, set $\gamma_2 = 0.2$ and compare the performance of the posterior means of the ordinary and instrumental variable models.
   - Perform a simulation to analyze how the degree of exogeneity of the instrument affects the performance of the posterior mean in the instrumental variable model.

7. **Simulation exercise of instrumental variables continues II**

   Program from scratch the Gibbs sampling algorithm of the instrumental model for the simulation exercise of the instrumental variables.

8. **The effect of institutions on per capita gross domestic product continues II**

   Estimate the structural Equation 7.1 using the instrumental variable model where the instrument of PAER is $\log(Mort)$. Compare the effect of property rights on per capita GDP of this model with the effect estimated in the example of the effect of institutions on

per capita gross domestic product. Use the file *6Institutions.csv* to do this exercise in our GUI, and set $\boldsymbol{B}_0 = 100\boldsymbol{I}_5$, $\boldsymbol{\beta}_0 = \mathbf{0}_5$, $\boldsymbol{\gamma}_0 = \mathbf{0}_2$, $\boldsymbol{G}_0 = 100\boldsymbol{I}_2$ $\alpha_0 = 3$ and $\boldsymbol{\Psi}_0 = 3\boldsymbol{I}_2$. The MCMC iterations, burn-in and thinning parameters are 50000, 1000 and 5, respectively.

9. **Multivariate probit with different regressors**

   Let's do a simulation exercise where $y_{i1}^* = 0.5 - 1.2x_{i11} + 0.7x_{i12} + 0.8x_{i3} + \mu_{i1}$ and $y_{i2}^* = 1.5 - 0.8x_{i21} + 0.5x_{i22} + \mu_{i2}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$, where all regressors distribute standard normal, and $N = 5000$. Use $\boldsymbol{\beta}_0 = \mathbf{0}$, $\boldsymbol{B}_0 = 1000\boldsymbol{B}$, $\alpha_0 = 4$ and $\boldsymbol{\Psi}_0 = 4\boldsymbol{I}_2$. Set number of iterations 2000 and a thinning parameter equal to 5.

   - Perform inference using the setting of Section 7.4, that is, assuming that x_{i3} could have an effect on y_{i2}.
   - Program a Gibss sampling algorithm taking into account that there are different regressors in each binary decision, that is, x_{i3} does not have an effect on y_{i2}.

8

Time series models

In this chapter, we provide a brief introduction to performing inference in time series models using a Bayesian framework. There is a large literature in time series in statistics and econometrics, and it would be impossible to present a good treatment in a few pages of an introductory book. However, there are excellent books in Bayesian inference in time series, see for instance, [415, 309, 313].

A time series is a sequence of observations collected in chronological order, allowing us to track how variables change over time. However, it also introduces technical challenges, as we must account for statistical features such as autocorrelation and stationarity. Since time series data is time-dependent, we adjust our notation. Specifically, we use t and T instead of i and N to explicitly indicate time.

Our starting point in this chapter is the *state-space representation* of time series models. Much of the Bayesian inference literature in time series adopts this approach, as it allows dynamic systems to be modeled in a structured way. This representation provides modularity, flexibility, efficiency, and interpretability in complex models where the state evolves over time. It also enables the use of recursive estimation methods, such as the *Kalman filter* for dynamic Gaussian linear models and the *particle filter* (also known as *sequential Monte Carlo*) for non-Gaussian and nonlinear state-space models. The latter method is especially useful for *online* predictions or when there are data storage limitations. These inferential tools are based on the sequential updating process of Bayes' rule, where the posterior at time t becomes the prior at time $t+1$ (see Equation 1.14).

Remember that we can run our GUI typing

R code. How to display our graphical user interface

```
shiny::runGitHub("besmarter/BSTApp", launch.browser = T)
```

in the **R** package console or any **R** code editor, and once our GUI is

deployed, select *Time series Models*. However, users should see Chapter 5 for details.

8.1 State-space representation

A *state-space model* is composed by of an *unobservable state vector* $\boldsymbol{\beta}_t \in \mathbb{R}^K$, and an *observed* measure $\boldsymbol{Y}_t \in \mathbb{R}^M$, $t = 1, 2, \ldots$ such that (i) $\boldsymbol{\beta}_t$ is a *Markov process*, this is, $\pi(\boldsymbol{\beta}_t \mid \boldsymbol{\beta}_{1:t-1}) = \pi(\boldsymbol{\beta}_t \mid \boldsymbol{\beta}_{t-1})$, all the information regarding $\boldsymbol{\beta}_t$ based on all its history up to $t-1$ is carried by $\boldsymbol{\beta}_{t-1}$, and (ii) $\boldsymbol{Y}_t$ is independent of $\boldsymbol{Y}_s$ conditional on $\boldsymbol{\beta}_t$, $s < t$ [309, Chap. 2].

These assumptions imply that $\pi(\boldsymbol{\beta}_{0:t}, \boldsymbol{Y}_{1:t}) = \pi(\boldsymbol{\beta}_0) \prod_{s=1}^{t} \pi(\boldsymbol{\beta}_s \mid \boldsymbol{\beta}_{s-1}) \pi(\boldsymbol{Y}_s \mid \boldsymbol{\beta}_s)$.[1]

There are three key aims in *state-space models*: *filtering*, *smoothing*, and *forecasting*. In *filtering*, we aim to estimate the current state given observations up to time t, specifically obtaining the density $\pi(\boldsymbol{\beta}_s \mid \boldsymbol{y}_{1:t})$ for $s = t$. In *smoothing*, we conduct a retrospective analysis of the system, obtaining $\pi(\boldsymbol{\beta}_s \mid \boldsymbol{y}_{1:t})$ for $s < t$. In *forecasting*, we forecast future observations by first obtaining $\pi(\boldsymbol{\beta}_s \mid \boldsymbol{y}_{1:t})$ as an intermediate step to compute $\pi(\boldsymbol{Y}_s \mid \boldsymbol{y}_{1:t})$ for $s > t$. A valuable feature of these methods is that all these densities can be calculated recursively. [309] show the recursive equations in Propositions 2.1 (filtering), 2.3 (smoothing) and 2.5 (forecasting).

An important class of *state-space models* is the *Gaussian linear state-space model*, also know as, *dynamic linear model*:

$$\begin{aligned} \boldsymbol{Y}_t &= \boldsymbol{X}_t \boldsymbol{\beta}_t + \boldsymbol{\mu}_t && \text{(Observation equations)} \\ \boldsymbol{\beta}_t &= \boldsymbol{G}_t \boldsymbol{\beta}_{t-1} + \boldsymbol{w}_t && \text{(States equations)}, \end{aligned}$$

where $\boldsymbol{\beta}_0 \sim N(\boldsymbol{b}_0, \boldsymbol{B}_0)$, $\boldsymbol{\mu}_t \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}_t)$, $\boldsymbol{w}_t \sim N(\boldsymbol{0}, \boldsymbol{\Omega}_t)$, $\boldsymbol{\beta}_0$, $\boldsymbol{\mu}_t$ and $\boldsymbol{w}_t$ are independent, $\boldsymbol{X}_t$ and $\boldsymbol{G}_t$ are $M \times K$ and $K \times K$ known matrices. Observe that this assumptions implies that $\boldsymbol{Y}_t \mid \boldsymbol{\beta}_t \sim N(\boldsymbol{X}_t \boldsymbol{\beta}_t, \boldsymbol{\Sigma}_t)$, and $\boldsymbol{\beta}_t \mid \boldsymbol{\beta}_{t-1} \sim N(\boldsymbol{G}_t \boldsymbol{\beta}_{t-1}, \boldsymbol{\Omega}_t)$.[2]

Let $\boldsymbol{\beta}_{t-1} \mid \boldsymbol{y}_{1:t-1} \sim N(\boldsymbol{b}_{t-1}, \boldsymbol{B}_{t-1})$, then, we can get the *Kalman filter* by getting

1. The one-step-ahead predictive distribution of $\boldsymbol{\beta}_t$ given $\boldsymbol{y}_{1:t-1}$ is $\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:t-1} \sim N(\boldsymbol{a}_t, \boldsymbol{R}_t)$, where $\boldsymbol{a}_t = \boldsymbol{G}_t \boldsymbol{b}_{t-1}$ and $\boldsymbol{R}_t = \boldsymbol{G}_t \boldsymbol{B}_{t-1} \boldsymbol{G}_t^\top + \boldsymbol{\Omega}_t$.
2. The one-step-ahead predictive distribution of $\boldsymbol{Y}_t$ given $\boldsymbol{y}_{1:t-1}$ is $\boldsymbol{Y}_t \mid \boldsymbol{y}_{1:t-1} \sim N(\boldsymbol{f}_t, \boldsymbol{Q}_t)$, where $\boldsymbol{f}_t = \boldsymbol{X}_t \boldsymbol{a}_t$ and $\boldsymbol{Q}_t = \boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t$.

[1] A *state-space model* where the states are random variables taking discrete values is called *hidden Markov model*.

[2] A general *state-space model* is given by $\boldsymbol{Y}_t = \boldsymbol{f}_t(\boldsymbol{\beta}_t, \boldsymbol{\mu}_t)$, and $\boldsymbol{\beta}_t = \boldsymbol{m}_t(\boldsymbol{\beta}_{t-1}, \boldsymbol{w}_t)$ for arbitrary functions $\boldsymbol{f}_t$ and $\boldsymbol{m}_t$, and distributions for $\boldsymbol{\mu}_t$ and $\boldsymbol{w}_t$, and a prior distribution for $\boldsymbol{\beta}_0$.

3. The distribution of the one-step-ahead prediction error $\boldsymbol{e}_t = \boldsymbol{Y}_t - \mathbb{E}[\boldsymbol{Y}_t \mid \boldsymbol{y}_{1:t-1}] = \boldsymbol{Y}_t - \boldsymbol{f}_t$ is $N(\boldsymbol{0}, \boldsymbol{Q}_t)$ [376, Chap. 6].
4. The filtering distribution of $\boldsymbol{\beta}_t$ given $\boldsymbol{y}_{1:t}$ is $\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:t} \sim N(\boldsymbol{b}_t, \boldsymbol{B}_t)$, where $\boldsymbol{b}_t = \boldsymbol{a}_t + \boldsymbol{K}_t \boldsymbol{e}_t$, $\boldsymbol{K}_t = \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{Q}_t^{-1}$ is the *Kalman gain*, and $\boldsymbol{B}_t = \boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{Q}_t^{-1} \boldsymbol{X}_t \boldsymbol{R}_t$.

The formal proofs of these results can be found in [309, Chap 2]. Just take into account that the logic of these results follow the results of the Seemingly unrelated regression (SUR) model in Section 7.2 for a particular time period. In addition, we know that the posterior distribution using information up to $t-1$ becomes the prior in t (see Equation 1.14, $\pi(\boldsymbol{\theta} \mid \mathbf{y}_{1:t}) \propto p(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:t-1})$). This is the updating process from $\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:t-1} \sim N(\boldsymbol{a}_t, \boldsymbol{R}_t)$ to $\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:t} \sim N(\boldsymbol{b}_t, \boldsymbol{B}_t)$. Moreover, the posterior mean and variance of the SUR model with independent conjugate priors for a particular time period can be written as $\boldsymbol{a}_t + \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} (\boldsymbol{y}_t - \boldsymbol{X}_t \boldsymbol{a}_t)$ and $\boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} \boldsymbol{X}_t \boldsymbol{R}_t^\top$, respectively. Let's see this, we know from Section 7.2 that $\boldsymbol{B}_t = (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{X}_t)^{-1}$ and $\boldsymbol{\beta}_t = \boldsymbol{B}_t (\boldsymbol{R}_t^{-1} \boldsymbol{a}_t + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{y}_t)$. Thus, let's show that both conditional posterior distributions are the same. In particular, the posterior mean in the *state-space representation* is $[\boldsymbol{I}_K - \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} \boldsymbol{X}_t] \boldsymbol{a}_t + \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} \boldsymbol{y}_t$, where

$$\begin{aligned}
\boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} &= \boldsymbol{R}_t \boldsymbol{X}_t^\top [\boldsymbol{\Sigma}_t^{-1} - \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1}] \\
&= \boldsymbol{R}_t [\boldsymbol{I}_K - \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1}] \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \\
&= \boldsymbol{R}_t (\boldsymbol{I}_K - [\boldsymbol{I}_K - \boldsymbol{R}_t^{-1} (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1}]) \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \\
&= (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1},
\end{aligned}$$

where the first equality uses the Woodbury matrix identity (matrix inversion lemma), and the third equality uses $\boldsymbol{D}(\boldsymbol{D} + \boldsymbol{E})^{-1} = \boldsymbol{I} - \boldsymbol{E}(\boldsymbol{D} + \boldsymbol{E})^{-1}$.

Thus, we have the following expression:

$$\begin{aligned}
&[\boldsymbol{I}_K - \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} \boldsymbol{X}_t] \boldsymbol{a}_t + \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} \boldsymbol{y}_t \\
&= [\boldsymbol{I}_K - (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t] \boldsymbol{a}_t + (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{y}_t \\
&= (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{R}_t^{-1} \boldsymbol{a}_t + (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{y}_t \\
&= (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} (\boldsymbol{R}_t^{-1} \boldsymbol{a}_t + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{y}_t) \\
&= (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} (\boldsymbol{R}_t^{-1} \boldsymbol{a}_t + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t \hat{\boldsymbol{\beta}}_t),
\end{aligned}$$

where the second equality uses the identity:

$$\boldsymbol{I} - (\boldsymbol{D} + \boldsymbol{E})^{-1} \boldsymbol{D} = (\boldsymbol{D} + \boldsymbol{E})^{-1} \boldsymbol{E},$$

and the estimator $\hat{\boldsymbol{\beta}}_t$ is defined as:

$$\hat{\boldsymbol{\beta}}_t = (\boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{y}_t.$$

This shows that the posterior mean is a weighted average of the prior mean

and the maximum likelihood estimator (which is the generalized least squares estimator).

The weights are linked to the signal-to-noise ratio, that is, the proportion of the total variability $(\boldsymbol{\Omega}_t + \boldsymbol{\Sigma}_t)$ due to the signal $(\boldsymbol{\Omega}_t)$ versus the noise $(\boldsymbol{\Sigma}_t)$. Note that in the simplest case where $M = K = 1$, and $\boldsymbol{X}_t = \boldsymbol{G}_t = 1$, then $\boldsymbol{K}_t = \boldsymbol{R}_t \boldsymbol{Q}_t^{-1} = (B_{t-1} + \Omega_t)/(B_{t-1} + \Omega_t + \Sigma_t)$. Thus, the weight associated with the observations is equal to 1 if $\Sigma_t = 0$, that is, the posterior mean is equal to the actual observation. On the other hand, if Σ_t increases compare to Ω_t, there is more weight to the prior information, and consequently, the posterior mean is smoother as it heavily dependents on the history. We ask in Exercise 1 to perform simulations with different signal-to-noise ratios to see the effects on the system.

The equality of variances of both approaches is as follows:

$$
\begin{aligned}
Var[\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:t}] &= \boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{X}_t \boldsymbol{R}_t \boldsymbol{X}_t^\top + \boldsymbol{\Sigma}_t)^{-1} \boldsymbol{X}_t \boldsymbol{R}_t \\
&= \boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top (\boldsymbol{\Sigma}_t^{-1} - \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1}) \boldsymbol{X}_t \boldsymbol{R}_t \\
&= \boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t \boldsymbol{R}_t + \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t \boldsymbol{R}_t \\
&= \boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t \boldsymbol{R}_t + \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t [\boldsymbol{I}_K - (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \boldsymbol{R}_t^{-1}] \boldsymbol{R}_t \\
&= \boldsymbol{R}_t - \boldsymbol{R}_t \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1} \\
&= \boldsymbol{R}_t [\boldsymbol{I}_K - \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1}] \\
&= \boldsymbol{R}_t [\boldsymbol{I}_K - (\boldsymbol{I}_K - \boldsymbol{R}_t^{-1} (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1})] \\
&= (\boldsymbol{R}_t^{-1} + \boldsymbol{X}_t^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{X}_t)^{-1},
\end{aligned}
$$

where the second equality uses the Woodbury matrix identity, the fourth equality uses $(\boldsymbol{D} + \boldsymbol{E})^{-1} \boldsymbol{D} = \boldsymbol{I} - (\boldsymbol{D} + \boldsymbol{E})^{-1} \boldsymbol{E}$, and the seventh equality uses $\boldsymbol{D}(\boldsymbol{D} + \boldsymbol{E})^{-1} = \boldsymbol{I} - \boldsymbol{E}(\boldsymbol{D} + \boldsymbol{E})^{-1}$.

The *Kalman filter* allows calculating recursively in a forward way $\pi(\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:t})$ from $\pi(\boldsymbol{\beta}_{t-1} \mid \boldsymbol{y}_{1:t-1})$ starting from $\pi(\boldsymbol{\beta}_0)$.

Let $\boldsymbol{\beta}_{t+1} \mid \boldsymbol{y}_{1:T} \sim N(\boldsymbol{s}_{t+1}, \boldsymbol{S}_{t+1})$, then we can get the *Kalman smoother* by $\boldsymbol{\beta}_t \mid \boldsymbol{y}_{1:T} \sim N(\boldsymbol{s}_t, \boldsymbol{S}_t)$, where $\boldsymbol{s}_t = \boldsymbol{b}_t + \boldsymbol{B}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{R}_{t+1}^{-1} (\boldsymbol{s}_{t+1} - \boldsymbol{a}_{t+1})$ and $\boldsymbol{S}_t = \boldsymbol{B}_t - \boldsymbol{B}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{R}_{t+1}^{-1} (\boldsymbol{R}_{t+1} - \boldsymbol{S}_{t+1}) \boldsymbol{R}_{t+1}^{-1} \boldsymbol{G}_{t+1} \boldsymbol{B}_t$. The proof can be found in [309, Chap 2].

Thus, we can calculate the *Kalman smoother* starting from $t = T - 1$, that is, $\boldsymbol{\beta}_T \mid \boldsymbol{y}_{1:T} \sim N(\boldsymbol{s}_T, \boldsymbol{S}_T)$. However, this is the filtering distribution at T, which means $\boldsymbol{s}_T = \boldsymbol{b}_T$ and $\boldsymbol{S}_T = \boldsymbol{B}_T$, and then, we should proceed recursively in a backward way.

Finally, the forecasting recursion in the *dynamic linear model*, given $\boldsymbol{a}_t(0) = \boldsymbol{b}_t$ and $\boldsymbol{R}_t(0) = \boldsymbol{B}_t$, $h \geq 1$, is given by

1. The forecasting distribution of $\boldsymbol{\beta}_{t+h} \mid \boldsymbol{y}_{1:t}$ is $N(\boldsymbol{a}_t(h), \boldsymbol{R}_t(h))$, where $\boldsymbol{a}_t(h) = \boldsymbol{G}_{t+h} \boldsymbol{a}_t(h-1)$ and $\boldsymbol{R}_t(h) = \boldsymbol{G}_{t+h} \boldsymbol{R}_t(h-1) \boldsymbol{G}_{t+h}^\top + \boldsymbol{\Omega}_{t+h}$.
2. The forecasting distribution $\boldsymbol{Y}_{t+h} \mid \boldsymbol{y}_{1:t}$ is $N(\boldsymbol{f}_t(h), \boldsymbol{Q}_t(h))$, where $\boldsymbol{f}_t(h) = \boldsymbol{X}_{t+h} \boldsymbol{a}_t(h)$ and $\boldsymbol{Q}_t(h) = \boldsymbol{X}_{t+h} \boldsymbol{R}_t(h) \boldsymbol{X}_{t+h}^\top + \boldsymbol{\Sigma}_{t+h}$.

The proof can be found in [309, Chap 2].

These recursive equations allow us to perform probabilistic forecasting h-steps-ahead for the state and observation equations.

These results demonstrate how to use these recursive equations for filtering, smoothing, and forecasting in *dynamic linear models* (*Gaussian linear state-space models*). Although these algorithms appear simple, they suffer from numerical instability, which can lead to non-symmetric and negative-definite covariance matrices. Thus, special care must be taken when working with them.

In addition, this setup assumes that $\boldsymbol{\Sigma}_t$ and $\boldsymbol{\Omega}_t$ are known. However, this is rarely the case in most situations. Therefore, we need to estimate them. One option is to perform maximum likelihood estimation. However, this approach does not account for the uncertainty associated with the fact that $\boldsymbol{\Sigma}_t$ and $\boldsymbol{\Omega}_t$ are unknown when their estimates are *plugged into* the *state space* recursions. On the other hand, we can use a Bayesian approach and perform the recursions associated with each posterior draw of the unknown parameters, thus taking their uncertainty into account.

The point of departure is the posterior distribution, such that

$$\pi(\boldsymbol{\theta}, \boldsymbol{\beta}_0, \ldots, \boldsymbol{\beta}_T \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{G}) \propto \pi(\boldsymbol{\beta}_0 \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}) \prod_{t=1}^{T} \pi(\boldsymbol{\beta}_t \mid \boldsymbol{\beta}_{t-1}, \boldsymbol{\theta})\pi(\boldsymbol{y}_t \mid \boldsymbol{\beta}_t, \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ is the vector of unknown parameters.

We can compute $\pi(\boldsymbol{\beta}_s, \boldsymbol{\theta} \mid \boldsymbol{y}_{1:t}) = \pi(\boldsymbol{\beta}_s \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:t})$, for $s = t$ (*filtering*), $s < t$ (*smoothing*), and $s > t$ (*forecasting*). The marginal posterior distribution of the states is $\pi(\boldsymbol{\beta}_s \mid \boldsymbol{y}_{1:t}) = \int_{\boldsymbol{\Theta}} \pi(\boldsymbol{\beta}_s \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})\pi(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:t}) d\boldsymbol{\theta}$.

We can use the Gibbs sampling algorithm to get the posterior draws in the *dynamic linear model* assuming conjugate families. In particular, let's see the univariate case with *random walk states*,

$$y_t = \boldsymbol{x}_t^\top \boldsymbol{\beta}_t + \mu_t \tag{8.1}$$

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \boldsymbol{w}_t, \tag{8.2}$$

where $\mu_t \sim N(0, \sigma^2)$ and $\boldsymbol{w}_t \sim N(\boldsymbol{0}, \operatorname{diag}\{\omega_1^2, \ldots, \omega_K^2\})$. We assume that $\pi(\sigma^2, \omega_1^2, \ldots, \omega_K^2, \boldsymbol{\beta}_0) = \pi(\sigma^2)\pi(\omega_1^2), \ldots, \pi(\omega_K^2)\pi(\boldsymbol{\beta}_0)$ where $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$, $\omega_k^2 \sim IG(\alpha_{k0}/2, \delta_{k0}/2)$, $k = 1, \ldots, K$, and $\boldsymbol{\beta}_0 \sim N(\boldsymbol{b}_0, \boldsymbol{B}_0)$. Thus, the conditional posterior distributions are $\sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\beta}_{1:T} \sim IG(\alpha_n/2, \delta_n/2)$, where $\alpha_n = T + \alpha_0$ and $\delta_n = \sum_{t=1}^{T}(y_t - \boldsymbol{x}_t^\top \boldsymbol{\beta}_t)^2 + \delta_0$, and $\omega_k^2 \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\beta}_{0:T} \sim IG(\alpha_{kn}/2, \delta_{kn}/2)$, where $\alpha_{kn} = T + \alpha_{k0}$ and $\delta_{kn} = \sum_{t=1}^{T}(\boldsymbol{\beta}_{t,k} - \boldsymbol{\beta}_{t-1,k})^2 + \delta_{k0}$. The vector of the dependent variable is $\boldsymbol{y}$, and all regressors are in $\boldsymbol{X}$.

We also need to sample the states from $\pi(\boldsymbol{\beta}_{1:T} \mid \boldsymbol{y}, \boldsymbol{X}, \sigma^2, \omega_1^2, \ldots, \omega_K^2)$. This can be done using the *forward filtering backward sampling* (FFBS) algorithm [67, 139, 375]. This algorithm is basically a simulation version of the *smoothing* recursion, which allows getting draws of the states, even if we do not have

analytical solutions, for instance, in non-linear settings. See below and [309, Chap. 3] for details. A word of caution here, users should be careful to set non-informative priors in this setting, and in general, settings where there are a large number of parameters (see [234, Chap. 8] for details). Thus, it is useful to use empirical Bayes methods focusing on relevant hyperparameters, for instance, the hyperparameters of the inverse-gamma distributions which define the signal-to-noise ratio.

We use the command *dlmGibbsDIG* from the *dlm* package in our GUI to perform Bayesian inference in the univariate *dynamic linear model* with *random walk states*. This function uses the FFBS algorithm, and assumes independent gamma priors for the precision (inverse of variance) parameters. In addition, this package uses the singular value decomposition to calculate the covariance matrices to avoid numerical instability.

Algorithm A21 shows how to perform inference in the univariate *dynamic linear model* with random walk states in our GUI. See also Chapter 5 for details regarding the dataset structure.

Algorithm A21 Dynamic linear models

1: Select *Time series Model* on the top panel
2: Select *Dynamic linear model* using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Set the hyperparameters of the *precision of the observation equation*: prior mean and variance.
6: Set the hyperparameters of the *pricision of the state equations*: just one set of prior mean and variance parameters.
7: Click the *Go!* button
8: Analyze results
9: Download posterior chains of variances of observation and state equations, and posterior chains of states using the *Download Results* button

Example: Simulation exercise of the dynamic linear model

We simulate the process $y_t = \beta_{t1} + x_t\beta_{t2} + \mu_t$ and $\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \boldsymbol{w}_t$, $t = 1, 2, \ldots, 200$, where $\boldsymbol{\beta}_t = [\beta_{t1} \;\; \beta_{t2}]^\top$, $\mu_t \sim N(0, 0.5^2)$, $\boldsymbol{w}_t \sim N(\mathbf{0}, \operatorname{diag}\{0.2, 0.1\})$, $x_t \sim N(1, 1)$, $\boldsymbol{\beta}_0$ and $\boldsymbol{B}_0$ are the OLS estimates and variance of the recursive OLS estimates (see below), respectively.

The following algorithm demonstrates how to perform inference using *dlmGibbsDIG* and compares the results to those of the maximum likelihood estimator, which is based on the *dlmMLE* function. We also use the *dlmSvd2var* function, which is based on singular value decomposition, to calculate the variance of the smoothing states. All these functions are from the *dlm* package in **R**.

Users can observe that we employ a straightforward strategy for setting the hyperparameters. First, we recursively estimate the model using ordinary least squares (OLS), progressively increasing the sample size, and save the location parameters. Next, we compute the covariance matrix of this sequence and use it to set the priors: the prior mean of the precision of the state vector is set equal to the inverse of the maximum element of the main diagonal of this covariance matrix ($a.theta$), and the prior variance is set equal to ten times this value ($b.theta$). For the observation equation, the prior mean of the precision is set equal to the inverse of the OLS variance estimate ($a.y$), and the prior variance is set equal to ten times this value ($b.y$). We perform some sensitivity analysis of the results regarding the hyperparameters, and it seems that the results are robust. However, we encourage giving more consideration to empirical Bayes methods for setting hyperparameters in *state-space models*.

R code. Simulation: Dynamic linear model

```
rm(list = ls()); set.seed(010101)
T <- 200; sig2 <- 0.5^2
x <- rnorm(T, mean = 1, sd = 1)
X <- cbind(1, x); B0 <- c(1, 0.5)
K <- length(B0)
e <- rnorm(T, mean = 0, sd = sig2^0.5)
Omega <- diag(c(0.2, 0.1))
w <- MASS::mvrnorm(T, c(0, 0), Omega)
Bt <- matrix(NA, T, K); Bt[1,] <- B0
yt <- rep(NA, T)
yt[1] <- X[1,]%*%B0 + e[1]
for(t in 1:T){
  if(t == 1){
    Bt[t,] <- w[t,]
  }else{
    Bt[t,] <- Bt[t-1,] + w[t,]
  }
  yt[t] <- X[t,]%*%Bt[t,] + e[t]
}
RegLS <- lm(yt ~ x)
SumRegLS <- summary(RegLS)
SumRegLS; SumRegLS$sigma^2
Bp <- matrix(RegLS$coefficients, T, K, byrow = TRUE)
S <- 20
for(t in S:T){
  RegLSt <- lm(yt[1:t] ~ x[1:t])
  Bp[t,] <- RegLSt$coefficients
}
# plot(Bp[S:T,2], type = "l")
VarBp <- var(Bp)
# State space model
ModelReg <- function(par){
  Mod <- dlm::dlmModReg(x, dV = exp(par[1]), dW = exp(par
    [2:3]), m0 = RegLS$coefficients, C0 = VarBp)
  return(Mod)
}
outMLEReg <- dlm::dlmMLE(yt, parm = rep(0, K+1), ModelReg)
exp(outMLEReg$par)
RegFilter <- dlm::dlmFilter(yt, ModelReg(outMLEReg$par))
RegSmoth <- dlm::dlmSmooth(yt, ModelReg(outMLEReg$par))
SmoothB2 <- RegSmoth$s[-1,2]
VarSmooth <- dlm::dlmSvd2var(u = RegSmoth[["U.S"]], RegSmoth
    [["D.S"]])
SDVarSmoothB2 <- sapply(2:(T+1), function(t){VarSmooth[[t]][
    K,K]^0.5})
LimInfB2 <- SmoothB2 - qnorm(0.975)*SDVarSmoothB2
LimSupB2 <- SmoothB2 + qnorm(0.975)*SDVarSmoothB2
# Gibbs
MCMC <- 2000; burnin <- 1000
a.y <- (SumRegLS$sigma^2)^(-1); b.y <- 10*a.y; a.theta <- (
    max(diag(VarBp)))^(-1); b.theta <- 10*a.theta
gibbsOut <- dlm::dlmGibbsDIG(yt, mod = dlm::dlmModReg(x), a.
    y = a.y, b.y = b.y, a.theta = a.theta, b.theta = b.theta
    , n.sample = MCMC, thin = 5, save.states = TRUE)
```

R code. Simulation: Dynamic linear model

```
B2t <- matrix(0, MCMC - burnin, T + 1)
for(t in 1:(T+1)){
  B2t[,t] <- gibbsOut[["theta"]][t,2,-c(1:burnin)]
}
Lims <- apply(B2t, 2, function(x){quantile(x, c(0.025,
    0.975))})
summary(coda::mcmc(gibbsOut[["dV"]]))
summary(coda::mcmc(gibbsOut[["dW"]]))
# Figure
require(latex2exp) # LaTeX equations in figures
xx <- c(1:(T+1), (T+1):1)
yy <- c(Lims[1,], rev(Lims[2,]))
plot   (xx, yy, type = "n", xlab = "Time", ylab = TeX("$\\
    beta_{t2}$"))
polygon(xx, yy, col = "lightblue", border = "lightblue")
xxML <- c(1:T, T:1)
yyML <- c(LimInfB2, rev(LimSupB2))
polygon(xxML, yyML, col = "blue", border = "blue")
lines(colMeans(B2t), col = "red", lw = 2)
lines(Bt[,2], col = "black", lw = 2)
lines(SmoothB2, col = "green", lw = 2)
title("State vector: Slope parameter")
```

Figure 8.1 shows the comparison between maximum likelihood (ML) and Bayesian inference. The light blue (Bayesian) and dark blue (maximum likelihood) shadows show the credible and confidence intervals at 95% for the state slope parameter (β_{t2}). We see that the Bayesian interval encompass the ML interval. This is a reflection of the extra uncertainty of the unknown variances. The black line is the actual trajectory of β_{t2}, the green and red lines are the *smoothing* recursions using the ML and Bayesian estimates (posterior mean), respectively.

Example: Effects of inflation on interest rate I

We use the dataset *16INTDEF.csv* provided by [422, Chaps. 10] to study the effects of inflation on the interest rate. The specification is

$$\Delta i_t = \beta_{t1} + \beta_{t2}\Delta\text{inf}_t + \beta_{t3}\Delta\text{def}_t + \mu_t$$

and

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \boldsymbol{w}_t,$$

where $\Delta z_t = z_t - z_{t-1}$ is the difference operator, i_t is the three-month T-bill rate, inf_t is the annual inflation rate based on the consumer price index (CPI), and def_t is the federal budget deficit as a percentage of gross domestic

FIGURE 8.1
Simulation: Dynamic linear model.

product (GDP) from 1948 to 2003 in the USA. In addition, $\mu_t \sim N(0, \sigma^2)$ and $\boldsymbol{w}_t \sim N(\mathbf{0}, \text{diag}\{\omega_1^2, \omega_2^2\})$. We assume inverse-gamma distributions for the priors of the scale parameters and set 12,000 MCMC iterations, 2,000 as burn-in, and 10 as the thinning parameter.

The following code shows how to perform this application. We use the variance of the recursive estimation of OLS to set the hyperparameters of the inverse-gamma distribution for the variances of $\boldsymbol{w}_t$, and the OLS estimate of the variance of the model to set the hyperparameters of the distribution of σ^2. Note that, as we are using the function *dlmGibbsDIG* from the *dlm* package, the hyperparameters are set in terms of precision parameters.

Figure 8.2 shows the posterior results of the effect of inflation on the interest rate. This is a fan chart indicating deciles from 10% to 90%. The red shaded area shows the range around the median value, and the black line represents the mean value of the state associated with the annual change in inflation. We see that the annual changes in interest rates are weakly positively related to annual changes in inflation.

R code. Dynamic linear model: Effects of inflation on interest rate

```
rm(list = ls()); set.seed(010101)
DataIntRate <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/16INTDEF.csv", sep = ",", header = TRUE, quote = "")
attach(DataIntRate); Xt <- cbind(diff(inf), diff(def))
K <- dim(Xt)[2] + 1; yt <- diff(i3)
T <- length(yt); RegLS <- lm(yt ~ Xt)
SumRegLS <- summary(RegLS); SumRegLS; SumRegLS$sigma^2
# Recursive OLS
Bp <- matrix(RegLS$coefficients, T, K, byrow = TRUE)
S <- 20
for(t in S:T){
  RegLSt <- lm(yt[1:t] ~ Xt[1:t,])
  Bp[t,] <- RegLSt$coefficients
}
VarBp <- var(Bp)
# State space model
ModelReg <- function(par){
  Mod <- dlm::dlmModReg(Xt, dV = exp(par[1]), dW = exp(par[2:(K+1)]), m0 = RegLS$coefficients,
  C0 = diag(VarBp))
  return(Mod)
}
MCMC <- 12000; burnin <- 2000; thin <- 10
a.y <- (SumRegLS$sigma^2)^(-1); b.y <- 10*a.y; a.theta <- (max(diag(VarBp)))^(-1); b.theta <- 10*a.theta
gibbsOut <- dlm::dlmGibbsDIG(yt, mod = dlm::dlmModReg(Xt), a.y = a.y, b.y = b.y, a.theta = a.theta, b.theta = b.theta, n.sample = MCMC, thin = 5, save.states = TRUE)
B2t <- matrix(0, MCMC - burnin, T + 1)
for(t in 1:(T+1)){
  B2t[,t] <- gibbsOut[["theta"]][t,2,-c(1:burnin)]
}
dV <- coda::mcmc(gibbsOut[["dV"]][-c(1:burnin)])
dW <- coda::mcmc(gibbsOut[["dW"]][-c(1:burnin),])
summary(dV); summary(dW)
plot(dV); plot(dW)
library(fanplot); library(latex2exp)
df <- as.data.frame(B2t)
plot(NULL, main="Percentiles", xlim = c(1, T+1), ylim = c(-1, 2), xlab = "Time", ylab = TeX("$\\beta_{t1}$"))
fan(data = df); lines(colMeans(B2t), col = "black", lw = 2)
abline(h=0, col = "blue")
```

We can extend the *dynamic linear model* with *random walk states* to take into account time invariant location parameters. In particular, we follow [104],

FIGURE 8.2
Effects of inflation on interest rate: Dynamic linear model.

who propose the *simulation smoother*. This algorithm overcomes some shortcomings of the FFBS algorithm, such as slow convergence and computational overhead. We focus on the case $M = 1$,

$$y_t = \boldsymbol{z}_t^\top \boldsymbol{\alpha} + \boldsymbol{x}_t^\top \boldsymbol{\beta}_t + \boldsymbol{h}_t^\top \boldsymbol{\epsilon}_t, \quad t = 1, 2, \ldots, T. \text{ (Observation equation)} \tag{8.3}$$

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + \boldsymbol{H}_t \boldsymbol{\epsilon}_t, \quad t = 1, 2, \ldots, T. \text{ (States equations)}, \tag{8.4}$$

where $\boldsymbol{z}_t$ and $\boldsymbol{x}_t$ are L-dimensional and K-dimensional vectors of regressors associated with time-invariant and time-varying parameters, respectively, $\boldsymbol{h}_t$ is a vector of dimension $1 + K$, $\boldsymbol{H}_t$ is a matrix of dimension $K \times (1 + K)$, $\boldsymbol{\beta}_0 = \mathbf{0}$ and $\boldsymbol{\epsilon}_t \sim N(\mathbf{0}_{1+K}, \sigma^2 \boldsymbol{I}_{1+K})$.

Observe that this specification encompasses Equations 8.1 and 8.2 setting $\boldsymbol{\epsilon}_t = [\mu_t \ \boldsymbol{w}_t^\top]^\top$, $\boldsymbol{h}_t = [1 \ 0 \ \ldots \ 0]$, $\boldsymbol{H}_t = [\mathbf{0}_K \ \boldsymbol{U}_{K\times K}]$ such that $\text{diag}\left\{\omega_1^2 \ \ldots \ \omega_K^2\right\} = \sigma^2 \boldsymbol{U}\boldsymbol{U}^\top$, $\boldsymbol{\alpha} = \mathbf{0}$, and $\boldsymbol{h}_t \boldsymbol{H}_t^\top = \mathbf{0}_K$.

The nice idea of [104] was to propose an efficient algorithm to get draws from $\boldsymbol{\eta}_t = \boldsymbol{F}_t \boldsymbol{\epsilon}_t$, where the most common choice is $\boldsymbol{F}_t = \boldsymbol{H}_t$, which means drawing samples from the perturbations of the states, and then, recovering the states from Equation 8.4 and $\boldsymbol{\beta}_0 = \mathbf{0}$. [104] present a more general version of the *state space model* that the one presented here.

Using the system given by Equations 8.3 and 8.4, $\boldsymbol{F}_t = \boldsymbol{H}_t$ and $\boldsymbol{h}_t \boldsymbol{H}_t^\top = \mathbf{0}_K$, the *filtering* recursions are given by $e_t = Y_t - \boldsymbol{z}_t^\top \boldsymbol{\alpha} - \boldsymbol{x}_t^\top \boldsymbol{b}_{t-1}$, $q_t = \boldsymbol{x}_t^\top \boldsymbol{B}_{t-1} \boldsymbol{x}_t + \boldsymbol{h}_t^\top \boldsymbol{h}_t$, $\boldsymbol{K}_t = \boldsymbol{B}_{t-1} \boldsymbol{x}_t q_t^{-1}$, $\boldsymbol{b}_t = \boldsymbol{b}_{t-1} + \boldsymbol{K}_t e_t$, and $\boldsymbol{B}_t = \boldsymbol{B}_{t-1} - \boldsymbol{B}_{t-1} \boldsymbol{x}_t \boldsymbol{K}_t^\top + \boldsymbol{H}_t \boldsymbol{H}_t^\top$, where $\boldsymbol{b}_0 = \mathbf{0}$ and $\boldsymbol{B}_0 = \boldsymbol{H}_0 \boldsymbol{H}_0^\top$. See system 2 in [104] for a more general case. We should save e_t (innovation vector), q_t (scale innovation variance) and $\boldsymbol{K}_t$ (*Kalman gain*) from this recursion.

Then, setting $\boldsymbol{r}_T = 0$ and $\boldsymbol{M}_T = \boldsymbol{0}$, we run backwards from $t = T-1, T-2, \ldots, 1$, the following recursions: $\boldsymbol{\Lambda}_{t+1} = \boldsymbol{H}_{t+1}\boldsymbol{H}_{t+1}^\top$, $\boldsymbol{C}_{t+1} = \boldsymbol{\Lambda}_{t+1} - \boldsymbol{\Lambda}_{t+1}\boldsymbol{M}_{t+1}\boldsymbol{\Lambda}_{t+1}^\top$, $\boldsymbol{\xi}_{t+1} \sim N(\boldsymbol{0}_K, \sigma^2\boldsymbol{C}_{t+1})$, $\boldsymbol{L}_{t+1} = \boldsymbol{I}_K - \boldsymbol{K}_{t+1}\boldsymbol{x}_{t+1}^\top$, $\boldsymbol{V}_{t+1} = \boldsymbol{\Lambda}_{t+1}\boldsymbol{M}_{t+1}\boldsymbol{L}_{t+1}$, $\boldsymbol{r}_t = \boldsymbol{x}_{t+1}e_{t+1}/q_{t+1} + \boldsymbol{L}_{t+1}^\top\boldsymbol{r}_{t+1} - \boldsymbol{V}_{t+1}^\top\boldsymbol{C}_{t+1}^{-1}\boldsymbol{\xi}_{t+1}$, $\boldsymbol{M}_t = \boldsymbol{x}_{t+1}\boldsymbol{x}_{t+1}^\top/q_{t+1} + \boldsymbol{L}_{t+1}^\top\boldsymbol{M}_{t+1}\boldsymbol{L}_{t+1} + \boldsymbol{V}_{t+1}^\top\boldsymbol{C}_{t+1}^{-1}\boldsymbol{V}_{t+1}$, and $\boldsymbol{\eta}_{t+1} = \boldsymbol{\Lambda}_{t+1}\boldsymbol{r}_{t+1} + \boldsymbol{\xi}_{t+1}$. [104] show that $\boldsymbol{\eta} = [\boldsymbol{\eta}_1^\top \ \ldots \ \boldsymbol{\eta}_T^\top]^\top$ is drawn from $p(\boldsymbol{H}_t\boldsymbol{\epsilon}_t \mid y_t, \boldsymbol{x}_t, \boldsymbol{z}_t, \boldsymbol{h}_t, \boldsymbol{H}_t, \boldsymbol{\alpha}, \sigma^2, t = 1, 2, \ldots, T)$. Thus, we can recover $\boldsymbol{\beta}_t$ using 8.4 and $\boldsymbol{\beta}_0 = \boldsymbol{0}_K$.

We assume in the model given by Equations 8.3 and 8.4 that $\boldsymbol{h}_t = [1 \ 0 \ \ldots \ 0]^\top$ and $\boldsymbol{H}_t = [\boldsymbol{0}_K \ \operatorname{diag}\{1/\tau_1 \ldots 1/\tau_K\}]$, and then perform Bayesian inference assuming independent priors, that is, $\pi(\boldsymbol{\beta}_0, \boldsymbol{\alpha}, \sigma^2, \boldsymbol{\tau}) = \pi(\boldsymbol{\beta}_0)\pi(\boldsymbol{\alpha})\pi(\sigma^2)\prod_{k=1}^K \pi(\tau_k^2)$ where $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$, $\tau_k^2 \sim G(v_0/2, v_0/2)$, $k = 1, \ldots, K$, $\boldsymbol{\alpha} \sim N(\boldsymbol{a}_0, \boldsymbol{A}_0)$ and $\boldsymbol{\beta}_0 \sim N(\boldsymbol{b}_0, \boldsymbol{B}_0)$. The conditional posterior distributions are $\sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\beta}_{0:T}, \boldsymbol{\alpha}, \boldsymbol{\tau} \sim IG(\alpha_n/2, \delta_n/2)$, where $\delta_n = \sum_{t=1}^T \left[(\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1})^\top\boldsymbol{\Psi}(\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}) + (y_t - \boldsymbol{z}_t^\top\boldsymbol{\alpha} - \boldsymbol{x}_t^\top\boldsymbol{\beta}_t)^\top(y_t - \boldsymbol{z}_t^\top\boldsymbol{\alpha} - \boldsymbol{x}_t^\top\boldsymbol{\beta}_t)\right] + \delta_0$ and $\alpha_n = T(K+1) + \alpha_0$, $\boldsymbol{\tau} = [\tau_1 \ \ldots \ \tau_K]$, $\boldsymbol{\Psi} = \operatorname{diag}\left\{\tau_1^2, \ldots, \tau_K^2\right\}$, and $\tau_k^2 \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\beta}_{0:T}, \sigma^2 \sim G(v_{1n}/2, v_{2kn}/2)$, where $v_{1n} = T + v_0$ and $v_{2kn} = \sigma^{-2}\sum_{t=1}^T(\boldsymbol{\beta}_{t,k} - \boldsymbol{\beta}_{t-1,k})^2 + v_0$, and $\boldsymbol{\alpha} \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{Z}, \sigma^2, \boldsymbol{\beta}_{1:T}, \boldsymbol{\tau} \sim N(\boldsymbol{a}_n, \boldsymbol{A}_n)$, where $\boldsymbol{A}_n = (\boldsymbol{A}_0^{-1} + \sigma^{-2}\sum_{t=1}^T \boldsymbol{z}_t\boldsymbol{z}_t^\top)^{-1}$ and $\boldsymbol{a}_n = \boldsymbol{A}_n(\boldsymbol{A}_0^{-1}\boldsymbol{a}_0 + \sigma^{-2}\sum_{t=1}^T \boldsymbol{z}_t(y_t - \boldsymbol{x}_t^\top\boldsymbol{\beta}_t))$. The vector of the dependent variable is $\boldsymbol{y}$, and all regressors are in $\boldsymbol{X}$ and $\boldsymbol{Z}$.

We can see that all the previous posterior distributions are conditional on the state vector $\boldsymbol{\beta}_{0:T}$, which can be sampled using the *simulation smoother* algorithm, conditional on draws of the time-invariant parameters. Thus, the *state space model* provides an excellent illustration of the modular nature of the Bayesian framework, where performing inference on more complex models often simply involves adding new blocks to an MCMC algorithm. This means we can break down a complex inferential problem into smaller, more manageable parts, which is a "divide and conquer" approach. This is possible due to the structure of the conditional posterior distributions. Exercise 3 asks you to perform a simulation of the model given by Equations 8.3 and 8.4, and to program the MCMC algorithm, including the *simulation smoother*.

8.2 ARMA processes

Since the seminal work of [53], autoregressive moving average (ARMA) models have become ubiquitous in time series analysis. Thus, we present a brief introduction to these models in this section.

Let's start with the linear Gaussian model with autorregresive errors,

$$y_t = \boldsymbol{x}_t^\top \boldsymbol{\beta} + \mu_t \tag{8.5}$$

$$\phi(L)\mu_t = \epsilon_t, \tag{8.6}$$

where $\boldsymbol{x}_t$ is a K-dimensional vector of regressors, $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$, $\phi(L) = 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p$ is a polynomial in the lag operator (L), where $Lz_t = z_{t-1}$, and in general, $L^r z_t = z_{t-r}$.

Thus, we see that stochastic error μ_t follows an *autoregressive process of order* p, that is, $\mu_t \sim AR(p)$. It is standard practice to assume that μ_t is second-order stationary, this implies that the mean, variance and autocovariance of μ_t are finite and independent of t and s, although $\mathbb{E}[\mu_t \mu_s]$ may depend on $|t-s|$. Then, all roots of $\phi(L)$ lie outside the unit circle, for instance, given an $AR(1)$, then, $1 - \phi_1 L = 0$, implies, $L = 1/\phi_1$ such that $|\phi_1| < 0$ for the process being second-order stationary.

The likelihood function conditional on the first p observations is

$$\begin{aligned} p(y_{p+1}, \ldots, y_T \mid y_p, \ldots, y_1, \boldsymbol{\theta}) &= \prod_{t=p+1}^{T} p(y_t \mid \mathcal{I}_{t-1}, \boldsymbol{\theta}) \\ &\propto \sigma^{-(T-p)} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{t=p+1}^{T} (y_t - \hat{y}_{t|t-1,\boldsymbol{\theta}})^2 \right\}, \end{aligned}$$

where $\mathcal{I}_{t-1}$ is the past information, $\boldsymbol{\theta}$ collects all parameters $(\boldsymbol{\beta}, \phi_1, \ldots, \phi_p, \sigma^2)$, and $\hat{y}_{t|t-1,\boldsymbol{\theta}} = (1 - \phi(L))y_t + \phi(L)\boldsymbol{x}^\top \boldsymbol{\beta}$.

We can see that multiplying the first expression in Equation 8.5 by $\phi(L)$, we can express the model as

$$y_t^* = \boldsymbol{x}_t^{*\top} \boldsymbol{\beta} + \epsilon_t \tag{8.7}$$

where $y_t^* = \phi(L)Y_t$ and $\boldsymbol{x}_t^* = \phi(L)\boldsymbol{x}_t$.

Thus, collecting all observations $t = p+1, p+2, \ldots, T$, we have $\boldsymbol{y}^* = \boldsymbol{X}^*\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{T-p})$, $\boldsymbol{y}^*$ is a $T-p$ dimensional vector, and $\boldsymbol{X}^*$ is a $(T-p) \times K$ dimensional matrix.

Assuming that $\boldsymbol{\beta} \mid \sigma \sim N(\boldsymbol{\beta}_0, \sigma^2 \boldsymbol{B}_0)$, $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$ and $\boldsymbol{\phi} \sim N(\boldsymbol{\phi}_0, \boldsymbol{\Phi}_0)\mathbb{1}(\boldsymbol{\phi} \in S_{\boldsymbol{\phi}})$, where $S_{\boldsymbol{\phi}}$ is the stationary region of $\boldsymbol{\phi} = [\phi_1 \ \ldots \ \phi_p]^\top$. Then, Equation 8.7 implies that $\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{\phi}, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \sigma^2 \boldsymbol{B}_n)$, where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^{*\top}\boldsymbol{X}^*)^{-1}$ and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^{*\top}\boldsymbol{y}^*)$. In addition, $\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{\phi}, \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2)$ where $\alpha_n = \alpha_0 + T - p$ and $\delta_n = \delta_0 + (\boldsymbol{y}^* - \boldsymbol{X}^*\boldsymbol{\beta})^\top(\boldsymbol{y}^* - \boldsymbol{X}^*\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)\boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)$. Thus, the previous conditional posterior distributions imply that we can use a Gibbs sampling algorithm to perform inference of these parameters [83].

We know from Equation 8.5 that $\mu_t = y_t - \boldsymbol{x}_t^\top \boldsymbol{\beta}$, from Equation 8.6 that $\mu_t = \phi_1 \mu_{t-1} + \cdots + \phi_p \mu_{t-p} + \epsilon_t$, $t = p+1, \ldots, T$. In matrix notation $\boldsymbol{\mu} = \boldsymbol{U}\boldsymbol{\phi} + \boldsymbol{\epsilon}$, where $\boldsymbol{\mu}$ is a $T-p$ dimensional vector, $\boldsymbol{U}$ is a $(T-p) \times p$ matrix whose t-th

row is $[\mu_{t-1} \ \dots \ \mu_{t-p}]$. Thus, the posterior distribution of $\boldsymbol{\phi} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X}$ is $N(\boldsymbol{\phi}_n, \boldsymbol{\Phi}_n)\mathbb{1}(\boldsymbol{\phi} \in S_{\boldsymbol{\phi}})$, where $\boldsymbol{\Phi}_n = (\boldsymbol{\Phi}_0^{-1} + \sigma^{-2}\boldsymbol{U}^\top\boldsymbol{U})$ and $\boldsymbol{\phi}_n = \boldsymbol{\Phi}_n(\boldsymbol{\Phi}_0^{-1}\boldsymbol{\phi}_0 + \sigma^{-2}\boldsymbol{U}^\top\boldsymbol{\mu})$ (see Exercise 4).

Drawing from the model under the stationarity restriction is straightforward: we simply sample from the multivariate normal distribution and discard draws that do not satisfy the stationarity condition. The proportion of draws that meet this restriction represents the conditional probability that the process is stationary.

Example: Effects of inflation on interest rate II

We specify a *dynamic linear model* in the example of the effects of inflation on interest rates to account for a potential dynamic relationship. However, we can introduce dynamics in this model by assuming

$$\Delta i_t = \beta_1 + \beta_2 \Delta inf_t + \beta_3 \Delta def_t + \mu_t,$$

where $\mu_t = \phi\mu_{t-1} + \epsilon_t$. This leads to the model:

$$\Delta i_t = \beta_1(1-\phi_1) + \phi_1 \Delta i_{t-1} + \beta_2(\Delta inf_t - \phi_1 \Delta inf_{t-1}) + \beta_3(\Delta def_t - \phi_1 \Delta def_{t-1}) + \epsilon_t.$$

Thus, we again use the dataset *16INTDEF.csv* provided by [422, Chaps. 10] to illustrate linear regressions with $AR(1)$ errors.

The following code demonstrates how to implement this application using vague priors, assuming $\alpha_0 = \delta_0 = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}$, $\boldsymbol{B}_0 = \boldsymbol{I}$, $\boldsymbol{\phi}_0 = \mathbf{0}$, and $\boldsymbol{\Phi}_0 = \boldsymbol{I}$. We use 15,000 MCMC iterations, with a burn-in of 5,000 and a thinning parameter of 5.

R code. AR(1) model: Effects of inflation on interest rate

```
rm(list = ls())
set.seed(010101)
DataIntRate <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/16INTDEF.csv", sep = ",", header = TRUE, quote = "")
attach(DataIntRate)
yt <- diff(i3); ytlag <- dplyr::lag(yt, n = 1)
T <- length(yt)
Xt <- cbind(diff(inf), diff(def)); Xtlag <- dplyr::lag(Xt, n = 1)
K <- dim(Xt)[2] + 1
Reg <- lm(yt ~ ytlag + I(Xt[,-1] - Xtlag))
SumReg <- summary(Reg); SumReg
PostSig2 <- function(Beta, Phi){
  Xstar<- matrix(NA, T-1, K - 1)
  ystar <- matrix(NA, T-1, 1)
  for(t in 2:T){
    Xstar[t-1,] <- Xt[t,] - Phi*Xt[t-1,]
    ystar[t-1,] <- yt[t] - Phi*yt[t-1]
  }
  Xstar <- cbind(1, Xstar)
  an <- T - 1 + a0
  dn <- d0 + t(ystar - Xstar%*%Beta)%*%(ystar - Xstar%*%Beta) + t(Beta - b0)%*%B0i%*%(Beta - b0)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2, Phi){
  Xstar<- matrix(NA, T-1, K - 1)
  ystar <- matrix(NA, T-1, 1)
  for(t in 2:T){
    Xstar[t-1,] <- Xt[t,] - Phi*Xt[t-1,]
    ystar[t-1,] <- yt[t] - Phi*yt[t-1]
  }
  Xstar <- cbind(1, Xstar)
  XtXstar <- t(Xstar)%*%Xstar
  Xtystar <- t(Xstar)%*%ystar
  Bn <- solve(B0i + XtXstar)
  bn <- Bn%*%(B0i%*%b0 + Xtystar)
  Beta <- MASS::mvrnorm(1, bn, sig2*Bn)
  return(Beta)
}
PostPhi <- function(sig2, Beta){
  u <- yt - cbind(1,Xt)%*%Beta
  U <- u[-T]
  ustar <- u[-1]
  UtU <- t(U)%*%U
  Utu <- t(U)%*%ustar
  Phin <- solve(Phi0i + sig2^(-1)*UtU)
  phin <- Phin%*%(Phi0i%*%phi0 + sig2^(-1)*Utu)
  Phi <- truncnorm::rtruncnorm(1, a = -1, b = 1, mean = phin, sd = Phin^0.5)
  return(Phi)
}

```

R code. AR(1) model: Effects of inflation on interest rate

```
# Hyperparameters
d0 <- 0.01; a0 <- 0.01
b0 <- rep(0, K); c0 <- 1;
B0 <- c0*diag(K); B0i <- solve(B0)
phi0 <- 0; Phi0 <- 1; Phi0i <- 1/Phi0
# MCMC parameters
mcmc <- 15000
burnin <- 5000
tot <- mcmc + burnin
thin <- 1
PostBetas <- matrix(0, mcmc+burnin, K)
PostSigma2s <- rep(0, mcmc+burnin)
PostPhis <- rep(0, mcmc+burnin)
Beta <- rep(0, K); Phi <- 0
sig2 <- SumReg$sigma^2; Phi <- SumReg$coefficients[2,1]
Beta <- SumReg$coefficients[c(1,3,4),1]
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  sig2 <- PostSig2(Beta = Beta, Phi = Phi)
  PostSigma2s[s] <- sig2
  Beta <- PostBeta(sig2 = sig2, Phi = Phi)
  PostBetas[s,] <- Beta
  Phi <- PostPhi(sig2 = sig2, Beta = Beta)
  PostPhis[s] <- Phi
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
     "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorBetas <- coda::mcmc(PostBetas[keep,])
summary(PosteriorBetas)
PosteriorSigma2 <- coda::mcmc(PostSigma2s[keep])
summary(PosteriorSigma2)
PosteriorPhi <- coda::mcmc(PostPhis[keep])
summary(PosteriorPhi)
dfBinf <- as.data.frame(PosteriorBetas[,2])
# Basic density
library(ggplot2)
p <- ggplot(dfBinf, aes(x=var1)) +
geom_density(color="darkblue", fill="lightblue") +
geom_vline(aes(xintercept=mean(var1)), color="blue",
    linetype="dashed", linewidth=1) +
geom_vline(aes(xintercept=quantile(var1, 0.025)), color="red
    ", linetype="dashed", linewidth=1) +
geom_vline(aes(xintercept=quantile(var1, 0.975)), color="red
    ", linetype="dashed", linewidth=1) +
labs(title="Density effect of inflation on interest rate", x
    ="Effect of inflation", y = "Density")
```

Figure 8.3 shows the posterior density plot of the effects of inflation rate on interest rate. The posterior mean of this coefficient is approximately 0.25, and the credible interval at 95% is (0, 0.46), which indicates again that the annual changes in interest rate are weakly positive related to annual changes in inflation (see Figure 8.2 as reference).

FIGURE 8.3
Density: Effects of inflation on interest rate.

Observe that the previous setting encompasses the particular relevant case $y_t \sim AR(p)$, it is just omitting the covariates such that $y_t = \mu_t$. [85] extend the Bayesian inference of linear regression with $AR(p)$ errors to $ARMA(p,q)$ errors using a *state-space* representation.

Setting $y_t = \mu_t$ such that $y_t = \sum_{s=1}^{p} \phi_j y_{t-s} + \sum_{s=1}^{q} \theta_s \epsilon_{t-s} + \epsilon_t$, letting $r = \max\{p, q+1\}$, $\phi_s = 0$ for $s > p$ and $\theta_s = 0$ for $s > q$, and defining $\boldsymbol{x}^\top = [1\ 0\ \ldots\ 0]$, and $\boldsymbol{H} = [1\ \psi_1\ \ldots\ \psi_{r-1}]^\top$ r-dimensional vectors, and

$$\boldsymbol{G} = \begin{bmatrix} \phi_1 & 1 & 0 & \ldots & 0 \\ \phi_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & & \\ \phi_{r-1} & 0 & 0 & \ldots & 1 \\ \phi_r & 0 & 0 & \ldots & 0 \end{bmatrix} = \begin{bmatrix} \phi_1 & \vdots & & & \\ \phi_2 & \vdots & & \boldsymbol{I}_{r-1} & \\ \vdots & \vdots & & & \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \phi_r & 0 & 0 & \ldots & 0 \end{bmatrix},$$

which is a $r \times r$ dimensional matrix, and give the *state* vector $\boldsymbol{\beta}_t =$

$[\beta_{1,t}\ \beta_{2,t}\ \ldots\ \beta_{r,t}]^\top$, the $ARMA$ model has the following representation:

$$\begin{aligned} y_t &= \boldsymbol{x}^\top \boldsymbol{\beta}_t \\ \boldsymbol{\beta}_t &= \boldsymbol{G}\boldsymbol{\beta}_{t-1} + \boldsymbol{H}\epsilon_t. \end{aligned}$$

This is a *dynamic linear model* where $\boldsymbol{\Sigma}_t = 0$, and $\boldsymbol{\Omega}_t = \sigma^2 \boldsymbol{H}\boldsymbol{H}^\top$ (see [309, 85]).

A notable advantage of the *state-space* representation of the $ARMA$ model is that the evaluation of the likelihood can be performed efficiently using the recursive laws. Extensions to autoregressive integrated moving average $ARIMA(p, d, q)$ models are discussed in [309, Chap. 3]. In $ARIMA(p, d, q)$ models, d refers to the level of integration (or differencing) required to eliminate the stochastic trend in a time series (see [126, Chap. 4] for details).

Example: $AR(2)$ **process**

Let's see the *state-space* representation of a stationary $AR(2)$ process with intercept, that is, $y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t$, where $\epsilon_t \sim N(0, \sigma^2)$. Thus, $\mathbb{E}[y_t] = \frac{\mu}{1-\phi_1-\phi_2}$, and variance $Var[y_t] = \frac{\sigma^2(1-\phi_2)}{1-\phi_2-\phi_1^2-\phi_1^2\phi_2-\phi_2^2+\phi_2^3}$.

In addition, we can proof that setting $z_t = Y_t - \bar{\mu}$, we have $z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \epsilon_t$ where $\mathbb{E}[z_t] = 0$, and these are equivalent representations (see Exercise 5). Then, setting $\boldsymbol{x}^\top = [1\ 0]$, $\boldsymbol{H} = [1\ 0]^\top$, $\boldsymbol{G} = \begin{bmatrix} \phi_1 & 1 \\ \phi_2 & 0 \end{bmatrix}$, $\boldsymbol{\beta}_t = [\beta_{t1}\ \beta_{t2}]^\top$, $\boldsymbol{\Sigma}_t = 0$ and $\boldsymbol{\Omega}_t = \sigma^2$ we have

$$\begin{aligned} z_t &= \boldsymbol{x}^\top \boldsymbol{\beta}_t && \text{(Observation equation)} \\ \boldsymbol{\beta}_t &= \boldsymbol{G}\boldsymbol{\beta}_{t-1} + \boldsymbol{H}\epsilon_t && \text{(States equations).} \end{aligned}$$

We use the function *stan_sarima* from the package *bayesforecast* to perform Bayesian inference in $ARMA$ models in our GUI. The following code shows how to simulate an $AR(2)$ process, and perform Bayesian inference using this function.

R code. Simulation and inference: AR(2) model

```
rm(list = ls()); set.seed(010101)
T <- 200; mu <- 0.5
phi1 <- 0.5; phi2 <- 0.3; sig <- 0.5
Ey <- mu/(1-phi1-phi2); Sigy <- sig*((1-phi2)/(1-phi2-phi1^2-phi2*phi1^2-phi2^2+phi2^3))^0.5
y <- rnorm(T, mean = Ey, sd = Sigy)
e <- rnorm(T, mean = 0, sd = sig)
for(t in 3:T){
  y[t] <- mu + phi1*y[t-1] + phi2*y[t-2] + e[t]
}
mean(y); sd(y)
y <- ts(y, start=c(1820, 1), frequency=1)
plot(y)
iter <- 10000; burnin <- 5000; thin <- 1; tot <- iter + burnin
library(bayesforecast)
sf1 <- bayesforecast::stan_sarima(y, order = c(2, 0, 0), prior_mu0 = normal(0, 1),
prior_ar = normal(0, 1), prior_sigma0 = inverse.gamma(0.01/2, 0.01/2),
seasonal = c(0, 0, 0), iter = tot, warmup = burnin, chains = 1)
keep <- seq(burnin+1, tot, thin)
Postmu <- sf1[["stanfit"]]@sim[["samples"]][[1]][["mu0"]][keep]
Postsig <- sf1[["stanfit"]]@sim[["samples"]][[1]][["sigma0"]][keep]
Postphi1 <- sf1[["stanfit"]]@sim[["samples"]][[1]][["ar0[1]"]][keep]
Postphi2 <- sf1[["stanfit"]]@sim[["samples"]][[1]][["ar0[2]"]][keep]
Postdraws <- coda::mcmc(cbind(Postmu, Postsig, Postphi1, Postphi2))
summary(Postdraws)
Quantiles for each variable:
               2.5%    25%    50%    75%  97.5%
Postmu   0.39914 0.5732 0.6625 0.7518 0.9346
Postsig  0.47696 0.5071 0.5248 0.5439 0.5829
Postphi1 0.42384 0.5159 0.5634 0.6089 0.6979
Postphi2 0.06034 0.1456 0.1920 0.2361 0.3286
plot(Postdraws)
```

We perform 10000 MCMC iterations plus a burn-in equal 5000 assuming $\sigma^2 \sim IG(0.01/2, 0.01/2)$, $\mu \sim N(0,1)$ and $\phi_k \sim N(0,1)$, $k = 1, 2$. The trace plots look well, and all 95% credible intervals encompass the population values.

Algorithm A22 shows how to perform inference in $ARMA(p,q)$ models using our GUI. See also Chapter 5 for details regarding the dataset structure.

Algorithm A22 Autoregressive Moving Average ($ARMA$) models

1: Select *Time series Model* on the top panel
2: Select $ARMA$ using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Set the order of the ARMA model, p and q parameters
6: Set the frequency: annual (1), quarterly (4), monthly (12), etc
7: Set the location and scale hyperparameters of the *intercept*, autoregressive (AR), moving average (MA) and standard deviation. Take into account that there is just one set of hyperparameters for AR and MA coefficients. This step is not necessary as by default our GUI uses non-informative priors
8: Click the *Go!* button
9: Analyze results
10: Download posterior chains and figures (density, autocorrelation and trace plots) using the *Download Results* button

The function *stan_sarima* uses the *Stan* software [389], which in turn employs *Hamiltonian Monte Carlo* (HMC). The following code illustrates how to perform Bayesian inference in the $AR(2)$ model by programming the HMC from scratch. It is important to note that this is only an illustration, as HMC is less efficient than the Gibbs sampler in this example. However, HMC can outperform traditional MCMC algorithms in more complex models, particularly when dealing with high-dimensional probability distributions or when MCMC struggles with poor mixing due to posterior correlation.

In the first block, we perform the simulation by setting $\mu = 0.5$, $\phi_1 = 0.5$, $\phi_2 = 0.3$, $\sigma = 0.25$, and a sample size of 200. We then set the hyperparameters and define the function to calculate the logarithm of the posterior distribution. The model is parametrized using $\tau = \log(\sigma^2)$, such that $\sigma^2 = \exp(\tau)$, which avoids issues related to the non-negativity restriction of σ^2. As a result, we need to account for the Jacobian due to this transformation, specifically $d\sigma^2/d\tau = \exp(\tau)$.

Next, we define the function to compute the gradient vector of the log posterior distribution. It is preferable to calculate the gradient vector analytically, as using finite differences can be computationally expensive. However, it is a good practice to check the analytical calculations by evaluating the function at the maximum posterior estimate, where the function should return values close to 0, or by comparing the results with finite differences at a few evaluation points.

The posterior distribution is given by[3]

$$\pi(\mu, \phi_1, \phi_2, \tau \mid \boldsymbol{y}) \propto \prod_{t=3}^{T} (\exp(\tau))^{-1/2} \exp\left\{-\frac{1}{2\exp(\tau)}(y_t - \mu - \phi_1 y_{t-1} - \phi_2 y_{t-2})^2\right\}$$
$$\times \exp\left\{-\frac{1}{2\sigma_\mu^2}(\mu - \mu_0)^2\right\} \times \exp\left\{-\frac{1}{2\sigma_{\phi_1}^2}(\phi_1 - \phi_{10})^2\right\}$$
$$\times \exp\left\{-\frac{1}{2\sigma_{\phi_2}^2}(\phi_2 - \phi_{20})^2\right\} \times \exp\left\{-(\alpha_0/2 + 1)\tau\right\} \exp\left\{-\delta_0/(2\exp(\tau))\right\} \exp(\tau).$$

The components of the gradient vector of the log posterior distribution are given by

$$\frac{\partial \log(\pi(\mu, \phi_1, \phi_2, \tau \mid \boldsymbol{y}))}{\partial \mu} = \frac{\sum_{t=3}^{T}(y_t - \mu - \phi_1 y_{t-1} - \phi_2 y_{t-2})}{\exp(\tau)} - \frac{1}{\sigma_\mu^2}(\mu - \mu_0)$$
$$\frac{\partial \log(\pi(\mu, \phi_1, \phi_2, \tau \mid \boldsymbol{y}))}{\partial \phi_1} = \frac{\sum_{t=3}^{T}(y_t - \mu - \phi_1 y_{t-1} - \phi_2 y_{t-2}) y_{t-1}}{\exp(\tau)} - \frac{1}{\sigma_{\phi_1}^2}(\phi_1 - \phi_{10})$$
$$\frac{\partial \log(\pi(\mu, \phi_1, \phi_2, \tau \mid \boldsymbol{y}))}{\partial \phi_2} = \frac{\sum_{t=3}^{T}(y_t - \mu - \phi_1 y_{t-1} - \phi_2 y_{t-2}) y_{t-2}}{\exp(\tau)} - \frac{1}{\sigma_{\phi_2}^2}(\phi_2 - \phi_{20})$$
$$\frac{\partial \log(\pi(\mu, \phi_1, \phi_2, \tau \mid \boldsymbol{y}))}{\partial \tau} = -\frac{(T-2)}{2} + \frac{\sum_{t=3}^{T}(y_t - \mu - \phi_1 y_{t-1} - \phi_2 y_{t-2})^2}{2\exp(\tau)}$$
$$- (\alpha_0/2 + 1) + \delta_0/(2\exp(\tau)) + 1.$$

Next, we provide the code for the Hamiltonian Monte Carlo, as outlined in Chapter 4. The initial values are set as follows: $\mu = \bar{y} = \frac{1}{T-2}\sum_{t=3}^{T} y_t$, $\phi_1 = \phi_2 = 0$, and $\tau = \exp\left(\frac{1}{T-2}\sum_{t=3}^{T}(y_t - \bar{y})^2\right)$, with M being the inverse covariance matrix of the posterior distribution evaluated at its maximum. Additionally, ϵ is randomly drawn from a uniform distribution between 0 and $2\epsilon_0$, and L is set to the highest integer near $1/\epsilon$, in order to approximately satisfy $L\epsilon = 1$, where $\epsilon_0 = 0.1$.

We can verify that all 95% credible intervals encompass the population values, and the posterior means are close to the population values. The acceptance rate averages above 65%, so we should consider increasing the base step (ϵ_0). Furthermore, we do not impose the stationarity conditions on ϕ_1 and ϕ_2. Exercise 6 asks to program an HMC that takes these requirements into account.

[3]Take into account that we do not consider the first two observations when present the likelihood, this is no an issue when there is a large sample size.

R code. Simulation and inference: AR(2) model using Hamiltonian Monte Carlo

```
# Simulation AR(2)
rm(list = ls()); set.seed(010101); T <- 1000; K <- 4
mu <- 0.5; phi1 <- 0.5; phi2 <- 0.3; sig <- 0.5
Ey <- mu/(1-phi1-phi2); Sigy <- sig*((1-phi2)/(1-phi2-phi1^2-phi2*phi1^2-phi2^2+phi2^3))^0.5
y <- rnorm(T, mean = Ey, sd = Sigy); e <- rnorm(T, mean = 0, sd = sig)
for(t in 3:T){
  y[t] <- mu + phi1*y[t-1] + phi2*y[t-2] + e[t]
}
# Hyperparameters
d0 <- 0.01; a0 <- 0.01; mu0 <- 0; MU0 <- 1
phi0 <- c(0, 0); Phi0 <- diag(2)
# Log posterior multiply by -1 to use optim
LogPost <- function(theta, y){
  mu <- theta[1]; phi1 <- theta[2]; phi2 <- theta[3]
  tau <- theta[4]; sig2 <- exp(tau); logLik <- NULL
  for(t in 3:T){
    logLikt <- dnorm(y[t], mean = mu + phi1*y[t-1] + phi2*y[t-2], sd = sig2^0.5, log = TRUE)
    logLik <- c(logLik, logLikt)
  }
  logLik <- sum(logLik)
  logPrior <- dnorm(mu, mean = mu0, sd = MU0^0.5, log = TRUE) + dnorm(phi1, mean = phi0[1], sd = Phi0[1,1]^0.5, log = TRUE) + dnorm(phi2, mean = phi0[2], sd = Phi0[2,2]^0.5, log = TRUE) + invgamma::dinvgamma(sig2, shape = a0/2, rate = d0/2, log = TRUE)
  logPosterior <- logLik + logPrior + tau
  return(-logPosterior) # Multiply by -1 to minimize using optim
}
theta0 <- c(mean(y), 0, 0, var(y))
Opt <- optim(theta0, LogPost, y = y, hessian = TRUE)
theta0 <- Opt$par; VarPost <- solve(Opt$hessian)
# Gradient log posterior
GradientTheta <- function(theta, y){
  mu <- theta[1]; phi1 <- theta[2]; phi2 <- theta[3]
  tau <- theta[4]; sig2 <- exp(tau); SumLik <- matrix(0, 3, 1)
  SumLik2 <- NULL
  for(t in 3:T){
    xt <- matrix(c(1, y[t-1], y[t-2]), 3, 1)
    SumLikt <- (y[t] - (mu + phi1*y[t-1] + phi2*y[t-2]))*xt
    SumLik2t <- (y[t] - (mu + phi1*y[t-1] + phi2*y[t-2]))^2
    SumLik <- rowSums(cbind(SumLik, SumLikt))
    SumLik2 <- sum(SumLik2, SumLik2t)
  }
  Grad_mu <- SumLik[1]/sig2 - (1/MU0)*(mu - mu0)
  Grad_phi1 <- SumLik[2]/exp(tau) - 1/Phi0[1,1]*(phi1 - phi0[1])
  Grad_phi2 <- SumLik[3]/exp(tau) - 1/Phi0[2,2]*(phi2 - phi0[2])
  Grad_tau <- -(T-2)/2 + SumLik2/(2*exp(tau)) - (a0/2 + 1) + d0/(2*exp(tau)) + 1
  Grad <- c(Grad_mu, Grad_phi1, Grad_phi2, Grad_tau)
  return(Grad)
}
```

R code. Simulation and inference: AR(2) model using Hamiltonian Monte Carlo

```
# Hamiltonian Monte Carlo function
HMC <- function(theta, y, epsilon, M){
  L <- ceiling(1/epsilon)
  Minv <- solve(M); thetat <- theta
  K <- length(thetat)
  mom <- t(mvtnorm::rmvnorm(1, rep(0, K), M))
  logPost_Mom_t <- -LogPost(thetat, y) +  mvtnorm::dmvnorm(t
    (mom), rep(0, K), M, log = TRUE)
  for(l in 1:L){
    if(l == 1 | l == L){
      mom <- mom + 0.5*epsilon*GradientTheta(theta, y)
      theta <- theta + epsilon*Minv%*%mom
    }else{
      mom <- mom + epsilon*GradientTheta(theta, y)
      theta <- theta + epsilon*Minv%*%mom
    }
  }
  logPost_Mom_star <- -LogPost(theta, y) +  mvtnorm::dmvnorm
    (t(mom), rep(0, K), M, log = TRUE)
  alpha <- min(1, exp(logPost_Mom_star-logPost_Mom_t))
  u <- runif(1)
  if(u <= alpha){
    thetaNew <- c(theta)
  }else{
    thetaNew <- thetat
  }
  rest <- list(theta = thetaNew, Prob = alpha)
  return(rest)
}
# Posterior draws
S <- 1000; burnin <- 1000; thin <- 2; tot <- S + burnin
thetaPost <- matrix(NA, tot, K)
ProbAccept <- rep(NA, tot)
theta0 <- c(mean(y), 0, 0, exp(var(y)))
M <- solve(VarPost); epsilon0 <- 0.1
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  epsilon <- runif(1, 0, 2*epsilon0)
  L <- ceiling(1/epsilon)
  HMCs <- HMC(theta = theta0, y, epsilon, M)
  theta0 <- HMCs$theta
  thetaPost[s,] <- HMCs$theta
  ProbAccept[s] <- HMCs$Prob
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
     "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
thetaF <- coda::mcmc(thetaPost[keep,])
summary(thetaF)
summary(exp(thetaF[,K]))
ProbAcceptF <- coda::mcmc(ProbAccept[keep])
summary(ProbAcceptF)
```

8.3 Stochastic volatility models

A notable example of non-linear and non-Gaussian *state-space models* is stochastic volatility models (SVMs), which are widely used to model the volatility of financial returns. SVMs have gained significant attention due to their flexibility, ability to capture complex dynamics such as asymmetries, and ease of generalization to simultaneously model multiple returns, making them advantageous over generalized autoregressive conditional heteroskedasticity (GARCH) models proposed by [49]. However, estimating SVMs is more challenging than estimating GARCH models. This is because GARCH models set variance in a deterministic manner, whereas SVMs do so stochastically. Consequently, GARCH models are typically estimated using maximum likelihood methods, while SVMs require Bayesian approaches, adding complexity to the estimation process.

The specification of the stochastic volatility model is given by

$$y_t = \boldsymbol{x}_t^\top \boldsymbol{\beta} + \exp\{0.5 h_t\}\, \mu_t \qquad \text{(Observation equation)} \tag{8.8}$$

$$h_t = \mu + \phi(h_{t-1} - \mu) + \sigma w_t \qquad \text{(State equation)}, \tag{8.9}$$

where y_t are the log-returns, $\boldsymbol{x}_t$ are controls, $\boldsymbol{\beta}$ are time-invariant location parameters, $\mu_t \sim N(0,1)$, $w_t \sim N(0,1)$, $\mu_t \perp w_t$, the initial log-variance process $h_0 \sim N(\mu, \sigma^2/(1-\phi^2))$, μ, ϕ and σ are the level, persistence and standard deviation of the log-variance, respectively.

Given the specification in Equations 8.8 and 8.9, we can write the observation equation as

$$\log\left\{(y_t - \boldsymbol{x}_t^\top \boldsymbol{\beta})^2\right\} = h_t + \log(\mu_t^2),$$

which leads to a linear, but non-Gaussian, *state-space model*. [227] approximate the distribution of $\log(\mu_t^2)$ by a mixture of normal distributions, that is,

$$\log(\mu_t^2) \mid l_t \sim N(m_{l_t}, s_{l_t}^2),$$

where $l_t \in \{1, 2, \ldots, 10\}$ defines the mixture component indicator at time t. Thus, the model can be written as

$$\log\left\{(y_t - \boldsymbol{x}_t^\top \boldsymbol{\beta})^2\right\} = h_t + \log(\mu_t^2),$$

and

$$h_t = \mu + \phi(h_{t-1} - \mu) + \sigma w_t.$$

This forms a linear and conditionally Gaussian *state-space model*, where

$$\log\left\{(y_t - \boldsymbol{x}_t^\top \boldsymbol{\beta})^2\right\} = m_{l_t} + h_t + \mu_t^2,$$

and

$$\mu_t \sim N(0, s_{l_t}^2).$$

We use the *stochvol* package in our GUI to perform MCMC inference in the SVMs [197]; this package is based on the MCMC algorithms proposed by [227]. The default prior distributions in the *stochvol* package are:

$$\boldsymbol{\beta} \sim N(\boldsymbol{b}_0, \boldsymbol{B}_0), \quad \mu \sim N(\mu_0, \sigma^2_{\mu 0}), \quad \frac{\phi+1}{2} \sim B(\alpha_0, \beta_0), \quad \sigma^2 \sim G\left(\frac{1}{2}, \frac{1}{2\sigma^2_{\sigma^2}}\right).$$

The prior distribution for ϕ is set to ensure stationarity of the process ($\phi \in (-1, 1)$). In most applications, $\phi \approx 1$, so the authors of the package recommend setting $\alpha_0 \gtrsim 5$ and $\beta_0 \approx 1.5$. The prior distribution for σ is $|N(0, \sigma^2_{\sigma^2})|$ (a half-normal distribution). This is recommended by the authors since the conjugate inverse-gamma distribution does not work well in this case, as it bounds σ away from 0, which is undesirable when modeling the log-variance of log-returns.

Algorithm A23 shows how to perform inference in stochastic volatility models using our GUI. See also Chapter 5 for details regarding the dataset structure.

Algorithm A23 Stochastic volatility models

1: Select *Time series Model* on the top panel
2: Select *Stochastic volatility* using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Set the hyperparameters: the mean and standard deviation of the Gaussian prior for the regression parameters, mean and standard deviation for the Gaussian prior distribution of the level of the log-volatility, shape parameters for the Beta prior distribution of the transformed persistence parameter, and the positive real number, which stands for the scaling of the transformed volatility of log-volatility. This step is not necessary as by default our GUI uses default values in *stochvol* package
6: Click the *Go!* button
7: Analyze results
8: Download posterior chains of the fixed coefficients, and the states using the *Download Results* button

Example: Simulation exercise of the stochastic volatility model

The following code shows how to simulate and perform Bayesian inference in the stochastic volatility model using the function *svsample* from the *stochvol* package. We set the stochastic volatility parameters to $\mu = -10$, $\phi = 0.95$, and $\sigma = 0.3$. We assume two regressors, which are distributed as standard normal, with $\boldsymbol{\beta} = [0.5\ 0.3]^\top$, and the sample size is 1250, which corresponds to approximately 5 years of daily returns. We use the default hyperparameters: 10000 MCMC iterations, a burn-in of 5000, and a thinning parameter of 5.

The summary statistics of the posterior draws show that all 95% credible intervals encompass the population parameters, and the posterior chains appear to have converged. Figure 8.4 displays the posterior results for the volatility (h_t). The posterior mean (blue) follows the "observed" series (black), and the 95% credible intervals (light blue) typically encompass the "observed" series.

R code. Simulation and inference: Stochastic volatility model

```r
rm(list = ls()); set.seed(010101)
T <- 1250; K <- 2
X <- matrix(rnorm(T*K), T, K)
B <- c(0.5, 0.3); mu <- -10; phi <- 0.95; sigma <- 0.3
h <- numeric(T); y <- numeric(T)
h[1] <- rnorm(1, mu, sigma / sqrt(1 - phi^2))  # Initial state
y[1] <- X[1,]%*%B + rnorm(1, 0, exp(h[1] / 2))           # Initial observation
for (t in 2:T) {
  h[t] <- mu + phi*(h[t-1]-mu) + rnorm(1, 0, sigma)
  y[t] <- X[t,]%*%B + rnorm(1, 0, sd = exp(0.5*h[t]))
}
df <- as.data.frame(cbind(y, X))
colnames(df) <- c("y", "x1", "x2")
MCMC <- 10000; burnin <- 10000; thin <- 5
res <- stochvol::svsample(y, designmatrix = X, draws = MCMC, burnin = burnin, thin = thin, priormu = c(0, 100), priorsigma = c(1), priorphi = c(5, 1.5), priorbeta =  c(0, 10000))
summary(res[["para"]][[1]][,-c(4,5)])
summary(res[["beta"]])
ht <- res[["latent"]][[1]]
library(dplyr)
library(ggplot2)
library(latex2exp)
ggplot2::theme_set(theme_bw())
x_means <- colMeans(ht)
x_quantiles <- apply(ht, 2, function(x) quantile(x, probs = c(0.025, 0.975)))
df <- tibble(t = seq(1, T), mean = x_means, lower = x_quantiles[1, ], upper = x_quantiles[2, ], x_true = h, observations = y)
plot_filtering_estimates <- function(df) {
  p <- ggplot(data = df, aes(x = t)) + geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 1, fill = "lightblue") + geom_line(aes(y = x_true), colour = "black", alpha = 1, linewidth = 0.5) + geom_line(aes(y = mean), colour = "blue", linewidth = 0.5) +  ylab(TeX("$h_{t}$")) + xlab("Time")
  print(p)
}
plot_filtering_estimates(df)
```

So far, we have used MCMC algorithms to perform inference in *state-space models*. These algorithms require all observations to estimate the unknown

FIGURE 8.4
Stochastic volatility model.

parameters, a process referred to as offline or batch inference. However, this approach has limitations when online inference is needed, as every new observation requires simulating a new posterior chain. This is because MCMC algorithms do not naturally adapt to sequential updates. In contrast, particle filter algorithms, which are a subset of sequential Monte Carlo (SMC) methods, are specifically designed for sequential use, making them suitable for online inference.

Remember from Chapter 4 that particle filters (sequential Monte Carlo) are algorithms that allow computing a numerical approximation to the filtering distribution $\pi(\boldsymbol{\theta}_{1:t} \mid \boldsymbol{y}_{1:t})$ sequentially in time. This is particularly relevant in non-linear and non-Gaussian models where there is no analytical solution for the filtering distribution.

The following code shows how to perform particle filtering in the vanilla stochastic volatility model assuming that the proposal distribution is the conditional prior distribution, that is, $q(h_t \mid h_{t-1}, y_t) = \pi(h_t \mid h_{t-1})$, which is normal with mean $\mu + \phi(h_{t-1} - \mu)$ and variance σ^2. This choice implies that the incremental importance weights are equal to $p(y_t \mid h_t)$, which is $N(0, \exp(h_t))$. Therefore, the weights are proportional to the likelihood function. We perform multinomial resampling every time period in the code, and start the algorithm in the stationary distribution of h_t. Remember that there are other resampling approaches that are more efficient, for instance, residual resampling (see Section 4.3). We ask in Exercise 7 to modify this code to perform resampling when the effective sample size is lower than 50% of the initial number of particles. In addition, we ask to program a sequential importance

sampling, and check why is important to perform resampling in this simple example.

Figure 8.5 illustrates the filtering recursion using SMC with uneven weights (blue line), even weights (purple line), bands corresponding to plus/minus two standard deviations (light blue shaded area), and the true state (black line).[4] The results indicate that SMC performs well even with a simple implementation, with no significant differences between using even and uneven weights (see Chapter 4).

In this example, we use the population parameters to perform the filtering recursion. However, this is not the case in practice, as we must estimate the time-invariant parameters. Therefore, more elaborate algorithms are required to achieve this (see Chapter 4). For instance, [12] propose particle Markov chain Monte Carlo, a family of methods that combines MCMC and SMC. See [101] for a tutorial on particle Metropolis-Hastings in **R**. A potential practical solution for applications that require sequential updating of a posterior distribution over an unbounded time horizon is to estimate the time-invariant parameters offline using MCMC algorithms up to a specific time period, and then update the state vector sequentially online during subsequent time periods, iterating this process. This is not optimal, but it can be practical.

[4]This standard deviation estimates the conditional posterior's standard deviation derived from the particles, not the estimator's standard deviation. The latter requires several independent particle runs on the same data.

R code. Simulation and inference: Stochastic volatility model programming sequential Monte Carlo from scratch

```r
rm(list = ls()); set.seed(010101)
T <- 1250; mu <- -10; phi <- 0.95; sigma <- 0.3
h <- numeric(T); y <- numeric(T)
h[1] <- rnorm(1, mu, sigma / sqrt(1 - phi^2))
y[1] <- rnorm(1, 0, exp(h[1] / 2))
for (t in 2:T) {
  h[t] <- mu + phi*(h[t-1]-mu) + rnorm(1, 0, sigma)
  y[t] <- rnorm(1, 0, sd = exp(0.5*h[t]))
}
N <- 10000
log_Weights <- matrix(NA, N, T)  # Log weights
Weights <- matrix(NA, N, T)  # Weights
WeightsST <- matrix(NA, N, T)  # Normalized weights
WeightsSTT <- matrix(1/N, N, T)  # Normalized weights bar
particles <- matrix(NA, N, T)   # Particles
particlesT <- matrix(NA, N, T)   # Particles bar
logalphas <- matrix(NA, N, T)   # Incremental importance
particles[, 1] <- rnorm(N, mu, sigma / sqrt(1 - phi^2))  # Stationary prior
log_Weights[, 1] <- dnorm(y[1], 0, sd = exp(0.5*particles[,1]), log = TRUE)  # Likelihood
Weights[, 1] <- exp(log_Weights[, 1])
WeightsST[, 1] <- Weights[, 1] / sum(Weights[, 1])
ind <- sample(1:N, size = N, replace = TRUE, prob = WeightsST[, 1]) # Resample
particles[, 1] <- particles[ind, 1] # Resampled particles
particlesT[, 1] <- particles[, 1] # Resampled particles
WeightsST[, 1] <- rep(1/N, N) # Resampled weights
pb <- winProgressBar(title = "progress bar", min = 0, max = T, width = 300)
for (t in 2:T) {
  particles[, t] <- rnorm(N, mu + phi*(particles[, t - 1] - mu), sigma)  # Sample from proposal
  logalphas[, t] <- dnorm(y[t], 0, sd = exp(0.5*particles[,t]), log = TRUE)
  Weights[, t] <- exp(logalphas[, t])
  WeightsST[, t] <- Weights[, t] / sum(Weights[, t])
  if(t < T){
    ind <- sample(1:N, size = N, replace = TRUE, prob = WeightsST[, t])
    particles[, 1:t] <- particles[ind, 1:t]
  }else{
    ind <- sample(1:N, size = N, replace = TRUE, prob = WeightsST[, t])
    particlesT[, 1:t] <- particles[ind, 1:t]
  }
  setWinProgressBar(pb, t, title=paste( round(t/T*100, 0), " % done"))
}
close(pb)
```

R code. Simulation and inference: Stochastic volatility model programming sequential Monte Carlo from scratch

```
FilterDist <- colSums(particles * WeightsST)
SDFilterDist <- (colSums(particles^2 * WeightsST) -
    FilterDist^2)^0.5
FilterDistT <- colSums(particlesT * WeightsSTT)
SDFilterDistT <- (colSums(particlesT^2 * WeightsSTT) -
    FilterDistT^2)^0.5
MargLik <- colMeans(Weights)
plot(MargLik, type = "l")
library(dplyr)
library(ggplot2)
require(latex2exp)
ggplot2::theme_set(theme_bw())
Tfig <- 250
keepFig <- 1:Tfig
df <- tibble(t = keepFig,
mean = FilterDist[keepFig],
lower = FilterDist[keepFig] - 2*SDFilterDist[keepFig],
upper = FilterDist[keepFig] + 2*SDFilterDist[keepFig],
meanT = FilterDistT[keepFig],
lowerT = FilterDistT[keepFig] - 2*SDFilterDistT[keepFig],
upperT = FilterDistT[keepFig] + 2*SDFilterDistT[keepFig],
x_true = h[keepFig])
plot_filtering_estimates <- function(df) {
  p <- ggplot(data = df, aes(x = t)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 1,
  fill = "lightblue") +
  geom_line(aes(y = x_true), colour = "black", alpha = 1,
  linewidth = 0.5) +
  geom_line(aes(y = mean), colour = "blue", linewidth = 0.5)
     +
  geom_line(aes(y = meanT), colour = "purple", linewidth =
    0.5) +
  ylab(TeX("$h_{t}$")) + xlab("Time")
  print(p)
}
plot_filtering_estimates(df)
```

FIGURE 8.5
Stochastic volatility model: Sequential Monte Carlo (SMC).

8.4 Vector Autoregressive models

Another widely used methodological approach in time series analysis is the vector autoregressive (VAR) model, which extends AR(p) models to the multivariate case. Since the seminal work by Sims (1980) [378], these models have become a cornerstone of macroeconomic research to perform forecasts, and impulse-response (structural) analysis. This chapter provides an introduction to Bayesian inference in VAR models, with detailed discussions available in [233, 108, 423, 71].

The *reduced-form* VAR(p) model can be written as

$$\boldsymbol{y}_t = \boldsymbol{v} + \sum_{j=1}^{p} \boldsymbol{A}_j \boldsymbol{y}_{t-j} + \boldsymbol{\mu}_t, \tag{8.10}$$

where $\boldsymbol{y}_t$ is a M-dimensional vector having information of M time series variables, $\boldsymbol{v}$ is a M-dimensional vector of intercepts, $\boldsymbol{A}_j$ are $M \times M$ matrices of coefficients, and $\boldsymbol{\mu}_t \overset{iid}{\sim} N_M(\mathbf{0}, \boldsymbol{\Sigma})$ are stochastic errors, $t = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, p$. Other deterministic terms and exogenous variables can be added to the specification without main difficulty, we do not do this to keep simply the notation. In addition, we assume that the stability condition is satisfied such that the stochastic process is stationary (see [189, Chap. 2] for details), and we have available p presample values for each variable.

Following the matrix-form notation of the multivariate regression model (see sections 3.4 and 7.1), we can set $\boldsymbol{Y} = [\boldsymbol{y_1}\ \boldsymbol{y_2}\ \ldots\ \boldsymbol{y_M}]$, which is an $T \times M$ matrix, $\boldsymbol{x}_t = [1\ \boldsymbol{y}_{t-1}^\top\ \ldots\ \boldsymbol{y}_{t-p}^\top]$ is a $(1+Mp)$-dimensional row vector, we define $k = 1 + Mp$ to facilitate notation, and set

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix},$$

which is a $T \times k$ matrix, $\boldsymbol{B} = [\boldsymbol{v}\ \boldsymbol{A}_1\ \boldsymbol{A}_2 \ldots \boldsymbol{A}_P]^\top$ is a $k \times M$ matrix of parameters, and $\boldsymbol{U} = [\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_M]$ is a $T \times M$-dimensional matrix of stochastic random errors such that $\boldsymbol{U} \sim N_{T\times M}(\boldsymbol{0}_{T\times M}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_T)$. Thus, we can express the VAR(p) model in the form of a multivariate regression model,

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{U}.$$

We can assume conjugate priors to facilitate computation, that is,

$$\pi(\boldsymbol{B}, \boldsymbol{\Sigma}) = \pi(\boldsymbol{B} \mid \boldsymbol{\Sigma})\pi(\boldsymbol{\Sigma}),$$

where $\boldsymbol{B} \mid \boldsymbol{\Sigma} \sim N_{k\times M}(\boldsymbol{B}_0, \boldsymbol{V}_0, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \sim IW(\boldsymbol{\Psi}_0, \alpha_0)$. Thus,

$$\pi(\boldsymbol{B}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X}) = \pi(\boldsymbol{B} \mid \boldsymbol{\Sigma}, \boldsymbol{Y}, \boldsymbol{X})\pi(\boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X}),$$

where $\boldsymbol{B} \mid \boldsymbol{\Sigma}, \boldsymbol{Y}, \boldsymbol{X} \sim N_{k\times M}(\boldsymbol{B}_n, \boldsymbol{V}_n, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \mid \boldsymbol{Y}, \boldsymbol{X} \sim IW(\boldsymbol{\Psi}_n, \alpha_n)$. The quantities $\boldsymbol{B}_n$, $\boldsymbol{V}_n$, $\boldsymbol{\Psi}_n$, and α_n are given by the following expressions:

$$\begin{gathered}
\boldsymbol{B}_n = (\boldsymbol{V}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \boldsymbol{X}^\top\boldsymbol{X}\widehat{\boldsymbol{B}}), \\
\boldsymbol{V}_n = (\boldsymbol{V}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}, \\
\boldsymbol{\Psi}_n = \boldsymbol{\Psi}_0 + \boldsymbol{S} + \boldsymbol{B}_0^\top\boldsymbol{V}_0^{-1}\boldsymbol{B}_0 + \widehat{\boldsymbol{B}}^\top\boldsymbol{X}^\top\boldsymbol{X}\widehat{\boldsymbol{B}} - \boldsymbol{B}_n^\top\boldsymbol{V}_n^{-1}\boldsymbol{B}_n, \\
\boldsymbol{S} = (\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{B}})^\top(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{B}}), \\
\widehat{\boldsymbol{B}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{Y},
\end{gathered}$$

and

$$\alpha_n = T + \alpha_0.$$

Thus, we see that once we express a VAR(p) model in the correct form, we can perform Bayesian inference as we did in the multivariate regression model. However, assuming conjugate priors has some limitations. First, VAR(p) models have many parameters. For instance, with 4 lags and 6 variables, we would have 150 location parameters $((1 + (6 \times 4)) \times 6)$ and 21 scale parameters $(6 \times (6 + 1)/2)$ for the covariance matrix. This can lead to a loss of precision, especially when using macroeconomic data, due to the typical lack of

large sample sizes. Therefore, it is desirable to impose prior restrictions on the model specification, which cannot be achieved using conjugate priors.

Second, natural conjugate priors do not allow for flexible extensions, such as having different regressors in different equations. Third, the prior structure implies that the prior covariance of the coefficients in any two equations must be proportional to each other. This is because the prior covariance form is $\boldsymbol{\Sigma} \otimes \boldsymbol{V}_0$. However, this does not always make sense in certain applications. For example, imposing zero prior restrictions on some coefficients would imply that the prior variance of these coefficients should be near zero, but this does not need to be true for all coefficients in the model.

To address the first issue, we can think of the VAR(p) specification in a similar way to the seemingly unrelated regression (SUR) model, where we have different regressors in different equations and account for unobserved dependence. This approach allows us to impose zero restrictions on the VAR(p) model, thereby improving its parsimony. Following the setup in Section 7.2, we have

$$\boldsymbol{y}_m = \boldsymbol{Z}_m \boldsymbol{\beta}_m + \boldsymbol{\mu}_m,$$

where $\boldsymbol{y}_m$ is a T-dimensional vector corresponding to the m-th time series variable, $\boldsymbol{Z}_m$ is a $T \times K_m$ matrix of regressors, $\boldsymbol{\beta}_m$ is a K_m-dimensional vector of location parameters, and $\boldsymbol{\mu}_m$ is a T-dimensional vector of stochastic errors, for $m = 1, 2, \ldots, M$.

Stacking the M equations, we can write $\boldsymbol{y} = \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{\mu}$ where $\boldsymbol{y} = \left[\boldsymbol{y}_1^\top \ \boldsymbol{y}_2^\top \ldots \boldsymbol{y}_M^\top\right]^\top$ is a MT-dimensional vector, $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_1^\top \ \boldsymbol{\beta}_2^\top \ldots \boldsymbol{\beta}_M^\top\right]^\top$ is a K dimensional vector, $K = \sum_{m=1}^{M} K_m$, note that having the same number of regressors implies $K = M \cdot k$ coefficients, $\boldsymbol{Z}$ is an $MT \times K$ block diagonal matrix composed of $\boldsymbol{Z}_m$, that is,

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Z}_2 & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{Z}_M \end{bmatrix},$$

and $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^\top \ \boldsymbol{\mu}_2^\top \ldots \ \boldsymbol{\mu}_M^\top\right]^\top$ is a MT-dimensional vector of stochastic errors such that $\boldsymbol{\mu} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_T)$.

We can use independent priors in this model to overcome the limitations of the conjugate prior, that is, $\pi(\boldsymbol{\beta}) \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\pi(\boldsymbol{\Sigma}^{-1}) \sim W(\alpha_0, \boldsymbol{\Psi}_0)$. Thus, we know from Section 7.2 that the posterior distributions are

$$\boldsymbol{\beta} \mid \boldsymbol{\Sigma}, \boldsymbol{y}, \boldsymbol{Z} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

$$\boldsymbol{\Sigma}^{-1} \mid \boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{Z} \sim W(\alpha_n, \boldsymbol{\Psi}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{Z}^\top(\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_T)\boldsymbol{Z} + \boldsymbol{B}_0^{-1})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{Z}^\top(\boldsymbol{\Sigma}^{-1} \otimes \boldsymbol{I}_T)\boldsymbol{y})$, $\alpha_n = \alpha_0 + T$ and $\boldsymbol{\Psi}_n = (\boldsymbol{\Psi}_0^{-1} + \boldsymbol{U}^\top\boldsymbol{U})^{-1}$, where $\boldsymbol{U}$ is an $T \times M$ matrix whose columns are $\boldsymbol{y}_m - \boldsymbol{Z}_m\boldsymbol{\beta}_m$.[5]

[5] We can also use the alternative representation presented in Section 7.2.

Observe that we have standard conditional posteriors, thus, we can employ a Gibbs sampling algorithm to get the posterior draws. We can calculate the prediction $\boldsymbol{y}_{T+1} = [y_{1T+1} \; y_{2T+1} \; \dots \; y_{MT+1}]^\top$ knowing that $\boldsymbol{y}_{T+1} \sim N(\boldsymbol{Z}_T\boldsymbol{\beta}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{Z}_T = \begin{bmatrix} \boldsymbol{z}_{1T}^\top & 0 & \dots & 0 \\ 0 & \boldsymbol{z}_{2T}^\top & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{z}_{MT}^\top \end{bmatrix},$$

and using the posterior draws of $\boldsymbol{\beta}^{(s)}$ and $\boldsymbol{\Sigma}^{(s)}$, $s = 1, 2, \dots, S$. We can also perform inference of functions of the parameters that are of main interest when using VAR models.

Note that independent priors offer more flexibility regarding prior information. For instance, we can set $\boldsymbol{\Psi}_0 = \boldsymbol{S}^{-1}$, $\alpha_0 = T$, $\boldsymbol{\beta}_0 = \boldsymbol{0}$, and $\boldsymbol{B}_0$ as a diagonal matrix, where the variance of the components associated with the coefficients in the m-th equation is such that the prior variance of the coefficients for the own lags is a_1/l^2, the variances for lag l of variable $m \neq j$ are $a_2 s_m^2/(l^2 s_j^2)$, and the variance of the intercepts is set to $a_3 s_m^2$, with $l = 1, 2, \dots, p$, where s_m is the estimated standard error of the residuals from an unrestricted univariate autoregression of variable m against a constant and its p lags [257, 233].

Note that setting $a_1 > a_2$ implies that own lags are more important as predictors than lags of other variables, and dividing by l^2 implies that more recent lags are more relevant than those further in the past. The specific choices of a_1, a_2, and a_3 ($a_k > 0$, $k = 1, 2, 3$) depend on the specific application, but it is generally easier to elicit these parameters rather than the $K(K+1)/2$ different components of $\boldsymbol{B}_0$.[6]

This setting is known as the *Minnesota prior*, as it is based on the seminal proposals for Bayesian VAR models by researchers at the University of Minnesota and the Federal Reserve Bank of Minneapolis [113, 257].[7]

An important non-linear function of parameters when performing VAR analysis is the *impulse response* function, which is, the response of one variable to an impulse in another variable in the model. The *impulse response* function can be deduced using the MA representation of the VAR model. In particular, we can write Equation 8.10 using the lag operator (see Section 8.2),

$$\boldsymbol{y}_t = \boldsymbol{v} + (\boldsymbol{A}_1 L + \boldsymbol{A}_2 L^2 + \dots + \boldsymbol{A}_p L^p)\boldsymbol{y}_t + \boldsymbol{\mu}_t, \tag{8.11}$$

thus $\boldsymbol{A}(L)\boldsymbol{y}_t = \boldsymbol{v} + \boldsymbol{\mu}_t$, where $\boldsymbol{A}(L) = \boldsymbol{I}_M - \boldsymbol{A}_1 L - \boldsymbol{A}_2 L^2 - \dots - \boldsymbol{A}_p L^p$. Let

[6] In our GUI, we use the *bvartools* package, which adopts a slightly different notation such that $a_2 = a_1\kappa_2$ and $a_3 = a_1\kappa_3$. Thus, we need to set a_1, κ_2, and κ_3.

[7] In the case that the variables are not stationary, which is more likely when using variables in levels (e.g., gross domestic product), we set $\boldsymbol{\beta}_0 = \boldsymbol{0}$, except for the elements associated with the first own lags of the dependent variables in each equation, where the prior mean is set to 1. Additionally, the original proposal of the Minnesota prior set $\boldsymbol{\Sigma} = \boldsymbol{S}/T$, meaning it did not account for uncertainty regarding $\boldsymbol{\Sigma}$.

$\boldsymbol{\Phi}(L) := \sum_{s=0}^{\infty} \boldsymbol{\Phi}_s L^s$ an operator such that $\boldsymbol{\Phi}(L)\boldsymbol{A}(L) = \boldsymbol{I}_M$. Thus, we have that $\boldsymbol{\Phi}(L)\boldsymbol{A}(L)\boldsymbol{y}_t = (\sum_{s=0}^{\infty} \boldsymbol{\Phi}_s L^s)\,\boldsymbol{v} + (\sum_{s=0}^{\infty} \boldsymbol{\Phi}_s L^s)\,\boldsymbol{\mu}_t = \boldsymbol{\mu} + \sum_{s=0}^{\infty} \boldsymbol{\Phi}_s \boldsymbol{\mu}_{t-s}$. Note that $L^s\boldsymbol{v} = \boldsymbol{v}$ because $\boldsymbol{v}$ is constant, thus we set $\sum_{s=0}^{\infty} \boldsymbol{\Phi}_s L^s \boldsymbol{v} = \sum_{s=0}^{\infty} \boldsymbol{\Phi}_s \boldsymbol{v} = \boldsymbol{\Phi}(1)\boldsymbol{v} = (\boldsymbol{I}_M - \boldsymbol{A}_1 - \boldsymbol{A}_2 - \cdots - \boldsymbol{A}_p)^{-1}\boldsymbol{v} := \boldsymbol{\mu}$, which is the mean of the process [189, Chap. 2]. Therefore, the MA representation of the VAR is

$$\boldsymbol{y}_t = \boldsymbol{\mu} + \sum_{s=0}^{\infty} \boldsymbol{\Phi}_s \boldsymbol{\mu}_{t-s}, \tag{8.12}$$

where $\boldsymbol{\Phi}_0 = \boldsymbol{I}_M$, and we can get the coefficients in $\boldsymbol{\Phi}_s$ by the recursion $\boldsymbol{\Phi}_s = \sum_{l=1}^{s} \boldsymbol{\Phi}_{s-l}\boldsymbol{A}_l$, $\boldsymbol{A}_l = \boldsymbol{0}$, $l > p$ and $s = 1, 2, ..$ [189, Chap. 2]. This impulse response function is called *forecast error impulse response function.*

The MA coefficients contain the impulse responses of the system. In particular, $\phi_{mj,s}$, which is the mj-th element of the matrix $\boldsymbol{\Phi}_s$, represents the response of the m-th variable to a unit shock of the variable j in the system, s periods ago, provided that the effect is not contaminated by other shocks in the system. The long-term effects (total multipliers) are given by $\boldsymbol{\Psi}_\infty := \sum_{s=1}^{\infty} \boldsymbol{\Phi}_s = (\boldsymbol{I}_M - \boldsymbol{A}_1 - \boldsymbol{A}_2 - \cdots - \boldsymbol{A}_p)^{-1}$.

An assumption in these *impulse response* functions is that a shock occurs in only one variable at a time. This can be questionable as different shocks may be correlated, consequently, occurring simultaneously. Thus, the *impulse response* analysis can be performed based on the alternative MA representation, $\boldsymbol{y}_t = \boldsymbol{\mu} + \sum_{s=0}^{\infty} \boldsymbol{\Phi}_s \boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{\mu}_{t-s} = \boldsymbol{\mu} + \sum_{s=0}^{\infty} \boldsymbol{\Theta}_s \boldsymbol{w}_{t-s}$, where $\boldsymbol{\Theta}_s = \boldsymbol{\Phi}_s\boldsymbol{P}$ and $\boldsymbol{w}_t = \boldsymbol{P}^{-1}\boldsymbol{\mu}_t$, $\boldsymbol{P}$ is a lower triangular matrix such that $\boldsymbol{\Sigma} = \boldsymbol{P}\boldsymbol{P}^\top$ (Cholesky factorization/decomposition). Note that the covariance matrix of $\boldsymbol{w}_t$ is $\boldsymbol{I}_M$ due to $\mathbb{E}[\boldsymbol{w}_t\boldsymbol{w}_t^\top] = \mathbb{E}[\boldsymbol{P}^{-1}\boldsymbol{\mu}_t\boldsymbol{\mu}_t^\top(\boldsymbol{P}^{-1})^\top] = \boldsymbol{P}^{-1}\boldsymbol{\Sigma}(\boldsymbol{P}^{-1})^\top = \boldsymbol{P}^{-1}\boldsymbol{P}\boldsymbol{P}^\top(\boldsymbol{P}^{-1})^\top = \boldsymbol{I}_M$.

In this representation is sensible to assume that each shock occurs independently due to the covariance matrix of $\boldsymbol{w}_t$ being an identity. In addition, a unit shock is a shock of size one standard deviation due the result of the covariance matrix. This is named the *ortogonalized impulse response*, where $\theta_{mj,s}$, which is the mj-th element of the matrix $\boldsymbol{\Theta}_s$, represents the response of the m-th variable to a standard deviation shock of the variable j in the system, s periods ago. The critical point with the ortogonalized impulse responses is that the order of the variables in the VAR is really important because implicitly establishes a recursive model, that is, the m-th equation in the system may contain $y_{1t}, y_{2t}, \ldots, y_{m-1t}$, but not $y_{mt}, y_{m+1t}, \ldots, y_{Mt}$ on the hand-right side of its equation. Thus, y_{mt} cannot have an instantaneous impact on y_{jt} for $j < m$ [189, Chap. 2].

Beyond the fascinating macroeconomic implications embedded in the specification of VAR models, the key point for this section is that we can infer impulse response functions using the posterior draws.

Example: US fiscal system

Let's use the dataset provided by [424] of the US fiscal system, where

ttr is the quarterly total tax revenue, *gs* is the quarterly total government spending, and *gdp* is the quarterly gross domestic product, all expressed in log, real, per person terms, and the period is 1948q1 to 2024q2. This dataset is the *18USAfiscal.csv* file. [283] analyze the US fiscal policy shocks using these variables.

Let's estimate a VAR model where $\boldsymbol{y}_t = [\Delta(ttr_t)\ \Delta(gs_t)\ \Delta(gdp_t)]^\top$, that is, we work with the log differences (variation rates), and we set $p = 1$. We use the package *bvartools* to estimate the *forecast error* and *ortogonalized* impulse response functions. We use vague independent priors setting $\boldsymbol{\beta}_0 = \mathbf{0}$, $\boldsymbol{B}_0 = 100\boldsymbol{I}$, $\boldsymbol{V}_0 = 5^{-1}\boldsymbol{I}$ and $\alpha_0 = 3$, and the Minnesota prior setting $a_1 = 2$, $\kappa_2 = 0.5$ and $\kappa_3 = 5$ (default values).[8]

The following code shows how to do this, take into account that we use the first 301 observations to estimate the model, and keep the last 4 observations to check the forecasting performance. Figures 8.6 and 8.7 show the impulse response functions of *gs* with respect to *gs*, the *forecast error impulse response* using vague independent priors, and the *orthogonalized impulse response* using the Minnesota prior, respectively. We see that the effect of the Minnesota prior is to decrease uncertainty. In addition, the forecasting exercise results indicate that these assumptions have same effects in this example. In particular, Figure 8.8 shows that the mean forecasts using the vague prior (green line) and the Minnesota prior (red line) are indistinguishable from the true observations (black line). However, the Minnesota prior enhances forecast precision, as its 95% predictive interval (blue shaded area) is narrower and fully contained within the 95% predictive interval obtained using vague priors (light blue shaded area). This improvement is attributable to the shrinkage properties of the Minnesota prior.

[8]The *bvartools* package uses the inverse Wishart distribution as prior for $\boldsymbol{\Sigma}$, where the hyperparameters are the degrees of freedom of the error term, and the prior error variance of endogenous variables.

R code. VAR model: US fiscal shocks

```
rm(list = ls()); set.seed(010101)
DataUSfilcal <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/18USAfiscal.csv", sep = ",", header = TRUE, quote = "")
attach(DataUSfilcal) # upload data
Y <- cbind(diff(as.matrix(DataUSfilcal[,-c(1:2)])))
T <- dim(Y)[1]-1; K <- dim(Y)[2]
Ynew <- Y[-c((T-2):(T+1)), ] # Use 4 last observations to check forecast
y1 <- Ynew[-1, 1]; y2 <- Ynew[-1, 2]; y3 <- Ynew[-1, 3]
X1 <- cbind(1, lag(Ynew)); X1 <- X1[-1,]
X2 <- cbind(1, lag(Ynew)); X2 <- X2[-1,]
X3 <- cbind(1, lag(Ynew)); X3 <- X3[-1,]
M <- dim(Y)[2]; K1 <- dim(X1)[2]; K2 <- dim(X2)[2]; K3 <- dim(X3)[2]
K <- K1 + K2 + K3
# Hyperparameters
b0 <- 0; c0 <- 100; V0 <- 5^(-1); a0 <- M
#Posterior draws
MCMC <- 10000; burnin <- 1000; H <- 10; YnewPack <- ts(Ynew)
model <- bvartools::gen_var(YnewPack, p = 1, deterministic = "const", iterations = MCMC, burnin = burnin) # Create model
model <- bvartools::add_priors(model, coef = list(v_i = c0^-1, v_i_det = c0^-1, const = b0), sigma = list(df = a0, scale = V0/a0), coint_var = FALSE) # Add priors
object <- bvartools::draw_posterior(model) # Posterior draws
ir <- bvartools::irf.bvar(object, impulse = "gs", response = "gs", n.ahead = H, type = "feir", cumulative = FALSE) # Calculate IR
# Plot IR
plot_IR <- function(df) {
  p <- ggplot(data = df, aes(x = t)) + geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 1, fill = "lightblue") + geom_line(aes(y = mean), colour = "blue", linewidth = 0.5) + ylab("Impulse response") + xlab("Time") + xlim(0, H)
  print(p)
}
dfNew <- tibble(t = 0:H, mean = as.numeric(ir[,2]), lower = as.numeric(ir[,1]), upper = as.numeric(ir[,3]))
FigNew <- plot_IR(dfNew)
# Using Minnesota prior
library(tibble)
modelMin <- bvartools::gen_var(YnewPack, p = 1, deterministic = "const", iterations = MCMC, burnin = burnin)
modelMin <- bvartools::add_priors(modelMin, minnesota = list(kappa0 = 2, kappa1 = 0.5, kappa3 = 5), coint_var = FALSE) # Minnesota prior
objectMin <- bvartools::draw_posterior(modelMin) # Posterior draws
irMin <- bvartools::irf.bvar(objectMin, impulse = "gs", response = "gs", n.ahead = H, type = "feir", cumulative = FALSE) # Calculate IR
dfNewMin <- tibble(t = 0:H, mean = as.numeric(irMin[,2]), lower = as.numeric(irMin[,1]), upper = as.numeric(irMin[,3]))
FigNewMin <- plot_IR(dfNewMin)
```

R code. VAR model: US fiscal shocks

```
### Forecasting
bvar_pred <- predict(object, n.ahead = 4, new_d = rep(1, 4))
bvar_predOR <- predict(objectMin, n.ahead = 4, new_d = rep
    (1, 4))
dfFore <- tibble(t = c((T-2):(T+1)), mean = as.numeric(bvar_
    pred[["fcst"]][["gs"]][,2]), lower = as.numeric(bvar_
    pred[["fcst"]][["gs"]][,1]), upper = as.numeric(bvar_
    pred[["fcst"]][["gs"]][,3]), mean1 = as.numeric(bvar_
    predOR[["fcst"]][["gs"]][,2]), lower1 = as.numeric(bvar_
    predOR[["fcst"]][["gs"]][,1]), upper1 = as.numeric(bvar_
    predOR[["fcst"]][["gs"]][,3]), true = as.numeric(Y[c((T
    -2):(T+1)),2]))
plot_FORE <- function(df) {
  p <- ggplot(data = dfFore, aes(x = t)) + geom_ribbon(aes(
    ymin = lower1, ymax = upper1), alpha = 1, fill = "blue")
     + geom_ribbon(aes(ymin = lower, ymax = upper), alpha =
    1, fill = "lightblue") + geom_line(aes(y = mean), colour
     = "green", linewidth = 0.5) + geom_line(aes(y = mean1),
     colour = "red", linewidth = 0.5) + geom_line(aes(y =
    true), colour = "black", linewidth = 0.5) + ylab("
    Forecast") + xlab("Time") + xlim(c((T-2),(T+1)))
  print(p)
}
FigFore <- plot_FORE(dfFore)
```

Algorithm A24 shows how to do perform inference in VAR models using our GUI. See also Chapter 5 for details regarding the dataset structure.

There are other good packages in **R** to perform Bayesian inference in VAR models. For instance, *bayesianVARs* package implements inference of reduced-form VARs with stochastic volatility [260], *BVAR* package performs inference using hierarchical priors [297], *bvarsv* implements time-varying parameters models [237], *bsvars* performs estimation of structural VAR models [424], and *bsvarSIGNs* to estimating structural VAR models with sign restrictions [426].

8.5 Summary

We present a brief review of Bayesian inference in time series models. In particular, we introduce the *state-space* representation and demonstrate how to perform inferential analysis for these models, focusing on the dynamic linear model and the stochastic volatility model. Additionally, we show how

FIGURE 8.6
Forecasting error impulse response: the government spending (gs) with respect to the government spending (gs).

FIGURE 8.7
Ortogonalized impulse response: the government spending (gs) with respect to the government spending (gs).

FIGURE 8.8
Forecast performance: the government spending (gs).

Algorithm A24 Vector Autoregressive models

1: Select *Time series Model* on the top panel
2: Select *VAR models* using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Set the number of lags (p)
6: Set the hyperparameters for the Minnesota prior: a_1, κ_2 and κ_3. This step is not necessary as by default our GUI uses default values in *bvartools* package
7: Select the type of *impulse response functions*: forecast error or orthogonalized, and ordinary or cumulative.
8: Set the time horizon for the impulse response functions and the forecasts
9: Click the *Go!* button
10: Analyze results
11: Download impulse responses and forecasts using the *Download Results* button

$ARMA(p, q)$ processes can be expressed in *state-space* form and provide methods for estimating such models.

We also include code for implementing computational inference algorithms,

such as sequential Monte Carlo (SMC), Hamiltonian Monte Carlo (HMC), and various Markov chain Monte Carlo (MCMC) methods. Finally, we introduce VAR(p) models, detailing how to perform impulse-response analysis and forecasting within this framework.

Time series analysis is a highly active research area with remarkable methodological developments and applications. Interested readers can refer to excellent materials in chapters 7 and 9 of [164], and chapters 17 to 20 of [71], along with the references therein.

8.6 Exercises

1. Simulate the *dynamic linear model* assuming $X_t \sim N(1, 0.1\sigma^2)$, $w_t \sim N(0, 0.5\sigma^2)$, $\mu_t \sim N(0, \sigma^2)$, $\beta_0 = 1$, $B_0 = 0.5\sigma^2$, $\sigma^2 = 0.25$, and $G_t = 1$, $t = 1, \ldots, 100$. Then, perform the filtering recursion fixing $\Sigma = 25 \times 0.25$, $\Omega_1 = 0.5\Sigma$ (high signal-to-noise ratio) and $\Omega_2 = 0.1\Sigma$ (low signal-to-noise ratio). Plot and compare the results.
2. Simulate the *dynamic linear model* $y_t = \beta_t x_t + \mu_t$, $\beta_t = \beta_{t-1} + w_t$, where $x_t \sim N(1, 0.1\sigma^2)$, $w_t \sim N(0, 0.5\sigma^2)$, $\mu_t \sim N(0, \sigma^2)$, $\beta_0 = 0$, $B_0 = 0.5\sigma^2$, and $\sigma^2 = 1$, $t = 1, \ldots, 100$. Perform the filtering and smoothing recursions from scratch.
3. Simulate the process $y_t = \alpha z_t + \beta_t x_t + \boldsymbol{h}^\top \boldsymbol{\epsilon}_t$, $\beta_t = \beta_{t-1} + \boldsymbol{H}^\top \boldsymbol{\epsilon}_t$, where $\boldsymbol{h}^\top = [1\ 0]$, $\boldsymbol{H}^\top = [0\ 1/\tau]$, $\boldsymbol{v}_t \sim N(\boldsymbol{0}_2, \sigma^2 \boldsymbol{I}_2)$, $x_t \sim N(1, 2\sigma^2)$, $z_t \sim N(0, 2\sigma^2)$, $\alpha = 2$, $\tau^2 = 5$ and $\sigma^2 = 0.1$, $t = 1, \ldots, 200$. Assume $\pi(\beta_0, \alpha, \sigma^2, \tau) = \pi(\beta_0)\pi(\alpha)\pi(\sigma^2)\pi(\tau^2)$ where $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$, $\tau^2 \sim G(v_0/2, v_0/2)$, $\alpha \sim N(a_0, A_0)$ and $\beta_0 \sim N(b_0, B_0)$ such that $\alpha_0 = \delta_0 = 1$, $v_0 = 5$, $a_0 = 0$, $A_0 = 1$, $\beta_0 = 0$, $B_0 = \sigma^2/\tau^2$. Program the MCMC algorithm including the *simulation smoother*.
4. Show that the posterior distribution of $\boldsymbol{\phi} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X}$ in the model $y_t = \boldsymbol{x}_t^\top \boldsymbol{\beta} + \mu_t$ where $\phi(L)\mu_t = \epsilon_t$ and $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$ is $N(\boldsymbol{\phi}_n, \boldsymbol{\Phi}_n)\mathbb{1}(\boldsymbol{\phi} \in S_{\boldsymbol{\phi}})$, where $\boldsymbol{\Phi}_n = (\boldsymbol{\Phi}_0^{-1} + \sigma^{-2}\boldsymbol{U}^\top \boldsymbol{U})$, $\boldsymbol{\phi}_n = \boldsymbol{\Phi}_n(\boldsymbol{\Phi}_0^{-1}\boldsymbol{\phi}_0 + \sigma^{-2}\boldsymbol{U}^\top \boldsymbol{\mu})$, and S_ϕ is the stationary region of $\boldsymbol{\phi}$.
5. Show that in the $AR(2)$ stationary process, $y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t$, where $\epsilon_t \sim N(0, \sigma^2)$, $\mathbb{E}[y_t] = \frac{\mu}{1-\phi_1-\phi_2}$, and $Var[y_t] = \frac{\sigma^2(1-\phi_2)}{1-\phi_2-\phi_1^2-\phi_1^2\phi_2-\phi_2^2+\phi_2^3}$.
6. Program a Hamiltonian Monte Carlo taking into account the stationary restrictions on ϕ_1 and ϕ_2, and ϵ_0 such that the acceptance rate is near 65%.
7. **Stochastic volatility model**

   •Program a sequential importance sampling (SIS) from scratch

in the vanilla stochastic volatility model setting $\mu = -10$, $\phi = 0.95$, $\sigma = 0.3$ and $T = 250$. Check what happen with its performance.

- Modify the sequential Monte Carlo (SMC) to perform multinomial resampling when the effective sample size is lower than 50% the initial number of particles.

8. Estimate the vanilla stochastic volatility model using the dataset *17ExcRate.csv*, provided by [331], which contains the exchange rate log daily returns for USD/EUR, USD/GBP, and GBP/EUR from one year before and after the WHO declared the COVID-19 pandemic on 11 March 2020.

9. Simulate the VAR(1) process

$$\begin{bmatrix} y_{1t} \\ y_{2t} \\ y_{3t} \end{bmatrix} = \begin{bmatrix} 2.8 \\ 2.2 \\ 1.3 \end{bmatrix} + \begin{bmatrix} 0.5 & 0 & 0 \\ 0.1 & 0.1 & 0.3 \\ 0 & 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} y_{1t-1} \\ y_{2t-1} \\ y_{3t-1} \end{bmatrix} + \begin{bmatrix} \mu_{1t} \\ \mu_{2t} \\ \mu_{3t} \end{bmatrix},$$

where $\boldsymbol{\Sigma} = \begin{bmatrix} 2.25 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 0.74 \end{bmatrix}$.

- Use vague independent priors setting $\boldsymbol{\beta}_0 = \mathbf{0}$, $\boldsymbol{B}_0 = 100\boldsymbol{I}$, $\boldsymbol{V}_0 = 5\boldsymbol{I}$ and $\alpha_0 = 3$, and estimate a VAR(1) model using the *rsurGibbs* function from the package *bayesm*. Then, program from scratch algorithms to perform inference of the *forecast error* and *ortogonalized* impulse response functions, and compare with the population impulse response functions, that is, using the population parameters.

- Using the previous setting perform inference of the *forecast error* and the *ortogonalized* impulse responses using the package *bvartools*.

9

Longitudinal/Panel data models

We describe how to perform inference in longitudinal/panel models using a Bayesian framework. In this context, multiple cross-sectional units are observed repeatedly over time, a structure referred to as panel data by econometricians and longitudinal data by statisticians. Specifically, we present models for continuous (normal), binary (logit), and count (Poisson) responses. Applications and exercises illustrate the potential of these models.

In longitudinal/panel data sets, we have y_{it} where $i = 1, 2, \ldots, N$ and $t = 1, 2, \ldots, T_i$. If $T_i = T$ for all i, the dataset is *balanced*; otherwise, it is *unbalanced*. Longitudinal data typically involves by far more cross-sectional units than time periods, this is called typically a *short panel*. It assumes that cross-sectional units are independent, though serial correlation exists within each unit over time, and unobserved heterogeneity for each unit must be accounted for. We can treat this unobserved heterogeneity as random variables, assuming it is either independent or dependent on control variables. Econometricians refer to these cases as *random effects* and *fixed effects*, respectively. The Bayesian literature takes a different approach, modeling the panel structure hierarchically, where the unobserved heterogeneity may or may not depend on other controls.[1]

Remember that the easiest way to run our GUI is typing

R code. How to display our graphical user interface

```
shiny::runGitHub("besmarter/BSTApp", launch.browser = T)
```

in the **R** package console or any **R** code editor, and once our GUI is deployed, select *Hierarchical Longitudinal Models*. However, users should see Chapter 5 for details.

[1] See [339] for a nice comparison of Frequentist and Bayesian treatments of panel data models.

9.1 Normal model

The longitudinal/panel normal model establishes $\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta}+\boldsymbol{W}_i\boldsymbol{b}_i+\boldsymbol{\mu}_i$ where $\boldsymbol{y}_i$ are T_i-dimensional vectors corresponding to units $i = 1, 2, \ldots, N$, $\boldsymbol{X}_i$ and $\boldsymbol{W}_i$ are $T_i \times K_1$ and $T_i \times K_2$ matrices, respectively. In the statistical literature, $\boldsymbol{\beta}$ is a K_1-dimensional vector of *fixed effects*, and $\boldsymbol{b}_i$ is a K_2-dimensional vector of unit-specific *random effects* that allow unit-specific means, and enable to capture marginal dependence among the observations on the cross-sectional units. We assume normal stochastic errors, $\boldsymbol{\mu}_i \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I}_{T_i})$, which means that the likelihood function is

$$
\begin{aligned}
p(\boldsymbol{\beta}, \boldsymbol{b}, \sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) &\propto \prod_{i=1}^{N} |\sigma^2\boldsymbol{I}_{T_i}|^{-1/2} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)\right\} \\
&= (\sigma^2)^{-\frac{\sum_{i=1}^{N} T_i}{2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)\right\},
\end{aligned}
$$

where $\boldsymbol{b} = [\boldsymbol{b}_1^\top, \boldsymbol{b}_2^\top, \ldots, \boldsymbol{b}_N^\top]^\top$.

Panel data modeling in the Bayesian approach assumes a hierarchical structure in the *random effects*. Following [82], there is a first stage where $\boldsymbol{b}_i \sim N(\boldsymbol{0}, \boldsymbol{D})$, $\boldsymbol{D}$ allows serial correlation within each cross-sectional unit i, and then, there is a second stage where $\boldsymbol{D} \sim IW(d_0, d_0\boldsymbol{D}_0)$. Thus, we can see that there is an additional layer of priors as there is a prior on the hyperparameter $\boldsymbol{D}$.

In addition, we have standard conjugate prior distributions for $\boldsymbol{\beta}$ and σ^2, $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\sigma^2 \sim IG(\alpha_0, \delta_0)$.

[82] propose a blocking algorithm to perform inference in longitudinal hierarchical models by considering the distribution of $\boldsymbol{y}_i$ marginalized over the random effects. Given that $\boldsymbol{y}_i \mid \boldsymbol{\beta}, \boldsymbol{b}_i, \sigma^2, \boldsymbol{X}_i, \boldsymbol{W}_i \sim N(\boldsymbol{X}_i\boldsymbol{\beta}+\boldsymbol{W}_i\boldsymbol{b}_i, \sigma^2\boldsymbol{I}_{T_i})$, we can see that $\boldsymbol{y}_i \mid \boldsymbol{\beta}, \boldsymbol{D}, \sigma^2, \boldsymbol{X}_i, \boldsymbol{W}_i \sim N(\boldsymbol{X}_i\boldsymbol{\beta}, \boldsymbol{V}_i)$, where $\boldsymbol{V}_i = \sigma^2\boldsymbol{I}_{T_i} + \boldsymbol{W}_i\boldsymbol{D}\boldsymbol{W}_i^\top$ given that $\mathbb{E}[\boldsymbol{b}_i] = \boldsymbol{0}$ and $Var[\boldsymbol{b}_i] = \boldsymbol{D}$. If we have just random intercepts, then $\boldsymbol{W}_i = \boldsymbol{i}_{T_i}$, where $\boldsymbol{i}_{T_i}$ is a T_i-dimensional vector of ones. Thus, $\boldsymbol{V}_i = \sigma^2\boldsymbol{I}_{T_i} + \sigma_b^2\boldsymbol{i}_{T_i}\boldsymbol{i}_{T_i}^\top$, the variance is $\sigma^2 + \sigma_b^2$ and the covariance is σ_b^2 within each cross-sectional unit through time.

We can deduce the posterior distribution of $\boldsymbol{\beta}$ given σ^2 and $\boldsymbol{D}$,

$$
\begin{aligned}
\pi(\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) \propto \exp&\left\{-\frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta})^\top\boldsymbol{V}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta})\right\} \\
&\times \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top\boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)\right\}.
\end{aligned}
$$

This implies that (see Exercise 1)

$$
\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),
$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^{N}\boldsymbol{X}_i^\top\boldsymbol{V}_i^{-1}\boldsymbol{X}_i)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^{N}\boldsymbol{X}_i^\top\boldsymbol{V}_i^{-1}\boldsymbol{y}_i)$.

We can use the likelihood $p(\boldsymbol{\beta}, \boldsymbol{b}_i, \sigma^2 \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W})$ to get the posterior distributions of $\boldsymbol{b}_i$, σ^2 and $\boldsymbol{D}$. In particular,

$$\begin{aligned}
\pi(\boldsymbol{b}_i \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) &\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)\right\} \\
&\quad \times \exp\left\{-\frac{1}{2}\sum_{i=1}^{N}\boldsymbol{b}_i^\top \boldsymbol{D}^{-1}\boldsymbol{b}_i\right\} \\
&\propto \exp\left\{-\frac{1}{2}\sum_{i=1}^{N}(-2\boldsymbol{b}_i^\top(\sigma^{-2}\boldsymbol{W}_i^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta})) + \boldsymbol{b}_i^\top(\sigma^{-2}\boldsymbol{W}_i^\top\boldsymbol{W}_i + \boldsymbol{D}^{-1})\boldsymbol{b}_i)\right\} \\
&\propto \exp\left\{-\frac{1}{2}(-2\boldsymbol{b}_i^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{B}_{ni}(\sigma^{-2}\boldsymbol{W}_i^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta})) + \boldsymbol{b}_i^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_i)\right\} \\
&= \exp\left\{-\frac{1}{2}(-2\boldsymbol{b}_i^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_{ni} + \boldsymbol{b}_i^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_i)\right\},
\end{aligned}$$

where $\boldsymbol{B}_{ni} = (\sigma^{-2}\boldsymbol{W}_i^\top\boldsymbol{W}_i + \boldsymbol{D}^{-1})^{-1}$ and $\boldsymbol{b}_{ni} = \boldsymbol{B}_{ni}(\sigma^{-2}\boldsymbol{W}_i^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta}))$.

We can complete the square in this expression by adding and subtracting $\boldsymbol{b}_{ni}^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_{ni}$. Thus,

$$\begin{aligned}
\pi(\boldsymbol{b}_i \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) &\propto \exp\left\{-\frac{1}{2}(-2\boldsymbol{b}_i^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_{ni} + \boldsymbol{b}_i^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_i + \boldsymbol{b}_{ni}^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_{ni} - \boldsymbol{b}_{ni}^\top\boldsymbol{B}_{ni}^{-1}\boldsymbol{b}_{ni})\right\} \\
&\propto \exp\left\{(\boldsymbol{b}_i - \boldsymbol{b}_{ni})^\top\boldsymbol{B}_{ni}^{-1}(\boldsymbol{b}_i - \boldsymbol{b}_{ni})\right\}.
\end{aligned}$$

This is the kernel of a multivariate normal distribution with mean $\boldsymbol{b}_{ni}$ and variance $\boldsymbol{B}_{ni}$. Thus,

$$\boldsymbol{b}_i \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim N(\boldsymbol{b}_{ni}, \boldsymbol{B}_{ni}),$$

Let's see the posterior distribution of σ^2,

$$\begin{aligned}
\pi(\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) &\propto (\sigma^2)^{-\frac{\sum_{i=1}^{N} T_i}{2}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)\right\} \\
&\quad \times (\sigma^2)^{-\alpha_0 - 1}\exp\left\{-\frac{\delta_0}{\sigma^2}\right\} \\
&= (\sigma^2)^{-\frac{\sum_{i=1}^{N} T_i}{2} - \alpha_0 - 1} \\
&\quad \times \exp\left\{-\frac{1}{\sigma^2}\left(\delta_0 + \sum_{i=1}^{N}\frac{(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)}{2}\right)\right\}.
\end{aligned}$$

Thus,

$$\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim IG(\alpha_n, \delta_n),$$

where $\alpha_n = \alpha_0 + \frac{1}{2}\sum_{i=1}^{N} T_i$ and $\delta_n = \delta_0 + \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)$.

The posterior distribution of $\boldsymbol{D}$ is the following,

$$\begin{aligned}\pi(\boldsymbol{D} \mid \boldsymbol{b}) &\propto |\boldsymbol{D}|^{-N/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^{N} \boldsymbol{b}_i^\top \boldsymbol{D}^{-1}\boldsymbol{b}_i\right\} \\ &\quad \times |\boldsymbol{D}|^{-(d_0+K_2+1)/2} \exp\left\{-\frac{1}{2} tr(d_0\boldsymbol{D}_0\boldsymbol{D}^{-1})\right\} \\ &= |\boldsymbol{D}|^{-(d_0+N+K_2+1)/2} \exp\left\{-\frac{1}{2} tr\left(\left(d_0\boldsymbol{D}_0 + \sum_{i=1}^{N} \boldsymbol{b}_i\boldsymbol{b}_i^\top\right)\boldsymbol{D}^{-1}\right)\right\}.\end{aligned}$$

This is the kernel of an inverse Wishart distribution with degrees of freedom $d_n = d_0 + N$ and scale matrix $\boldsymbol{D}_n = d_0\boldsymbol{D}_0 + \sum_{i=1}^{N} \boldsymbol{b}_i\boldsymbol{b}_i^\top$. Thus,

$$\boldsymbol{D} \mid \boldsymbol{b} \sim IW(d_n, \boldsymbol{D}_n).$$

Observe that the posterior distribution of $\boldsymbol{D}$ dependents just on $\boldsymbol{b}$.

All the posterior conditional distributions belong to standard families, this implies that we can use a Gibbs sampling algorithm to perform inference in these hierarchical normal models.

Example: The relation between productivity and public investment

We used the dataset named *8PublicCap.csv* used by [326] to analyze the relation between public investment and gross state product in the setting of a spatial panel dataset consisting of 48 US states from 1970 to 1986. In particular, we perform inference based on the following equation

$$\log(\text{gsp}_{it}) = b_i + \beta_1 + \beta_2 \log(\text{pcap}_{it}) + \beta_3 \log(\text{pc}_{it}) + \beta_4 \log(\text{emp}_{it}) + \beta_5 \text{unemp}_{it} + \mu_{it},$$

where gsp in the gross state product, pcap is public capital, and pc is private capital all in USD, emp is employment (people), and unemp is the unemployment rate in percentage.

Algorithm A25 shows how to perform inference in hierarchical longitudinal normal models in our GUI. See also Chapter 5 for details regarding the dataset structure.

We ask in Exercise 2 to run this application in our GUI using 10000 MCMC iterations plus a burn-in equal to 5000 iterations, and a thinning parameter equal to 1. We also used the default values for the hyperparameters of the prior distributions, that is, $\boldsymbol{\beta}_0 = \boldsymbol{0}_5$, $\boldsymbol{B}_0 = \boldsymbol{I}_5$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 5$ and $\boldsymbol{D}_0 = \boldsymbol{I}_1$. It seems that all posterior draws come from stationary distributions, as suggested by the diagnostics and posterior plots (see Exercise 2).

The following code uses the command *MCMChregress* from the package *MCMCpack* to run this application. This command is also used by our GUI to perform inference in hierarchical longitudinal normal models.

We can see that the 95% symmetric credible intervals for public capital, private capital, employment, and unemployment are (-2.54e-02, -2.06e-02),

Algorithm A25 Hierarchical longitudinal normal models

1: Select *Hierarchical Longitudinal Model* on the top panel
2: Select *Normal* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Write down the formula of the *fixed effects* equation in the **Main Equation: Fixed Effects** box. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation
6: Write down the formula of the *random effects* equation in the **Main Equation: Random Effects** box without writing the dependent variable, that is, starting the equation with the *tilde* ("∼") symbol. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation. If there are just random intercepts do not write anything in this box
7: Write down the name of the grouping variable, that is, the variable that indicates the cross-sectional units
8: Set the hyperparameters of the *fixed effects*: mean vector, covariance matrix, shape and scale parameters. This step is not necessary as by default our GUI uses non-informative priors
9: Set the hyperparameters of the *random effects*: degrees of freedom and scale matrix of the inverse Wishart distribution. This step is not necessary as by default our GUI uses non-informative priors
10: Click the *Go!* button
11: Analyze results
12: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

(2.92e-01, 2.96e-01), (7.62e-01, 7.67e-01) and (-5.47e-03, -5.31e-03), respectively. The posterior mean elasticity estimate of public capital to gsp is -0.023, that is, an increase by 1% in public capital means a 0.023% decrease in gross state product. The posterior mean estimates of private capital and employment elasticities are 0.294 and 0.765, respectively. In addition, 1 percentage point increase in the unemployment rate means a decrease of 0.54% in gsp. It seems that all these variables are statistically relevant. In addition, the posterior mean estimates of the variance associated with the unobserved heterogeneity and stochastic errors are 1.06e-01 and 1.45e-03. We obtained the posterior chain of the proportion of the variance associated with the unobserved heterogeneity. The 95% symmetric credible interval is (0.98, 0.99) for

this proportion, that is, unobserved heterogeneity is very important to explain the total variability.

R code. The relationship between productivity and public investment, MCMChregress command

```
rm(list = ls())
set.seed(12345)
DataGSP <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/8PublicCap.
    csv", sep = ",", header = TRUE, quote = "")
attach(DataGSP)
K1 <- 5; K2 <- 1
b0 <- rep(0, K1); B0 <- diag(K1)
r0 <- 5; R0 <- diag(K2)
a0 <- 0.001; d0 <- 0.001
Resultshreg <- MCMCpack::MCMChregress(fixed = log(gsp)~log(
    pcap)+log(pc)+log(emp)+unemp, random = ~1, group = "id",
     data = DataGSP, burnin = 5000, mcmc = 10000, thin = 1,
    r = r0, R = R0, nu = a0, delta = d0)
Betas <- Resultshreg[["mcmc"]][,1:K1]
Sigma2RanEff <- Resultshreg[["mcmc"]][,54]
Sigma2 <- Resultshreg[["mcmc"]][,55]
summary(Betas)
Quantiles for each variable:
            2.5%       25%       50%      75%     97.5%
beta.(Intercept)  2.3145  2.3246  2.3301  2.335  2.3455
beta.log(pcap)   -0.0254 -0.0239 -0.0231 -0.022 -0.0206
beta.log(pc)      0.2917  0.2930  0.2937  0.294  0.2957
beta.log(emp)     0.7619  0.7637  0.7646  0.765  0.7672
beta.unemp       -0.0054 -0.0054 -0.0053 -0.005 -0.0053
summary(Sigma2RanEff)
Quantiles for each variable:
2.5%     25%     50%     75%   97.5%
0.07208 0.09086 0.10331 0.11751 0.15600
summary(Sigma2)
Quantiles for each variable:
2.5%      25%      50%      75%    97.5%
0.001316 0.001403 0.001451 0.001501 0.001606
summary(Sigma2RanEff/(Sigma2RanEff+Sigma2))
Quantiles for each variable:
2.5%    25%    50%    75%  97.5%
0.9799 0.9842 0.9861 0.9879 0.9909
```

There are many extensions of this model, for instance, [82] propose to introduce heteroskedasticity in this model by assuming $\mu_{it} \mid \tau_{it} \sim N(0, \sigma^2/\tau_{it})$, $\tau_{it} \sim G(v/2, v/2)$. We ask in Exercise 2 to perform inference in the relation between productivity and public investment example using this setting. Another potential extension is to allow dependence between $\boldsymbol{b}_i$ and some con-

trols, let's say $\boldsymbol{z}_i$, a K_3-dimensional vector, and assume $\boldsymbol{b}_i \sim N(\boldsymbol{Z}_i\boldsymbol{\gamma}, \boldsymbol{D})$ where $\boldsymbol{Z}_i = \boldsymbol{I}_{K_2} \otimes \boldsymbol{z}_i^\top$, and complete the model using a prior for $\boldsymbol{\gamma}$, $\boldsymbol{\gamma} \sim N(\boldsymbol{\gamma}_0, \boldsymbol{\Gamma}_0)$. We ask to perform a simulation using this setting in Exercise 3.

Example: Simulation exercise of the longitudinal normal model with heteroskedasticity

Let's perform a simulation exercise to assess some potential extensions of the longitudinal hierarchical normal model. The point of departure is to assume that

$$y_{it} = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \beta_3 x_{it3} + b_i + w_{it1} b_{i1} + \mu_{it},$$

where $x_{itk} \sim N(0,1)$, $k = 1,2,3$, $w_{it1} \sim N(0,1)$, $b_i \sim N(0, 0.7^{1/2})$, $b_{i1} \sim N(0, 0.6^{1/2})$, $\mu_{it} \sim N(0, (0.1/\tau)^{1/2})$, $\tau_{it} \sim G(v/2, v/2)$ and $\boldsymbol{\beta} = [0.5\ 0.4\ 0.6\ -0.6]^\top$, $i = 1, 2, \ldots, 50$. The sample size is 2000 in an *unbalanced panel structure*.

Following same stages as in this section and Exercise 1, the posterior conditional distributions assuming that $\mu_{it} \mid \tau_{it} \sim N(0, \sigma^2/\tau_{it})$, $\tau_{it} \sim G(v/2, v/2)$ are given by

$$\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{\tau}, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

where $\boldsymbol{\tau} = [\tau_{it}]^\top$, $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{X}_i)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{y}_i)$, $\boldsymbol{V}_i = \sigma^2 \boldsymbol{\Psi}_i + \sigma_b^2 \boldsymbol{i}_{T_i} \boldsymbol{i}_{T_i}^\top$ and $\boldsymbol{\Psi}_i = diag\left\{\tau_{it}^{-1}\right\}$.

$$\boldsymbol{b}_i \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\tau}, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim N(\boldsymbol{b}_{ni}, \boldsymbol{B}_{ni}),$$

where $\boldsymbol{B}_{ni} = (\sigma^{-2}\boldsymbol{W}_i^\top \boldsymbol{\Psi}_i^{-1} \boldsymbol{W}_i + \boldsymbol{D}^{-1})^{-1}$ and $\boldsymbol{b}_{ni} = \boldsymbol{B}_{ni}(\sigma^{-2}\boldsymbol{W}_i^\top \boldsymbol{\Psi}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta}))$.

$$\sigma^2 \mid \boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim IG(\alpha_n, \delta_n),$$

where $\alpha_n = \alpha_0 + \frac{1}{2}\sum_{i=1}^N T_i$ and $\delta_n = \delta_0 + \frac{1}{2}\sum_{i=1}^N (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top \boldsymbol{\Psi}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)$.

$$\boldsymbol{D} \mid \boldsymbol{b} \sim IW(d_n, \boldsymbol{D}_n),$$

where $d_n = d_0 + N$ and $\boldsymbol{D}_n = d_0\boldsymbol{D}_0 + \sum_{i=1}^N \boldsymbol{b}_i\boldsymbol{b}_i^\top$. And

$$\tau_{it} \mid \sigma^2, \boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim G(v_{1n}/2, v_{2ni}/2),$$

where $v_{1n} = v + 1$ and $v_{2ni} = v + \sigma^{-2}(y_{it} - \boldsymbol{x}_{it}^\top\boldsymbol{\beta} - \boldsymbol{w}_{it}^\top\boldsymbol{b}_i)^2$.

The following code implements this simulation, and gets draws of the posterior distributions. We set MCMC iterations, burn-in and thinning parameters equal to 5000, 1000 and 1, respectively. In addition, $\boldsymbol{\beta}_0 = \boldsymbol{0}_5$, $\boldsymbol{B}_0 = \boldsymbol{I}_5$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 2$, $\boldsymbol{D}_0 = \boldsymbol{I}_2$ and $v = 5$.

R code. Simulation exercise: Longitudinal normal model with heteroskedasticity from scratch

```
rm(list = ls()); set.seed(010101)
NT <- 2000; N <- 50
id <- c(1:N, sample(1:N, NT - N,replace=TRUE))
table(id)
x1 <- rnorm(NT); x2 <- rnorm(NT); x3 <- rnorm(NT)
X <- cbind(1, x1, x2, x3); K1 <- dim(X)[2]
w1 <- rnorm(NT); W <- cbind(1, w1)
K2 <- dim(W)[2]; B <- c(0.5, 0.4, 0.6, -0.6)
D <- c(0.7, 0.6)
b1 <- rnorm(N, 0, sd = D[1]^0.5)
b2 <- rnorm(N, 0, sd = D[2]^0.5)
b <- cbind(b1, b2)
v <- 5; tau <- rgamma(NT, shape = v/2, rate = v/2)
sig2 <- 0.1; u <- rnorm(NT, 0, sd = (sig2/tau)^0.5)
y <- NULL
for(i in 1:NT){
  yi <- X[i,]%*%B + W[i,]%*%b[id[i],] + u[i]
  y <- c(y, yi)
}
Data <- as.data.frame(cbind(y, x1, x2, x3, w1, id))
mcmc <- 5000; burnin <- 1000; thin <- 1; tot <- mcmc +
    burnin
b0 <- rep(0, K1); B0 <- diag(K1); B0i <- solve(B0)
r0 <- K2; R0 <- diag(K2); a0 <- 0.001; d0 <- 0.001
PostBeta <- function(sig2, D, tau){
  XVX <- matrix(0, K1, K1)
  XVy <- matrix(0, K1, 1)
  for(i in 1:N){
    ids <- which(id == i)
    Ti <- length(ids)
    Wi <- W[ids, ]
    taui <- tau[ids]
    Vi <- sig2*solve(diag(1/taui)) + Wi%*%D%*%t(Wi)
    ViInv <- solve(Vi)
    Xi <- X[ids, ]
    XVXi <- t(Xi)%*%ViInv%*%Xi
    XVX <- XVX + XVXi
    yi <- y[ids]
    XVyi <- t(Xi)%*%ViInv%*%yi
    XVy <- XVy + XVyi
  }
  Bn <- solve(B0i + XVX)
  bn <- Bn%*%(B0i%*%b0 + XVy)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
```

R code. Simulation exercise: Longitudinal normal model with heteroskedasticity from scratch

```
Postb <- function(Beta, sig2, D, tau){
  Di <- solve(D);   bis <- matrix(0, N, K2)
  for(i in 1:N){
    ids <- which(id == i)
    Wi <- W[ids, ]; Xi <- X[ids, ]
    yi <- y[ids]; taui <- tau[ids]
    Taui <- solve(diag(1/taui))
    Wtei <- sig2^(-1)*t(Wi)%*%Taui%*%(yi - Xi%*%Beta)
    Bni <- solve(sig2^(-1)*t(Wi)%*%Taui%*%Wi + Di)
    bni <- Bni%*%Wtei
    bi <- MASS::mvrnorm(1, bni, Bni)
    bis[i, ] <- bi
  }
  return(bis)
}
PostSig2 <- function(Beta, bs, tau){
  an <- a0 + 0.5*NT
  ete <- 0
  for(i in 1:N){
    ids <- which(id == i)
    Xi <- X[ids, ]; yi <- y[ids]
    Wi <- W[ids, ]; taui <- tau[ids]
    Taui <- solve(diag(1/taui))
    ei <- yi - Xi%*%Beta - Wi%*%bs[i, ]
    etei <- t(ei)%*%Taui%*%ei
    ete <- ete + etei
  }
  dn <- d0 + 0.5*ete
  sig2 <- MCMCpack::rinvgamma(1, shape = an, scale = dn)
  return(sig2)
}
PostD <- function(bs){
  rn <- r0 + N
  btb <- matrix(0, K2, K2)
  for(i in 1:N){
    bsi <- bs[i, ]
    btbi <- bsi%*%t(bsi)
    btb <- btb + btbi
  }
  Rn <- d0*R0 + btb
  Sigma <- MCMCpack::riwish(v = rn, S = Rn)
  return(Sigma)
}
PostTau <- function(sig2, Beta, bs){
  v1n <- v + 1
  v2n <- NULL
  for(i in 1:NT){
    Xi <- X[i, ]; yi <- y[i]
    Wi <- W[i, ]; bi <- bs[id[i],]
    v2ni <- v + sig2^(-1)*(yi - Xi%*%Beta - Wi%*%bi)^2
    v2n <- c(v2n, v2ni)
  }
  tau <- rgamma(NT, shape = rep(v1n/2, NT), rate = v2n/2)
  return(tau)
}
```

R code. Simulation exercise: Longitudinal normal model with heteroskedasticity from scratch

```
PostBetas <- matrix(0, tot, K1); PostDs <- matrix(0, tot, K2
    *(K2+1)/2)
PostSig2s <- rep(0, tot); Postbs <- array(0, c(N, K2, tot))
PostTaus <- matrix(0, tot, NT); RegLS <- lm(y ~ X - 1)
SumLS <- summary(RegLS)
Beta <- SumLS[["coefficients"]][,1]
sig2 <- SumLS[["sigma"]]^2; D <- diag(K2)
tau <- rgamma(NT, shape = v/2, rate = v/2)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  bs <- Postb(Beta = Beta, sig2 = sig2, D = D, tau = tau)
  D <- PostD(bs = bs)
  Beta <- PostBeta(sig2 = sig2, D = D, tau = tau)
  sig2 <- PostSig2(Beta = Beta, bs = bs, tau = tau)
  tau <- PostTau(sig2 = sig2, Beta = Beta, bs = bs)
  PostBetas[s,] <- Beta
  PostDs[s,] <- matrixcalc::vech(D)
  PostSig2s[s] <- sig2
  Postbs[, , s] <- bs
  PostTaus[s,] <- tau
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]; Ds <- PostDs[keep,]
bs <- Postbs[, , keep]; sig2s <- PostSig2s[keep]
taus <- PostTaus[keep,]
summary(coda::mcmc(Bs))
Quantiles for each variable:
          2.5%     25%     50%     75%   97.5%
var1  0.07833  0.2412  0.3232  0.4022  0.5619
var2  0.34101  0.3598  0.3697  0.3793  0.3988
var3  0.59596  0.6150  0.6251  0.6351  0.6574
var4 -0.63722 -0.6165 -0.6067 -0.5966 -0.5785
summary(coda::mcmc(Ds))
Quantiles for each variable:
       2.5%     25%      50%      75%   97.5%
var1  0.4720  0.5995  0.68858  0.79206 1.05285
var2 -0.2721 -0.1405 -0.08185 -0.02482 0.09186
var3  0.3689  0.4644  0.52978  0.60946 0.81999
summary(coda::mcmc(sig2s))
 Quantiles for each variable:
2.5%    25%    50%    75%  97.5%
0.1022 0.1157 0.1324 0.1683 0.3217
```

We can see that all the 95% credible intervals encompass the population parameters, except for the second *fixed effect* and the variance of the model, but both for a tiny margin.

9.2 Logit model

We can use the framework of Section 9.1 to perform inference in models with longitudinal/panel data of binary response variables. In particular, let $y_{it} \sim B(\pi_{it})$, where $\text{logit}(\pi_{it}) = \log\left(\frac{\pi_{it}}{1-\pi_{it}}\right) \equiv y_{it}^*$, such that $y_{it}^* \sim N(\boldsymbol{x}_{it}^\top \boldsymbol{\beta} + \boldsymbol{w}_{it}^\top \boldsymbol{b}_i, \sigma^2)$. Thus, we can *augment* the model with the latent variable y_{it}^* and perform inference using a Metropolis-within-Gibbs sampling algorithm based on the posterior conditional distributions from the previous section.

We can implement a Gibbs sampling algorithm to sample draws from the posterior conditional distributions of $\boldsymbol{\beta}$, σ^2, $\boldsymbol{b}_i$, and $\boldsymbol{D}$ using the equations in Section 9.1 conditional on $\boldsymbol{y}_i^*$. Then, we can use a random walk Metropolis-Hastings algorithm to sample y_{it}^*, where the proposal distribution is Gaussian with mean y_{it}^* and variance v^2, that is, $y_{it}^{*c} = y_{it}^* + \epsilon_{it}$, where $\epsilon_{it} \sim \mathcal{N}(0, v^2)$, and v is a tuning parameter to achieve good acceptance rates.

Finally, for making predictions, we should take into account that $\mathbb{E}[\pi_{it}] = \frac{1}{1+\exp\left\{(\boldsymbol{x}_{it}^\top \boldsymbol{\beta} + \boldsymbol{w}_{it}^\top \boldsymbol{b}_i)/\sqrt{1+\left(\frac{16\sqrt{3}}{15\pi}\right)^2 \sigma^2}\right\}}$ [112, pag. 136].

The posterior distribution of this model is

$$
\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{b}_i, \boldsymbol{D}, \boldsymbol{y}^* \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) \propto \prod_{i=1}^{N} \prod_{t=1}^{T_i} \Big\{ & \pi_{it}^{y_{it}} (1-\pi_{it})^{1-y_{it}} \\
& \times (\sigma^2)^{-1} \exp\left\{ -\frac{1}{2\sigma^2} (y_{it}^* - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i)^\top (y_{it}^* - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i) \right\} \Big\} \\
& \times \exp\left\{ -\frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_0) \right\} \\
& \times \exp\left\{ -\frac{1}{2} \sum_{i=1}^{N} \boldsymbol{b}_i^\top \boldsymbol{D}^{-1} \boldsymbol{b}_i \right\} \\
& \times (\sigma^2)^{-\alpha_0 - 1} \exp\left\{ -\frac{\delta_0}{\sigma^2} \right\} \\
& \times |\boldsymbol{D}|^{-(d_0 + K_2 + 1)/2} \exp\left\{ -\frac{1}{2} tr(d_0 \boldsymbol{D}_0 \boldsymbol{D}^{-1}) \right\}.
\end{aligned}
$$

We can get samples of y_{it}^* from a normal distribution with mean equal to $\boldsymbol{x}_{it}^\top \boldsymbol{\beta} + \boldsymbol{w}_{it}^\top \boldsymbol{b}_i$ and variance σ^2, and use these samples to get $\pi_{it} = \frac{1}{1+e^{-y_{it}^*}}$,

$y_{it}^{*c} = y_{it}^{*} + \epsilon_{it}$ and $\pi_{it}^{c} = \frac{1}{1+e^{-y_{it}^{*c}}}$, and calculate the acceptance rate of the Metropolis-Hastings algorithm,

$$\alpha = \min\left(1, \frac{\pi_{it}^{c y_{it}}(1-\pi_{it}^{c})^{(1-y_{it})} \times \exp\left\{-\frac{1}{2\sigma^2}(y_{it}^{c*} - \boldsymbol{x}_{it}^{\top}\boldsymbol{\beta} - \boldsymbol{w}_{it}^{\top}\boldsymbol{b}_i)^{\top}(y_{it}^{c*} - \boldsymbol{x}_{it}^{\top}\boldsymbol{\beta} - \boldsymbol{w}_{it}^{\top}\boldsymbol{b}_i)\right\}}{\pi_{it}^{y_{it}}(1-\pi_{it})^{(1-y_{it})} \times \exp\left\{-\frac{1}{2\sigma^2}(y_{it}^{*} - \boldsymbol{x}_{it}^{\top}\boldsymbol{\beta} - \boldsymbol{w}_{it}^{\top}\boldsymbol{b}_i)^{\top}(y_{it}^{*} - \boldsymbol{x}_{it}^{\top}\boldsymbol{\beta} - \boldsymbol{w}_{it}^{\top}\boldsymbol{b}_i)\right\}}\right).$$

Example: Doctor visits in Germany

We used the dataset *9VisitDoc.csv* provided by [417][2]. We analyze the determinants of a binary variable (DocVis), which equals 1 if an individual visited a physician in the last three months and 0 otherwise. The dataset contains 32,837 observations of 9,197 individuals in an *unbalanced longitudinal/panel* dataset over the years 1995–1999 from the German Socioeconomic Panel Data.

The specification is given by

$$\begin{aligned}\text{logit}(\pi_{it}) = \beta_1 &+ \beta_2\text{Age} + \beta_3\text{Male} + \beta_4\text{Sport} + \beta_5\text{LogInc} \\ &+ \beta_6\text{GoodHealth} + \beta_7\text{BadHealth} + b_i + b_{i1}\text{Sozh},\end{aligned}$$

where $\pi_{it} = p(\text{DocVis}_{it} = 1)$.

This specification controls for age, a gender indicator (with 1 representing male), whether the individual practices any sport (with 1 for sport), the logarithm of monthly gross income, and self-perception of health status, where "good" and "bad" are compared to a baseline of "regular". Additionally, we assume that unobserved heterogeneity is linked to whether the individual receives welfare payments (with Sozh equal to 1 for receiving welfare).

We set 10,000 MCMC iterations, plus 1,000 burn-in, and a thinning parameter equal to 10. In addition, $\boldsymbol{\beta}_0 = \mathbf{0}_7$, $\boldsymbol{B}_0 = \boldsymbol{I}_7$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 5$, and $\boldsymbol{D}_0 = \boldsymbol{I}_2$.

The Algorithm A26 shows how to perform inference of the hierarchical longitudinal logit model using our GUI. We show in the following code how to perform inference of this example using the command *MCMChlogit* from the *MCMCpack* package. We fixed the variance for over-dispersion (σ^2) setting *FixOD = 1* in this example. Our GUI does not fix this value, that is, it sets *FixOD = 0*, which is the default value in the command *MCMChlogit*. We ask to replicate this example using our GUI in Exercise 4. The command *MCMChlogit* uses an adaptive algorithm to tune v based on an optimal acceptance rate equal to 0.44.

[2]See *http://qed.econ.queensu.ca/jae/2004-v19.4/winkelmann/* for details

R code. Doctor visits in Germany

```
rm(list = ls())
set.seed(12345)
Data <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/9VisitDoc.csv", sep = ",", header = TRUE, quote = "")
attach(Data)
K1 <- 7; K2 <- 2; N <- 9197
b0 <- rep(0, K1); B0 <- diag(K1)
r0 <- 5; R0 <- diag(K2)
a0 <- 0.001; d0 <- 0.001
RegLogit <- glm(DocVis ~ Age + Male + Sport + LogInc + GoodHealth + BadHealth, family = binomial(link = "logit"))
SumLogit <- summary(RegLogit)
Beta0 <- SumLogit[["coefficients"]][,1]
mcmc <- 10000; burnin <- 1000; thin <- 10
# MCMChlogit
Resultshlogit <- MCMCpack::MCMChlogit(fixed = DocVis ~ Age + Male + Sport + LogInc + GoodHealth + BadHealth, random = ~Sozh, group="id", data = Data, burnin = burnin, mcmc = mcmc, thin = thin, mubeta = b0, Vbeta = B0, r = r0, R = R0, nu = a0, delta = d0, beta.start = Beta0, FixOD = 1)
Betas <- Resultshlogit[["mcmc"]][,1:K1]
Sigma2RanEff <- Resultshlogit[["mcmc"]][,c(K2*N+K1+1, 2*N+K1+K2^2)]
summary(Betas)
Quantiles for each variable:
            2.5%       25%       50%      75%    97.5%
beta.(Intercept) -1.1085 -0.6428 -0.4169 -0.166  0.280
beta.Age          0.0051  0.0078  0.0095  0.010  0.013
beta.Male        -1.1914 -1.1325 -1.0981 -1.065 -1.008
beta.Sport        0.2256  0.2846  0.3159  0.348  0.401
beta.LogInc       0.1782  0.2357  0.2661  0.299  0.367
beta.GoodHealth  -1.1648 -1.1046 -1.0701 -1.040 -0.983
beta.BadHealth    1.2233  1.3242  1.3716  1.426  1.533
summary(Sigma2RanEff)
Quantiles for each variable:
                              2.5%    25%   50%   75% 97.5%
VCV.(Intercept).(Intercept) 2.0749 2.1709 2.238 2.303 2.422
VCV.Sozh.Sozh               0.3536 0.4875 0.626 0.906 1.271
```

The results suggest that age, sports, income and a bad perception of health status increase the probability of visiting the physician, the posterior estimates have 95% symmetric credible intervals equal to (5.1e-03, 1.3e-02), (0.23, 0.40), (0.18, 0.37) and (1.22, 1.53), whereas men have a lower probability of visiting a physician, the 95% credible interval is (-1.19, -1.01), and individuals who

have a good perception of their health status also have a lower probability of visiting the doctor, the 95% credible interval is (-1.16, -0.98). The 95% credible interval of the variances of the unobserved heterogeneity associated with the welfare program is (0.35, 1.27).

Algorithm A26 Hierarchical longitudinal logit models

1: Select *Hierarchical Longitudinal Model* on the top panel
2: Select *Logit* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Write down the formula of the *fixed effects* equation in the **Main Equation: Fixed Effects** box. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation
6: Write down the formula of the *random effects* equation in the **Main Equation: Random Effects** box without writing the dependent variable, that is, starting the equation with the *tilde* ("$\sim$") symbol. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation. If there are just random intercepts do not write anything in this box
7: Write down the name of the grouping variable, that is, the variable that indicates the cross-sectional units
8: Set the hyperparameters of the *fixed effects*: mean vector, covariance matrix, shape and scale parameters. This step is not necessary as by default our GUI uses non-informative priors
9: Set the hyperparameters of the *random effects*: degrees of freedom and scale matrix of the inverse Wishart distribution. This step is not necessary as by default our GUI uses non-informative priors
10: Click the *Go!* button
11: Analyze results
12: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

9.3 Poisson model

We can use same ideas as in Section 9.2 to perform inference in longitudinal/panel datasets where the dependent variable takes non-negative integers. Let's assume that $y_{it} \sim P(\lambda_{it})$ where $\log(\lambda_{it}) = y_{it}^*$ such that $y_{it}^* \sim N(\boldsymbol{x}_{it}^\top \boldsymbol{\beta} + \boldsymbol{w}_{it}^\top \boldsymbol{b}_i, \sigma^2)$. We can *augment* the model with the latent variable y_{it}^*, and again use a Metropolis-within-Gibbs algorithm to perform inference in this model.

The posterior distribution of this model is

$$\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{b}_i, \boldsymbol{D}, \boldsymbol{y}^* \mid \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) \propto \prod_{i=1}^{N} \prod_{t=1}^{T_i} \{ & \lambda_{it}^{y_{it}} \exp\{-\lambda_{it}\} \\
& \times (\sigma^2)^{-1} \exp\left\{ -\frac{1}{2\sigma^2} (y_{it}^* - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i)^\top (y_{it}^* - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i) \right\} \Bigg\} \\
& \times \exp\left\{ -\frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_0) \right\} \\
& \times \exp\left\{ -\frac{1}{2} \sum_{i=1}^{N} \boldsymbol{b}_i^\top \boldsymbol{D}^{-1} \boldsymbol{b}_i \right\} \\
& \times (\sigma^2)^{-\alpha_0 - 1} \exp\left\{ -\frac{\delta_0}{\sigma^2} \right\} \\
& \times |\boldsymbol{D}|^{-(d_0 + K_2 + 1)/2} \exp\left\{ -\frac{1}{2} tr(d_0 \boldsymbol{D}_0 \boldsymbol{D}^{-1}) \right\}.
\end{aligned}$$

We can get samples of y_{it}^* from a normal distribution with mean equal to $\boldsymbol{x}_{it}^\top \boldsymbol{\beta} + \boldsymbol{w}_{it}^\top \boldsymbol{b}_i$ and variance σ^2, and use these samples to get $\lambda_{it} = \exp(y_{it}^*)$, $y_{it}^{*c} = y_{it}^* + \epsilon_{it}$, where $\epsilon_{it} \sim \mathcal{N}(0, v^2)$, v is a tuning parameter to get good acceptance rates, and $\lambda_{it}^c = \exp(y_{it}^{*c})$. The acceptance rate of the Metropolis-Hastings algorithm is

$$\alpha = \min\left(1, \frac{\lambda_{it}^{c y_{it}} \exp(-\lambda_{it}^c) \times \exp\left\{ -\frac{1}{2\sigma^2} (y_{it}^{c*} - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i)^\top (y_{it}^{c*} - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i) \right\}}{\lambda_{it}^{y_{it}} \exp(-\lambda_{it}) \times \exp\left\{ -\frac{1}{2\sigma^2} (y_{it}^* - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i)^\top (y_{it}^* - \boldsymbol{x}_{it}^\top \boldsymbol{\beta} - \boldsymbol{w}_{it}^\top \boldsymbol{b}_i) \right\}} \right).$$

In addition, we should use the posterior conditional distributions from Section 9.1 to complete the algorithm getting samples of $\boldsymbol{\beta}$, σ^2, $\boldsymbol{b}_i$ and $\boldsymbol{D}$ replacing y_{it} by y_{it}^*.

We should take into account for doing predictions that $\mathbb{E}[\lambda_{it}] = \exp\left\{ \boldsymbol{x}_{it}^\top \boldsymbol{\beta} + \boldsymbol{w}_{it}^\top \boldsymbol{b}_i + 0.5\sigma^2 \right\}$ [112, pag. 137].

Example: Simulation exercise

Let's perform a simulation exercise to assess the performance of the hierarchical longitudinal Poisson model. The point of departure is to assume that

$$y_{it}^* = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \beta_3 x_{it3} + b_i + w_{it1} b_{i1},$$

where $x_{itk} \sim N(0,1)$ for $k = 1,2,3$, $w_{it1} \sim N(0,1)$, $b_i \sim N(0, 0.7^{1/2})$, $b_{i1} \sim N(0, 0.6^{1/2})$, and $\boldsymbol{\beta} = [0.5\ 0.4\ 0.6\ \ -0.6]^\top$, with $i = 1,2,\ldots,50$. Additionally, $y_{it} \sim P(\lambda_{it})$, where $\lambda_{it} = \exp(y^*_{it})$. The sample size is 1000 in an *unbalanced panel structure*.

We set the priors as $\boldsymbol{\beta}_0 = \mathbf{0}_4$, $\boldsymbol{B}_0 = \boldsymbol{I}_4$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 2$, and $\boldsymbol{D}_0 = \boldsymbol{I}_2$. The number of MCMC iterations, burn-in, and thinning parameters are 15,000, 5,000, and 10, respectively.

The following code shows how to perform inference in the hierarchical longitudinal Poisson model programming the Metropolis-within-Gibbs sampler.

R code. Simulation exercise: Hierarchical longitudinal Poisson model

```
rm(list = ls()); set.seed(010101)
NT <- 1000; N <- 50
id <- c(1:N, sample(1:N, NT - N,replace=TRUE))
x1 <- rnorm(NT); x2 <- rnorm(NT); x3 <- rnorm(NT)
X <- cbind(1, x1, x2, x3)
K1 <- dim(X)[2]; w1 <- rnorm(NT)
W <- cbind(1, w1); K2 <- dim(W)[2]
B <- c(0.5, 0.4, 0.6, -0.6)
D <- c(0.7, 0.6); sig2 <- 0.1
b1 <- rnorm(N, 0, sd = D[1]^0.5)
b2 <- rnorm(N, 0, sd = D[2]^0.5)
b <- cbind(b1, b2)
yl <- NULL
for(i in 1:NT){
  ylmeani <- X[i,]%*%B + W[i,]%*%b[id[i],]
  yli <- rnorm(1, ylmeani, sig2^0.5)
  yl <- c(yl, yli)
}
lambdait <- exp(yl); y <- rpois(NT, lambdait)
Data <- as.data.frame(cbind(y, x1, x2, x3, w1, id))
mcmc <- 15000; burnin <- 5000; thin <- 10; tot <- mcmc +
    burnin
b0 <- rep(0, K1); B0 <- diag(K1); B0i <- solve(B0)
r0 <- K2; R0 <- diag(K2); a0 <- 0.001; d0 <- 0.001
LatentMHV1 <- function(tuning, Beta, bs, sig2){
  ylhat <- rep(0, NT)
  accept <- NULL
  for(i in 1:NT){
    ids <- which(id == i)
    yi <- y[i]
    ylhatmeani <- X[i,]%*%Beta + W[i,]%*%bs[id[i],]
    ylhati <- rnorm(1, ylhatmeani, sd = sig2^0.5)
    lambdahati <- exp(ylhati)
    ei <- rnorm(1, 0, sd = tuning)
    ylpropi <- ylhati + ei
    lambdapropi <- exp(ylpropi)
    logPosthati <- sum(dpois(yi, lambdahati, log = TRUE) +
    dnorm(ylhati, ylhatmeani, sig2^0.5, log = TRUE))
    logPostpropi <- sum(dpois(yi, lambdapropi, log = TRUE) +
     dnorm(ylpropi, ylhatmeani, sig2^0.5, log = TRUE))
    alphai <- min(1, exp(logPostpropi - logPosthati))
    ui <- runif(1)
    if(ui <= alphai){
      ylhati <- ylpropi; accepti <- 1
    }else{
      ylhati <- ylhati; accepti <- 0
    }
    ylhat[i] <- ylhati
    accept <- c(accept, accepti)
  }
  res <- list(ylhat = ylhat, accept = mean(accept))
  return(res)
}
```

R code. Simulation exercise: Hierarchical longitudinal Poisson model

```
PostBeta <- function(D, ylhat, sig2){
  XVX <- matrix(0, K1, K1); XVy <- matrix(0, K1, 1)
  for(i in 1:N){
    ids <- which(id == i); Ti <- length(ids)
    Wi <- W[ids, ]
    Vi <- diag(Ti)*sig2 + Wi%*%D%*%t(Wi)
    ViInv <- solve(Vi); Xi <- X[ids, ]
    XVXi <- t(Xi)%*%ViInv%*%Xi
    XVX <- XVX + XVXi
    yi <- ylhat[ids]
    XVyi <- t(Xi)%*%ViInv%*%yi
    XVy <- XVy + XVyi
  }
  Bn <- solve(B0i + XVX); bn <- Bn%*%(B0i%*%b0 + XVy)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
Postb <- function(Beta, D, ylhat, sig2){
  Di <- solve(D); bis <- matrix(0, N, K2)
  for(i in 1:N){
    ids <- which(id == i)
    Wi <- W[ids, ]; Xi <- X[ids, ]
    yi <- ylhat[ids]
    Wtei <- sig2^(-1)*t(Wi)%*%(yi - Xi%*%Beta)
    Bni <- solve(sig2^(-1)*t(Wi)%*%Wi + Di)
    bni <- Bni%*%Wtei
    bi <- MASS::mvrnorm(1, bni, Bni)
    bis[i, ] <- bi
  }
  return(bis)
}
PostD <- function(bs){
  rn <- r0 + N; btb <- matrix(0, K2, K2)
  for(i in 1:N){
    bsi <- bs[i, ]; btbi <- bsi%*%t(bsi)
    btb <- btb + btbi
  }
  Rn <- d0*R0 + btb
  Sigma <- MCMCpack::riwish(v = rn, S = Rn)
  return(Sigma)
}
PostSig2 <- function(Beta, bs, ylhat){
  an <- a0 + 0.5*NT; ete <- 0
  for(i in 1:N){
    ids <- which(id == i)
    Xi <- X[ids, ]
    yi <- ylhat[ids]
    Wi <- W[ids, ]
    ei <- yi - Xi%*%Beta - Wi%*%bs[i, ]
    etei <- t(ei)%*%ei
    ete <- ete + etei
  }
  dn <- d0 + 0.5*ete
  sig2 <- MCMCpack::rinvgamma(1, shape = an, scale = dn)
  return(sig2)
}
```

R code. Simulation exercise: Hierarchical longitudinal Poisson model

```
PostBetas <- matrix(0, tot, K1); PostDs <- matrix(0, tot, K2*(K2+1)/2)
Postbs <- array(0, c(N, K2, tot)); PostSig2s <- rep(0, tot)
Accepts <- rep(NULL, tot)
RegPois <- glm(y ~ X - 1, family = poisson(link = "log"))
SumPois <- summary(RegPois)
Beta <- SumPois[["coefficients"]][,1]
sig2 <- sum(SumPois[["deviance.resid"]]^2)/SumPois[["df.residual"]]
D <- diag(K2); bs1 <- rnorm(N, 0, sd = D[1,1]^0.5)
bs2 <- rnorm(N, 0, sd = D[2,2]^0.5); bs <- cbind(bs1, bs2)
tuning <- 0.1; ropt <- 0.44
tunepariter <- seq(round(tot/10, 0), tot, round(tot/10, 0)); l <- 1
pb <- winProgressBar(title = "progress bar", min = 0, max = tot, width = 300)
for(s in 1:tot){
  LatY <- LatentMHV1(tuning = tuning, Beta = Beta, bs = bs, sig2 = sig2)
  ylhat <- LatY[["ylhat"]]
  bs <- Postb(Beta = Beta, D = D, ylhat=ylhat, sig2 = sig2)
  D <- PostD(bs = bs)
  Beta <- PostBeta(D = D, ylhat = ylhat, sig2 = sig2)
  sig2 <- PostSig2(Beta = Beta, bs = bs, ylhat = ylhat)
  PostBetas[s,] <- Beta
  PostDs[s,] <- matrixcalc::vech(D)
  Postbs[, , s] <- bs; PostSig2s[s] <- sig2
  AcceptRate <- LatY[["accept"]]
  Accepts[s] <- AcceptRate
  if(AcceptRate > ropt){
    tuning = tuning*(2-(1-AcceptRate)/(1-ropt))
  }else{
    tuning = tuning/(2-AcceptRate/ropt)
  }
  if(s == tunepariter[l]){
    print(AcceptRate); l <- l + 1
  }
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0), "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]; Ds <- PostDs[keep,]
bs <- Postbs[, , keep]; sig2s <- PostSig2s[keep]
summary(coda::mcmc(Bs))
 Quantiles for each variable:
       2.5%     25%     50%     75%   97.5%
var1  0.1038  0.3803  0.5199  0.6534  0.9259
var2  0.2432  0.3166  0.3608  0.4003  0.4796
var3  0.4213  0.5017  0.5453  0.5885  0.6682
var4 -0.7038 -0.6149 -0.5729 -0.5269 -0.4459
summary(coda::mcmc(Ds))
Quantiles for each variable:
       2.5%     25%     50%      75%   97.5%
var1  0.3331  0.4788  0.5732  0.69316 0.99135
var2 -0.3354 -0.1926 -0.1277 -0.06692 0.03674
var3  0.1252  0.2182  0.2780  0.34731 0.51055
```

We can see that all 95% credible intervals encompass the population parameters of the *fixed effects*, the posterior medians are relatively near the population values. However, we do not get good posterior estimates of the covariance matrix of the *random effects* as the 95% credible intervals do not encompass the second element of the diagonal of this matrix. In addition, the posterior draws of this algorithm over-estimates the over-dispersion parameter.

We can perform inference for the hierarchical longitudinal Poisson model in our GUI using Algorithm A27. Our GUI is based on the *MCMChpoisson* command from the *MCMCpack* package.

9.4 Summary

In this chapter, we present how to perform inference in longitudinal/panel data models from a Bayesian perspective. In particular, the Bayesian approach uses a hierarchical structure, where the *random effects* have priors that depend on hyperparameters, which in turn also have priors. We cover the three most common cases: continuous, binary, and count dependent variables. The basic models presented in this chapter can be easily extended to more flexible cases, given the hierarchical structure.

9.5 Exercises

1. Show that the posterior distribution of $\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{D}$ is $N(\boldsymbol{\beta}_n, \boldsymbol{B}_n)$, where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{X}_i)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{y}_i)$.
2. **The relation between productivity and public investment example continues**
   - Perform inference of this example using our GUI.
   - Program from scratch a Gibbs sampling algorithm to perform this application. Set $\boldsymbol{\beta}_0 = \boldsymbol{0}_5$, $\boldsymbol{B}_0 = \boldsymbol{I}_5$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 5$ and $\boldsymbol{D}_0 = \boldsymbol{I}_1$.
   - Perform inference in this example assuming that $\mu_{it} \mid \tau_{it} \sim N(0, \sigma^2/\tau_{it})$ and $\tau_{it} \sim G(v/2, v/2)$ setting $v = 5$.
3. **Simulation exercise of the longitudinal normal model continues**

Algorithm A27 Hierarchical longitudinal Poisson models

1: Select *Hierarchical Longitudinal Model* on the top panel
2: Select *Poisson* model using the left radio button
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Select MCMC iterations, burn-in and thinning parameters using the *Range sliders*
5: Write down the formula of the *fixed effects* equation in the **Main Equation: Fixed Effects** box. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation
6: Write down the formula of the *random effects* equation in the **Main Equation: Random Effects** box without writing the dependent variable, that is, starting the equation with the *tilde* ("∼") symbol. This formula must be written using the syntax of the *formula* command of **R** software. This equation includes intercept by default, do not include it in the equation. If there are just random intercepts do not write anything in this box
7: Write down the name of the grouping variable, that is, the variable that indicates the cross-sectional units
8: Set the hyperparameters of the *fixed effects*: mean vector, covariance matrix, shape and scale parameters. This step is not necessary as by default our GUI uses non-informative priors
9: Set the hyperparameters of the *random effects*: degrees of freedom and scale matrix of the inverse Wishart distribution. This step is not necessary as by default our GUI uses non-informative priors
10: Click the *Go!* button
11: Analyze results
12: Download posterior chains and diagnostic plots using the *Download Posterior Chains* and *Download Posterior Graphs* buttons

*At the time of writing this book there was an issue with the function *MCMChpoisson* from *MCMCpack*. We contact the maintainer, but users may have issues running this algorithm or running this function directly in **R**.

Assume that

$$y_{it} = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \beta_3 x_{it3} + \beta_4 z_{i1} + b_i + w_{it1} b_{i1} + \mu_{it},$$

where $x_{itk} \sim N(0,1)$, $k = 1,2,3$, $z_{i1} \sim B(0.5)$, $w_{it1} \sim N(0,1)$, $b_i \sim N(0, 0.7^{1/2})$, $b_{i1} \sim N(0, 0.6^{1/2})$, $\mu_{it} \sim N(0, 0.1^{1/2})$ $\boldsymbol{\beta} = [0.5\ 0.4\ 0.6\ -0.6\ 0.7]^\top$, $i = 1,2,\ldots,50$, and the sample size is 2,000 in an *unbalanced panel structure*. In addition, we assume that $\boldsymbol{b}_i$ dependents on $\boldsymbol{z}_i = [1\ z_{i1}]^\top$ such that $\boldsymbol{b}_i \sim N(\boldsymbol{Z}_i\boldsymbol{\gamma}, \boldsymbol{D})$ where

$\boldsymbol{Z}_i = \boldsymbol{I}_{K_2} \otimes \boldsymbol{z}_i^\top$, and $\boldsymbol{\gamma} = [1\ 1\ 1\ 1]$. The prior for $\boldsymbol{\gamma}$ is $N(\boldsymbol{\gamma}_0, \boldsymbol{\Gamma}_0)$ where we set $\boldsymbol{\gamma}_0 = \boldsymbol{0}_4$ and $\boldsymbol{\Gamma}_0 = \boldsymbol{I}_4$.

- Perform inference in this model without taking into account the dependence between $\boldsymbol{b}_i$ and z_{i1}, and compare the posterior estimates with the population parameters.
- Perform inference in this model taking into account the dependence between $\boldsymbol{b}_i$ and z_{i1}, and compare the posterior estimates with the population parameters.

4. **Doctor visits in Germany continues I**

   Replicate this example using our GUI, which by default does not fix the over-dispersion parameter (σ^2), and compare the results with the results of this example in Section 9.2.

5. **Simulation exercise of the longitudinal logit model**

   Perform a simulation exercise to assess the performance of the hierarchical longitudinal logit model. The point of departure is to assume that

$$y_{it}^* = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \beta_3 x_{it3} + b_i + w_{it1} b_{i1},$$

   where $x_{itk} \sim N(0,1)$, $k = 1,2,3$, $w_{it1} \sim N(0,1)$, $b_i \sim N(0, 0.7^{1/2})$, $b_{i1} \sim N(0, 0.6^{1/2})$, $\boldsymbol{\beta} = [0.5\ 0.4\ 0.6\ \ -0.6]^\top$, $i = 1,2,\ldots,50$, and $y_{it} \sim B(\pi_{it}$, where $\pi_{it} = 1/(1+\exp(y_{it}^*))$. The sample size is 1,000 in an *unbalanced panel structure*.

   - Perform inference using the command *MCMChlogit* fixing the over-dispersion parameter, and using $\boldsymbol{\beta}_0 = \boldsymbol{0}_4$, $\boldsymbol{B}_0 = \boldsymbol{I}_4$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 2$ and $\boldsymbol{D}_0 = \boldsymbol{I}_2$.
   - Program from scratch a Metropolis-within-Gibbs algorithm to perform inference in this simulation.

6. **Doctor visits in Germany continues II**

   Take a sub-sample of the first 500 individuals of the datatset *9VisitDoc.csv* to perform inference in the number of visits to doctors (*DocNum*) with the same specification of the example of **Doctor visits in Germany** of Section 9.2.

10

Bayesian model averaging

We outline in this chapter a framework for addressing model uncertainty and averaging across different models in a probabilistically consistent manner. The discussion tackles two major computational challenges in Bayesian model averaging: the vast space of possible models and the absence of analytical solutions for the marginal likelihood.

We begin by illustrating the approach within the Gaussian linear model, assuming exogeneity of the regressors, and extend the analysis to cases with endogenous regressors, and dynamic models. Additionally, we adapt the framework to generalized linear models, including the logit, gamma, and Poisson families. Lastly, we explore alternative methods for computing marginal likelihoods, especially when the Bayesian information criterion's asymptotic approximation proves inadequate.

Remember that we can run our GUI typing

R code. How to display our graphical user interface

```
shiny::runGitHub("besmarter/BSTApp", launch.browser = T)
```

in the **R** package console or any **R** code editor, and once our GUI is deployed, select *Bayesian Model Averaging*. However, users should see Chapter 5 for other options and details.

10.1 Foundation

Remember from Chapter 1 that Bayesian model averaging (BMA) is an approach which takes into account model uncertainty. In particular, we consider uncertainty in the regressors (variable selection) in a regression framework

where there are K possible explanatory variables.[1] This implies 2^K potential models indexed by parameters $\boldsymbol{\theta}_m$, $m = 1, 2, \ldots, 2^K$.

Following [377], the posterior model probability is

$$\pi(\mathcal{M}_j|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathcal{M}_j)\pi(\mathcal{M}_j)}{\sum_{m=1}^{2^K} p(\boldsymbol{y}|\mathcal{M}_m)\pi(\mathcal{M}_m)},$$

where $\pi(\mathcal{M}_j)$ is the prior model probability,[2]

$$p(\boldsymbol{y}|\mathcal{M}_j) = \int_{\boldsymbol{\Theta}_j} p(\boldsymbol{y}|\boldsymbol{\theta}_j, \mathcal{M}_j)\pi(\boldsymbol{\theta}_j|\mathcal{M}_j)d\boldsymbol{\theta}_j$$

is the marginal likelihood, and $\pi(\boldsymbol{\theta}_j|\mathcal{M}_j)$ is the prior distribution of $\boldsymbol{\theta}_j$ conditional on model $\mathcal{M}_j$.

Following [322], the posterior distribution of $\boldsymbol{\theta}$ is

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) = \sum_{m=1}^{2^K} \pi(\boldsymbol{\theta}_m|\boldsymbol{y}, \mathcal{M}_m)\pi(\mathcal{M}_m|\boldsymbol{y})$$

The posterior distribution of the parameter vector $\boldsymbol{\theta}$ under model $\mathcal{M}_m$ is denoted as $\pi(\boldsymbol{\theta}_m|\boldsymbol{y}, \mathcal{M}_m)$. The posterior mean of $\boldsymbol{\theta}$ is given by:

$$\mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}] = \sum_{m=1}^{2^K} \hat{\boldsymbol{\theta}}_m \, \pi(\mathcal{M}_m|\boldsymbol{y}),$$

where $\hat{\boldsymbol{\theta}}_m$ represents the posterior mean under model $\mathcal{M}_m$.

The variance of the k-th element of $\boldsymbol{\theta}$ given the data $\boldsymbol{y}$ is:

$$\mathrm{Var}(\theta_{km}|\boldsymbol{y}) = \sum_{m=1}^{2^K} \pi(\mathcal{M}_m|\boldsymbol{y}) \, \widehat{\mathrm{Var}}(\theta_{km}|\boldsymbol{y}, \mathcal{M}_m) + \sum_{m=1}^{2^K} \pi(\mathcal{M}_m|\boldsymbol{y}) \left(\hat{\theta}_{km} - \mathbb{E}[\theta_{km}|\boldsymbol{y}]\right)^2,$$

where $\widehat{\mathrm{Var}}(\theta_{km}|\boldsymbol{y}, \mathcal{M}_m)$ denotes the posterior variance of the k-th element of $\boldsymbol{\theta}$ under model $\mathcal{M}_m$.

The posterior variance highlights how BMA accounts for model uncertainty. The first term represents the weighted variance of each model, averaged across all potential models, while the second term reflects the stability of the estimates across models. The greater the variation in estimates between models, the higher the posterior variance.

[1] Take into account that K can increase when interaction terms and/or polynomial terms of the original control variables are included.

[2] We attach equal prior probabilities to each model in our GUI. However, this choice gives more prior probability to the set of models of medium size (think about the k-th row of Pascal's triangle). An interesting alternative is to use the Beta-Binomial prior proposed by [250].

The posterior predictive distribution is

$$\pi(\boldsymbol{y}_0|\boldsymbol{y}) = \sum_{m=1}^{2^K} p_m(\boldsymbol{y}_0|\boldsymbol{y}, \mathcal{M}_m)\pi(M_m|\boldsymbol{y})$$

where $p_m(\boldsymbol{y}_0|\boldsymbol{y}, \mathcal{M}_m) = \int_{\boldsymbol{\Theta}_m} p(\boldsymbol{y}_0|\boldsymbol{y}, \boldsymbol{\theta}_m, \mathcal{M}_m)\pi(\boldsymbol{\theta}_m|\boldsymbol{y}, \mathcal{M}_m)d\boldsymbol{\theta}_m$ is the posterior predictive distribution under model $\mathcal{M}_m$.

Another important statistic in BMA is the posterior inclusion probability associated with variable $\boldsymbol{x}_k$, $k = 1, 2, \ldots, K$, which is

$$PIP(\boldsymbol{x}_k) = \sum_{m=1}^{2^K} \pi(\mathcal{M}_m|\boldsymbol{y}) \times \mathbb{1}_{k,m},$$

where $\mathbb{1}_{k,m} = \left\{ \begin{array}{lll} 1 & if & \boldsymbol{x}_k \in \mathcal{M}_m \\ \\ 0 & if & \boldsymbol{x}_k \notin \mathcal{M}_m \end{array} \right\}$.

[226] suggest that posterior inclusion probabilities (PIP) less than 0.5 are evidence against the regressor, $0.5 \leq PIP < 0.75$ is weak evidence, $0.75 \leq PIP < 0.95$ is positive evidence, $0.95 \leq PIP < 0.99$ is strong evidence, and $PIP \geq 0.99$ is very strong evidence.

There are two main computational issues in implementing BMA based on variable selection. First, the number of models in the model space is 2^K, which sometimes can be enormous. For instance, three regressors imply just eight models, see Table 10.1, but 40 regressors implies approximately 1.1e+12 models. Take into account that models always include the intercept, and all regressors should be standardized to avoid scale issues.[3] The second computational issue is calculating the marginal likelihood $p(\boldsymbol{y}|\mathcal{M}_j) = \int_{\boldsymbol{\Theta}_j} p(\boldsymbol{y}|\boldsymbol{\theta}_j, \mathcal{M}_j)\pi(\boldsymbol{\theta}_j|\mathcal{M}_j)d\boldsymbol{\theta}_j$, which most of the time does not have an analytic solution.

TABLE 10.1
Space of models: Three regressors.

Regressor	Inclusion							
x_1	1	1	1	1	0	0	0	0
x_2	1	1	0	0	1	1	0	0
x_3	1	0	1	0	1	0	1	0

Notes: "1" indicates inclusion of the regressor, and "0" indicates no inclusion. The space of models is composed by 8 models. The model always includes intercept.

The first computational issue is basically a problem of ranking models. This can be tackled using different approaches, such as Occam's window criterion [263, 324], reversible jump Markov chain Monte Carlo computation

[3]Scaling variables is always an important step in variable selection.

[171], Markov chain Monte Carlo model composition [264], and multiple testing using intrinsic priors [68] or nonlocal prior densities [213]. We focus on Occam's window and Markov chain Monte Carlo model composition in our GUI.[4]

In Occam's window, a model is discarded if its predictive performance is much worse than that of the best model [263, 324]. Thus, models not belonging to $\mathcal{M}' = \left\{\mathcal{M}_j : \frac{\max_m \pi(\mathcal{M}_m|\boldsymbol{y})}{\pi(\mathcal{M}_j|\boldsymbol{y})} \leq c\right\}$ should be discarded, where c is chosen by the user ([263] propose $c = 20$). In addition, complicated models than are less supported by the data than simpler models are also discarded, that is, $\mathcal{M}'' = \left\{\mathcal{M}_j : \exists \mathcal{M}_m \in \mathcal{M}', \mathcal{M}_m \subset \mathcal{M}_j, \frac{\pi(\mathcal{M}_m|\boldsymbol{y})}{\pi(\mathcal{M}_j|\boldsymbol{y})} > 1\right\}$. Then, the set of models used in BMA is $\mathcal{M}^* = \mathcal{M}' \cap \mathcal{M}''^c \in \mathcal{M}$. [324] find that the number of models in $\mathcal{M}^*$ is normally less than 25.

However, the previous theoretical framework requires finding the model with the maximum a posteriori model probability ($\max_m \pi(\mathcal{M}_m|\boldsymbol{y})$), which implies calculating all possible models in $\mathcal{M}$. This is computationally burdensome. Hence, a heuristic approach is proposed by [321] based on ideas of [263]. The search strategy is based on a series of nested comparisons of ratios of posterior model probabilities. Let $\mathcal{M}_0$ be a model with one regressor less than model $\mathcal{M}_1$, then:

- If $\log(\pi(\mathcal{M}_0|\boldsymbol{y})/\pi(\mathcal{M}_1|\boldsymbol{y})) > \log(O_R)$, then $\mathcal{M}_1$ is rejected and $\mathcal{M}_0$ is considered.
- If $\log(\pi(\mathcal{M}_0|\boldsymbol{y})/\pi(\mathcal{M}_1|\boldsymbol{y})) \leq -\log(O_L)$, then $\mathcal{M}_0$ is rejected, and $\mathcal{M}_1$ is considered.
- If $\log(O_L) < \log(\pi(\mathcal{M}_0|\boldsymbol{y})/\pi(\mathcal{M}_1|\boldsymbol{y})) \leq \log(O_R)$, $\mathcal{M}_0$ and $\mathcal{M}_1$ are considered.

Here O_R is a number specifying the maximum ratio for excluding models in Occam's window, and $O_L = 1/O_R^2$ is defined by default in [321]. The search strategy can be "up", adding one regressor, or "down", dropping one regressor (see [263] for details about the down and up algorithms). The leaps and bounds algorithm [143] is implemented to improve the computational efficiency of this search strategy [321]. Once the set of potentially acceptable models is defined, we discard all the models that are not in $\mathcal{M}'$, and the models that are in $\mathcal{M}''$ where 1 is replaced by $\exp\{O_R\}$ due to the leaps and bounds algorithm giving an approximation to BIC, so as to ensure that no good models are discarded.

The second approach that we consider in our GUI to tackle the model space size issue is Markov chain Monte Carlo model composition (MC3) [265]. In particular, given the space of models $\mathcal{M}_m$, we simulate a chain of $\mathcal{M}_s$ models, $s = 1, 2, ..., S << 2^K$, where the algorithm randomly extracts a candidate

[4]Variable selection (model selection or regularization) is a topic related to model uncertainty. Approaches such as stochastic search variable selection (spike and slab) [156, 157] and Bayesian Lasso [304] are good examples of how to tackle this issue. See Chapter 12.

model $\mathcal{M}_c$ from a neighborhood of models ($nbd(\mathcal{M}_m)$) that consists of the actual model itself and the set of models with either one variable more or one variable less [324]. Therefore, there is a transition kernel in the space of models $q(\mathcal{M}_m \to \mathcal{M}_c)$, such that $q(\mathcal{M}_m \to \mathcal{M}_c) = 0 \ \forall \mathcal{M}_c \notin nbd(\mathcal{M}_m)$ and $q(\mathcal{M}_m \to \mathcal{M}_c) = \frac{1}{|nbd(\mathcal{M}_m)|} \ \forall \mathcal{M}_m \in nbd(\mathcal{M}_m)$, $|nbd(\mathcal{M}_m)|$ being the number of neighbors of $\mathcal{M}_m$. This candidate model is accepted with probability

$$\alpha(\mathcal{M}_{s-1}, \mathcal{M}_c) = \min\left\{\frac{|nbd(\mathcal{M}_m)|p(\boldsymbol{y}|\mathcal{M}_c)\pi(\mathcal{M}_c)}{|nbd(\mathcal{M}^c)|p(\boldsymbol{y}|\mathcal{M}_{(s-1)})\pi(\mathcal{M}_{(s-1)})}, 1\right\}.$$

Observe that by construction $|nbd(\mathcal{M}_m)| = |nbd(\mathcal{M}_c)| = k$, except in extreme cases where a model has only one regressor or has all regressors.

The Bayesian information criterion is a possible solution for the second computational issue in BMA, that is, calculating the marginal likelihood when there is no an analytic solution. Defining $h(\boldsymbol{\theta}|\mathcal{M}_j) = -\frac{\log(p(\boldsymbol{y}|\boldsymbol{\theta}_j,\mathcal{M}_j)\pi(\boldsymbol{\theta}_j|\mathcal{M}_j))}{N}$, then $p(\boldsymbol{y}|\mathcal{M}_j) = \int_{\boldsymbol{\Theta}_j} \exp\{-Nh(\boldsymbol{\theta}|\mathcal{M}_j)\}\, d\boldsymbol{\theta}_j$. If N is sufficiently large (technically $N \to \infty$), we can make the following assumptions [194]:

- We can use the Laplace method for approximating integrals [399].
- The posterior mode is reached at the same point as the maximum likelihood estimator (MLE), denoted by $\hat{\boldsymbol{\theta}}_{MLE}$.

We get the following results under these assumptions:

$$p(\boldsymbol{y}|\mathcal{M}_j) \approx \left(\frac{2\pi}{N}\right)^{K_j/2} |\boldsymbol{\Sigma}|^{-1/2} \exp\left\{-Nh(\hat{\boldsymbol{\theta}}_j^{MLE}|\mathcal{M}_j)\right\},\ N \to \infty,$$

where $\boldsymbol{\Sigma}$ is the inverse of the Hessian matrix of $h(\hat{\boldsymbol{\theta}}_j^{MLE}|\mathcal{M}_j)$, and $K_j = dim\{\boldsymbol{\theta}_j\}$.

This implies

$$\begin{aligned}\log\left(p(\boldsymbol{y}|\mathcal{M}_j)\right) \approx & \frac{K_j}{2}\log(2\pi) - \frac{K_j}{2}\log(N) - \frac{1}{2}\log(|\boldsymbol{\Sigma}|) + \log(p(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_j^{MLE}, \mathcal{M}_j)) \\ & + \log(\pi(\hat{\boldsymbol{\theta}}_j^{MLE}|\mathcal{M}_j)),\ N \to \infty.\end{aligned}$$

Since $\frac{K_j}{2}\log(2\pi)$ and $\log(\pi(\hat{\boldsymbol{\theta}}_j^{MLE}|\mathcal{M}_j))$ are constants as functions of $\boldsymbol{y}$, and $|\boldsymbol{\Sigma}|$ is bounded by a finite constant, we have

$$log\left(p(\boldsymbol{y}|\mathcal{M}_j)\right) \approx -\frac{K_j}{2}\log(N) + \log(p(\boldsymbol{y}|\hat{\boldsymbol{\theta}}_j^{MLE}, \mathcal{M}_j)) = -\frac{BIC}{2},\ N \to \infty.$$

The marginal likelihood thus asymptotically converges to a linear transformation of the Bayesian Information Criterion (BIC), significantly simplifying its calculation. In addition, the BIC is consistent, that is, the probability of uncovering the population statistical model converges to one as the sample size

converges to infinity given a $\mathcal{M}$-closed view [38, Chap. 6], that is, one of the models in consideration is the population statistical model (data generating process) [366, 60]. In case that there is an $\mathcal{M}$-completed view of nature, that is, there is a true data generating process, but the space of models that we are comparing does not include it, the BIC asymptotically selects the model that minimizes the Kullback-Leiber (KL) divergence to the true (population) model [94, Chap. 4].

10.2 The Gaussian linear model

The Gaussian linear model specifies $\boldsymbol{y} = \alpha \boldsymbol{i}_N + \boldsymbol{X}_m \boldsymbol{\beta}_m + \boldsymbol{\mu}_m$ such that $\boldsymbol{\mu}_m \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n)$, and $\boldsymbol{X}_m$ does not have the column of ones. Following [234], the conjugate prior for the location parameters is $\boldsymbol{\beta}_m | \sigma^2 \sim N(\boldsymbol{\beta}_{m0}, \sigma^2 \boldsymbol{B}_{m0})$, and the priors for σ^2 and α can be improper, as these parameters are common to all models $\mathcal{M}_m$. Particularly, $\pi(\sigma^2) \propto 1/\sigma^2$ (Jeffreys' prior for the linear Gaussian model, see [200]) and $\pi(\alpha) \propto 1$.

The selection of the hyperparameters of $\boldsymbol{\beta}_m$ is more critical, as these parameters are not common to all models. A very common prior for the location parameters in the BMA literature is the Zellner's prior [430], where $\boldsymbol{\beta}_{m0} = \boldsymbol{0}_m$ and $\boldsymbol{B}_{m0} = (g_m \boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1}$. Observe that this covariance matrix is similar to the covariance matrix of the ordinary least squares estimator of the location parameters. This suggests that there is compatibility between the prior information and the sample information, and the only parameter to elicit is $g_m \geq 0$, which facilitates the elicitation process, as eliciting covariance matrices is a very hard endeavor.

Following same steps as in Section 3.3, the posterior conditional distribution of $\boldsymbol{\beta}_m$ has covariance matrix $\sigma^2 \boldsymbol{B}_{mn}$, where $\boldsymbol{B}_{mn} = ((1+g_m)\boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1}$ (Exercise 1), which means that $g_m = 0$ implies a non-informative prior, whereas $g_m = 1$ implies that prior and data information have same weights. We follow [129], who recommend

$$g_m = \begin{Bmatrix} 1/K^2, & N \leq K^2 \\ 1/N, & N > K^2 \end{Bmatrix}.$$

Given the likelihood function,

$$p(\boldsymbol{\beta}_m, \sigma^2 | \boldsymbol{y}, \boldsymbol{X}_m) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - \alpha \boldsymbol{i}_N - \boldsymbol{X}_m \boldsymbol{\beta}_m)^\top (\boldsymbol{y} - \alpha \boldsymbol{i}_N - \boldsymbol{X}_m \boldsymbol{\beta}_m) \right\},$$

the marginal likelihood associated with model $\mathcal{M}_m$ is proportional to (Exercise 1)

$$p(\boldsymbol{y} | \mathcal{M}_m) \propto \left(\frac{g_m}{1+g_m} \right)^{k_m/2} \left[(\boldsymbol{y} - \bar{y} \boldsymbol{i}_N)^\top (\boldsymbol{y} - \bar{y} \boldsymbol{i}_N) - \frac{1}{1+g_m} (\boldsymbol{y}^\top \boldsymbol{P}_{X_m} \boldsymbol{y}) \right]^{-(N-1)/2},$$

where all parameters are indexed to model $\mathcal{M}_m$, $\boldsymbol{P}_{X_m} = \boldsymbol{X}_m(\boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1}\boldsymbol{X}_m$ is the projection matrix on the space generated by the columns of $\boldsymbol{X}_m$, and $\bar{y}$ is the sample mean of $\boldsymbol{y}$.

We implement in our GUI four approaches to perform BMA in the Gaussian linear model: the BIC approximation using the Occam's window approach, the MC3 algorithm using the analytical expression for calculating the marginal likelihood, an instrumental variable approach based on conditional likelihoods, and dynamic variable selection.

Example: Simulation exercise

Let's perform a simulation exercise to assess the performance of the BIC approximation using the Occam's window, and the Markov chain Monte Carlo model composition approaches. Let's set a model where the computational burden is low and we know the data generating process (population statistical model). In particular, we set 10 regressors such that $x_k \sim N(1,1)$, $k = 1,\ldots,6$, and $x_k \sim B(0.5)$, $k = 7,\ldots,10$. We set $\boldsymbol{\beta} = [1\ 0\ 0\ 0\ 0.5\ 0, 0, 0, 0, -0.7]^\top$ such that just x_1, x_5 and x_{10} are relevant to drive $y_i = 1 + \boldsymbol{x}^\top\boldsymbol{\beta} + \mu_i$, $\mu_i \sim N(0, 0.5^2)$. Observe that we just have $2^{10} = 1024$ models in this setting, thus, we can calculate the posterior model probability for each model.

Our GUI uses the commands *bicreg* and *MC3.REG* from the package *BMA* to perform Bayesian model averaging in the linear regression model using the BIC approximation and MC3, respectively. These commands in turn are based on [320] and [324]. The following code shows how to perform the simulation and get the posterior mean and standard deviation using these commands with the default values of hyperparameters and tuning parameters.

R code. Simulation exercise: Bayesian model averaging, small setting

```
rm(list = ls()); set.seed(010101)
N <- 1000
K1 <- 6; K2 <- 4; K <- K1 + K2
X1 <- matrix(rnorm(N*K1,1 ,1), N, K1)
X2 <- matrix(rbinom(N*K2, 1, 0.5), N, K2)
X <- cbind(X1, X2); e <- rnorm(N, 0, 0.5)
B <- c(1,0,0,0,0.5,0,0,0,0,-0.7)
y <- 1 + X%*%B + e
BMAglm <- BMA::bicreg(X, y, strict = FALSE, OR = 50)
summary(BMAglm)
```

R code. Simulation exercise: Bayesian model averaging, small setting

```
BMAreg <- BMA::MC3.REG(y, X, num.its=500)
Models <- unique(BMAreg[["variables"]])
nModels <- dim(Models)[1]
nVistModels <- dim(BMAreg[["variables"]])[1]
PMP <- NULL
for(m in 1:nModels){
  idModm <- NULL
  for(j in 1:nVistModels){
    if(sum(Models[m,] == BMAreg[["variables"]][j,]) == K){
      idModm <- c(idModm, j)
    }else{
      idModm <- idModm
    }
  }
  PMPm <- sum(BMAreg[["post.prob"]][idModm])
  PMP <- c(PMP, PMPm)
}
PIP <- NULL
for(k in 1:K){
  PIPk <- sum(PMP[which(Models[,k] == 1)])
  PIP <- c(PIP, PIPk)
}
plot(PIP)
Means <- matrix(0, nModels, K)
Vars <- matrix(0, nModels, K)
for(m in 1:nModels){
  idXs <- which(Models[m,] == 1)
  if(length(idXs) == 0){
    Regm <- lm(y ~ 1)
  }else{
    Xm <- X[, idXs]
    Regm <- lm(y ~ Xm)
    SumRegm <- summary(Regm)
    Means[m, idXs] <- SumRegm[["coefficients"]][-1,1]
    Vars[m, idXs] <- SumRegm[["coefficients"]][-1,2]^2
  }
}
BMAmeans <- colSums(Means*PMP)
BMAsd <- (colSums(PMP*Vars)  + colSums(PMP*(Means-matrix(rep
    (BMAmeans, each = nModels), nModels, K))^2))^0.5
BMAmeans
[1]  1.001771e+00 -5.322016e-05  6.635422e-06  3.721457e-07
     4.976335e-01
[6] -1.271339e-04  1.000932e-08  2.107441e-05  6.578654e-06
    -7.035557e-01
BMAsd
[1] 1.527261e-02 1.353624e-03 5.936816e-04 1.163947e-04
    1.566698e-02 1.987360e-03
[7] 2.778896e-05 1.270579e-03 6.997305e-04 3.093389e-02
BMAmeans/BMAsd
```

We can see from the results that the BIC approximation with the Occam's window, and the MC3 algorithm perform a good job finding the relevant regressors, and their posterior BMA means are very close to the population values. We also see that the BMA results are very similar in the two approaches.

We can perform Bayesian model averaging in our GUI for linear Gaussian models using the BIC approximation and MC3 using Algorithms A28 and A29, respectively. We ask in Exercise 2 to perform BMA using the dataset *10ExportDiversificationHHI.csv* from [211].

Algorithm A28 Bayesian model averaging in linear Gaussian models using the Bayesian information criterion

1: Select *Bayesian Model Averaging* on the top panel
2: Select *Normal data* model using the left radio button
3: Select *BIC* using the right radio button under **Which type do you want to perform?**
4: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
5: Type the *OR* number of the Occam's window in the box under **OR: Number between 5 and 50**, this is not necessary as by default there is 50
6: Click the *Go!* button
7: Analyze results: After a few seconds or minutes, a table appears showing, for each regressor in the dataset, the PIP (posterior inclusion probability, **p!=0**), the BMA posterior mean (**EV**), the BMA standard deviation (**SD**), and the posterior mean for models with the highest PMP. At the bottom of the table, for the models with the largest PMP, the number of variables (**nVar**), the coefficient of determination (**r2**), the BIC, and the PMP (**post prob**) are displayed
8: Download posterior results using the *Download results using BIC*. There are two files, the first has the best models by row according to the PMP (last column) indicating with a 1 inclusion of the variable (0 indicates no inclusion), and the second file has the PIP, the BMA expected value and standard deviation for each variable in the dataset

We show in the following code how to program a MC3 algorithm from scratch to perform BMA using the setting from the previous simulation exercise. The first part of the code is the function to calculate the log marginal likelihood. This is a small simulation setting, thus we can calculate the marginal likelihood for all 1024 models, and then calculate the posterior model probability standardizing using the model with the largest log marginal likelihood. We see from the results that this model is the data generating process (population statistical model). We also find that the posterior inclusion probabilities

Algorithm A29 Bayesian model averaging in linear Gaussian models using Markov chain Monte Carlo model composition

1: Select *Bayesian Model Averaging* on the top panel
2: Select *Normal data* model using the left radio button
3: Select *MC3* using the right radio button under **Which type do you want to perform?**
4: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
5: Select MC3 iterations using the *Range slider* under the label **MC3 iterations:**
6: Click the *Go!* button
7: Analyze results: After a few seconds or minutes, a table appears showing, for each regressor in the dataset, the PIP (posterior inclusion probability, **p!=0**), the BMA posterior mean (**EV**), the BMA standard deviation (**SD**), and the posterior mean for models with the highest PMP. At the bottom of the table, for the models with the largest PMP, the number of variables (**nVar**), the coefficient of determination (**r2**), the BIC, and the PMP (**post prob**) are displayed
8: Download posterior results using the *Download results using BIC*. There are two files, the first has the best models by row according to the PMP (last column) indicating with a 1 inclusion of the variable (0 indicates no inclusion), and the second file has the PIP, the BMA expected value and standard deviation for each variable in the dataset

for x_1, x_5 and x_{10} are 1, whereas the PIP for the other variables are less than 0.05.

Although BMA allows incorporating model uncertainty in a regression framework, sometimes it is desirable to select just one model. Two compelling alternatives are the model with the largest posterior model probability, and the median probability model. The latter is the model which includes every predictor that has posterior inclusion probability higher than 0.5. The first model is the best alternative for prediction in the case of a 0–1 loss function [95], whereas the second is the best alternative when there is a quadratic loss function in prediction [22]. In this simulation, the two criteria indicate selection of the data generating process.

We also show how to estimate the posterior mean and standard deviation based on BMA in this code. We see that the posterior means are very close to the population parameters.

R code. Simulation exercise: Bayesian model averaging, small setting from scratch

```
rm(list = ls()); set.seed(010101)
N <- 1000; K1 <- 6; K2 <- 4; K <- K1 + K2
X1 <- matrix(rnorm(N*K1,1 ,1), N, K1)
X2 <- matrix(rbinom(N*K2, 1, 0.5), N, K2)
X <- cbind(X1, X2); e <- rnorm(N, 0, 0.5)
B <- c(1,0,0,0,0.5,0,0,0,0,-0.7); y <- 1 + X%*%B + e
LogMLfunt <- function(Model){
  indr <- Model == 1
  kr <- sum(indr)
  if(kr > 0){
    gr <- ifelse(N > kr^2, 1/N, kr^(-2))
    Xr <- matrix(Xnew[ , indr], ncol = kr)
    PX <- Xr%*%solve(t(Xr)%*%Xr)%*%t(Xr)
    s2pos <- c((t(y - mean(y))%*%(y - mean(y))) - t(y)%*%PX%*%y/(1 + gr))
    mllMod <- (kr/2)*log(gr/(1+gr))-(N-1)/2*log(s2pos)
  }else{
    gr <- ifelse(N > kr^2, 1/N, kr^(-2))
    s2pos <- c((t(y - mean(y))%*%(y - mean(y))))
    mllMod <- (kr/2)*log(gr/(1+gr))-(N-1)/2*log(s2pos)
  }
  return(mllMod)
}
combs <- expand.grid(c(0,1), c(0,1), c(0,1), c(0,1), c(0,1), c(0,1), c(0,1), c(0,1), c(0,1), c(0,1))
Xnew <- apply(X, 2, scale)
mll <- sapply(1:2^K, function(s){LogMLfunt(matrix(combs[s,], 1, K))})
MaxPMP <- which.max(mll); StMarLik <- exp(mll-max(mll))
PMP <- StMarLik/sum(StMarLik)
PMP[MaxPMP]; combs[MaxPMP,]
PIP <- NULL
for(k in 1:K){
  PIPk <- sum(PMP[which(combs[,k] == 1)]); PIP <- c(PIP, PIPk)
}
PIP
nModels <- dim(combs)[1]; Means <- matrix(0, nModels, K)
Vars <- matrix(0, nModels, K)
for(m in 1:nModels){
  idXs <- which(combs[m,] == 1)
  if(length(idXs) == 0){
    Regm <- lm(y ~ 1)
  }else{
    Xm <- X[, idXs]; Regm <- lm(y ~ Xm)
    SumRegm <- summary(Regm)
    Means[m, idXs] <- SumRegm[["coefficients"]][-1,1]
    Vars[m, idXs] <- SumRegm[["coefficients"]][-1,2]^2
  }
}
BMAmeans <- colSums(Means*PMP)
BMAsd <- (colSums(PMP*Vars)  + colSums(PMP*(Means-matrix(rep(BMAmeans, each = nModels), nModels, K))^2))^0.5
```

The following part of the code demonstrates how to perform the MC3 algorithm. While this algorithm is not strictly necessary for this small-dimensional problem, it serves as a useful pedagogical exercise. The starting point is to set $S = 100$ random models and order their log marginal likelihoods. The logic of the algorithm is to select the worst model among the S models and propose a candidate model to compete against it. We repeat this process for 1000 iterations (as shown in the code). Note that 1000 iterations is fewer than the number of potential models (1024). This is the essence of the MC3 algorithm: performing fewer iterations than the number of models in the space.[5]

In our algorithm, we analyze all model scenarios using different conditionals and reasonably assume the same prior model probability for all models, with the same cardinality for both the actual and candidate models. The posterior model probability (PMP) can be calculated in several ways. One method is to recover the unique models from the final set of S models, calculate the log marginal likelihood for these models, and then standardize by the best model among them. Another method involves calculating the PMP using the complete set of S final models, accounting for the fact that some models may appear multiple times in the set, which requires summing the PMPs of repeated models. A third method is to calculate the PMP based on the relative frequency with which a model appears in the final set of S models. These three methods can yield different PMPs, particularly when the number of MC3 iterations is small. In our example, using 1000 MC3 iterations, the data-generating process receives the highest PMP across all three methods.

A noteworthy aspect of this algorithm is that we can obtain a single model after significantly increasing the number of iterations (for example, try using 10,000 iterations). This can be advantageous if we require only one model. However, this approach neglects model uncertainty, which could be a desirable characteristic in some cases. As a challenge, we suggest programming an algorithm that yields S different models after completing the MC3 iterations (Exercise 3).

[5]Note that we have split the *while* loop in the code into two pages.

R code. Simulation exercise: Bayesian model averaging, small setting from scratch

```
M <- 100
Models <- matrix(rbinom(K*M, 1, p = 0.5), ncol=K, nrow = M)
mllnew <- sapply(1:M,function(s){LogMLfunt(matrix(Models[s
    ,], 1, K))})
oind <- order(mllnew, decreasing = TRUE)
mllnew <- mllnew[oind]; Models <- Models[oind, ]; iter <-
    1000
pb <- winProgressBar(title = "progress bar", min = 0, max =
    iter, width = 300); s <- 1
while(s <= iter){
  ActModel <- Models[M,]; idK <- which(ActModel == 1)
  Kact <- length(idK)
  if(Kact < K & Kact > 1){
    CardMol <- K; opt <- sample(1:3, 1)
    if(opt == 1){ # Same
      CandModel <- ActModel
    }else{
      if(opt == 2){ # Add
        All <- 1:K; NewX <- sample(All[-idK], 1)
        CandModel <- ActModel; CandModel[NewX] <- 1
      }else{ # Subtract
        LessX <- sample(idK, 1); CandModel <- ActModel
        CandModel[LessX] <- 0
      }
    }
  }else{
    CardMol <- K + 1
    if(Kact == K){
      opt <- sample(1:2, 1)
      if(opt == 1){ # Same
        CandModel <- ActModel
      }else{ # Subtract
        LessX <- sample(1:K, 1); CandModel <- ActModel
        CandModel[LessX] <- 0
      }
    }else{
      if(K == 1){
        opt <- sample(1:3, 1)
        if(opt == 1){ # Same
          CandModel <- ActModel
        }else{
          if(opt == 2){ # Add
            All <- 1:K; NewX <- sample(All[-idK], 1)
            CandModel <- ActModel; CandModel[NewX] <- 1
          }else{ # Subtract
            LessX <- sample(idK, 1); CandModel <- ActModel
            CandModel[LessX] <- 0
          }
        }
      }else{ # Add
        NewX <- sample(1:K, 1); CandModel <- ActModel
        CandModel[NewX] <- 1
      }
    }
  }
```

R code. Simulation exercise: Bayesian model averaging, small setting from scratch

```
  LogMLact <- LogMLfunt(matrix(ActModel, 1, K))
  LogMLcand <- LogMLfunt(matrix(CandModel, 1, K))
  alpha <- min(1, exp(LogMLcand-LogMLact))
  u <- runif(1)
  if(u <= alpha){
    mllnew[M] <- LogMLcand; Models[M, ] <- CandModel
    oind <- order(mllnew, decreasing = TRUE)
    mllnew <- mllnew[oind]; Models <- Models[oind, ]
  }else{
    mllnew <- mllnew; Models <- Models
  }
  s <- s + 1
  setWinProgressBar(pb, s, title=paste( round(s/iter*100, 0)
    ,"% done"))
}
close(pb)
ModelsUni <- unique(Models)
mllnewUni <- sapply(1:dim(ModelsUni)[1], function(s){
    LogMLfunt(matrix(ModelsUni[s,], 1, K))})
StMarLik <- exp(mllnewUni-mllnewUni[1])
PMP <- StMarLik/sum(StMarLik) # PMP based on unique selected
     models
nModels <- dim(ModelsUni)[1]
StMarLik <- exp(mllnew-mllnew[1])
PMPold <- StMarLik/sum(StMarLik) # PMP all selected models
PMPot <- NULL
PMPap <- NULL
FreqMod <- NULL
for(m in 1:nModels){
  idModm <- NULL
  for(j in 1:M){
    if(sum(ModelsUni[m,] == Models[j,]) == K){
      idModm <- c(idModm, j)
    }else{
      idModm <- idModm
    }
  }
  PMPm <- sum(PMPold[idModm]) # PMP unique models using sum
    of all selected models
  PMPot <- c(PMPot, PMPm)
  PMPapm <- length(idModm)/M # PMP using relative frequency
    in all selected models
  PMPap <- c(PMPap, PMPapm)
  FreqMod <- c(FreqMod, length(idModm))
}
```

R code. Simulation exercise: Bayesian model averaging, small setting from scratch

```
PIP <- NULL
for(k in 1:K){
  PIPk <- sum(PMP[which(ModelsUni[,k] == 1)])
  PIP <- c(PIP, PIPk)
}
Means <- matrix(0, nModels, K)
Vars <- matrix(0, nModels, K)
for(m in 1:nModels){
  idXs <- which(ModelsUni[m,] == 1)
  if(length(idXs) == 0){
    Regm <- lm(y ~ 1)
  }else{
    Xm <- X[, idXs]
    Regm <- lm(y ~ Xm)
    SumRegm <- summary(Regm)
    Means[m, idXs] <- SumRegm[["coefficients"]][-1,1]
    Vars[m, idXs] <- SumRegm[["coefficients"]][-1,2]^2
  }
}
BMAmeans <- colSums(Means*PMP)
BMAsd <- (colSums(PMP*Vars)  + colSums(PMP*(Means-matrix(rep
    (BMAmeans, each = nModels), nModels, K))^2))^0.5
```

An important issue to account for regressors (model) uncertainty in the identification of causal effects, rather than finding good predictors (association relationships), is endogeneity. Thus, we also implement the instrumental variable approach of Section 7.3 to tackle this issue in BMA. We assume that $\boldsymbol{\gamma} \sim N(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{\beta} \sim N(\mathbf{0}, \boldsymbol{I})$, and $\boldsymbol{\Sigma}^{-1} \sim W(3, \boldsymbol{I})$ [224].

[245] propose an algorithm based on conditional Bayes factors [110] that allows embedding MC3 within a Gibbs sampling algorithm. Given the candidate (M_c^{2nd}) and actual (M_{s-1}^{2nd}) models for the iteration s in the second stage, the conditional Bayes factor is

$$CBF^{2nd} = \frac{p(\boldsymbol{y}|M_c^{2nd}, \boldsymbol{\gamma}, \boldsymbol{\Sigma})}{p(\boldsymbol{y}|M_{s-1}^{2nd}, \boldsymbol{\gamma}, \boldsymbol{\Sigma})},$$

where

$$\begin{aligned} p(\boldsymbol{y}|M_c^{2nd}, \boldsymbol{\gamma}, \boldsymbol{\Sigma}) &= \int_{\mathcal{M}^{2nd}} p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\Sigma}) \pi(\boldsymbol{\beta}|M_c^{2nd}) d\boldsymbol{\beta} \\ &\propto |\boldsymbol{B}_n|^{-1/2} \exp\left\{\frac{1}{2}\boldsymbol{\beta}_n{}^\top \boldsymbol{B}_n^{-1} \boldsymbol{\beta}_n\right\}. \end{aligned}$$

In the first stage,

$$CBF^{1st} = \frac{p(\boldsymbol{y}|M_c^{1st}, \boldsymbol{\beta}, \boldsymbol{\Sigma})}{p(\boldsymbol{y}|M_{s-1}^{1st}, \boldsymbol{\beta}, \boldsymbol{\Sigma})},$$

where

$$\begin{aligned} p(\boldsymbol{y}|M_c^{1st}, \boldsymbol{\beta}, \boldsymbol{\Sigma}) &= \int_{\mathcal{M}^{1st}} p(\boldsymbol{y}|\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\Sigma}) \pi(\boldsymbol{\gamma}|M_c^{1st}) d\boldsymbol{\gamma} \\ &\propto |\boldsymbol{G}_n|^{-1/2} \exp\left\{\frac{1}{2}\boldsymbol{\gamma}_n{}^\top \boldsymbol{G}_n^{-1} \boldsymbol{\gamma}_n\right\}. \end{aligned}$$

These conditional Bayes factors assume $\pi(M^{1st}, M^{2sd}) \propto 1$. See [245] for more details of the instrumental variable BMA algorithm.[6]

We perform instrumental variable BMA in our GUI using the package *ivbma*. The Algorithm A30 shows how to perform this in our GUI.

[6] [232] and [244] propose other frameworks for BMA taking into account endogeneity.

Algorithm A30 Instrumental variable Bayesian model averaging in linear Gaussian models

1: Select *Bayesian Model Averaging* on the top panel
2: Select *Normal data* model using the left radio button
3: Select *Instrumental variable* using the right radio button under **Which type do you want to perform?**
4: Upload the dataset containing the dependent variable, endogenous regressors, and exogenous regressors including the constant (see Section 5.6 for details). User should select first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
5: Upload the dataset containing the instruments (see Section 5.6 for details). User should select first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File (Instruments)** legend
6: Write down the number of endogenous regressors in the box labeled **Number of Endogenous variables**
7: Select MCMC iterations and burn-in using the *Range slider* under the labels **MCMC iterations:** and **Burn-in Sample:**
8: Click the *Go!* button
9: Analyze results: After a few seconds or minutes, two tables appear showing, for each regressor in the dataset, the PIP (posterior inclusion probability, **p!=0**), and the BMA posterior mean (**EV**). The top table shows the results of the second stage (main equation), and the bottom table shows the results of the first stage (auxiliary equations)
10: Download posterior results using the *Download results using IV*. There are three files, the first file has the posterior inclusion probabilities of each variable, and the BMA posterior means of the coefficients in the first stage equations, the second file shows these results for the second stage (main equation), and the third file has the posteriors chains of all parameters by iteration.

Let's perform a simulation exercise to assess the performance of the instrumental variable BMA to uncover the data generating process in presence of endogeneity.

Example: Simulation exercise

Let's assume that $y_i = 2 + 0.5x_{i1} - x_{i2} + x_{i3} + \mu_i$, where $x_{i1} = 4z_{i1} - z_{i2} + 2z_{i3} + \epsilon_{i1}$ and $x_{i2} = -2z_{i1} + 3z_{i2} - z_{i3} + \epsilon_{i2}$, such that $[\epsilon_{i1}\ \epsilon_{i2}\ \mu_i]^\top \sim N(\mathbf{0}, \mathbf{\Sigma})$, where $\mathbf{\Sigma} = \begin{bmatrix} 1 & 0 & 0.8 \\ 0 & 1 & 0.5 \\ 0.8 & 0.5 & 1 \end{bmatrix}$, for $i = 1, 2, \ldots, 1000$. The endogeneity arises due to the correlation between μ_i and x_{i1} and x_{i2} through the stochastic errors. In addition, there are three instruments, $z_{il} \sim U(0, 1)$, for $l = 1, 2, 3$, and another 18 regressors believed to influence y_i, which are distributed according to a standard normal distribution.

The following code shows how to perform IV BMA using the *ivbma* package. We see from the results that the PIP of x_{i1}, x_{i2}, intercept and x_{i3} are equal to 1, whereas the remaining PIP are close to 0. In addition, the BMA means are also close to the population values. The PIP of the first stage equations, as well as their BMA posterior means, are very close to the populations values. The same happens with the covariance matrix.

We ask in Exercise 4 to perform BMA based on the BIC approximation and MC3 in this simulation setting. In addition, we ask in Exercise 5 to use the datasets *11ExportDiversificationHHI.csv* and *12ExportDiversificationHHIInstr.csv* to perform IV BMA assuming that the log of per capita gross domestic product is endogenous (*avglgdpcap*). See [211] for details.

R code. Simulation exercise: Instrumental variable Bayesian model averaging

```
rm(list = ls())
set.seed(010101)
simIV <- function(delta1,delta2,beta0,betas1,betas2,beta2,Sigma,n,z) {
  eps <- matrix(rnorm(3*n),ncol=3) %*% chol(Sigma)
  xs1 <- z%*%delta1 + eps[,1]
  xs2 <- z%*%delta2 + eps[,2]
  x2 <- rnorm(dim(z)[1])
  y <- beta0+betas1*xs1+betas2*xs2+beta2*x2 + eps[,3]
  X <- as.matrix(cbind(xs1,xs2,1,x2))
  colnames(X) <- c("x1en","x2en","cte","xex")
  y <- matrix(y,dim(z)[1],1)
  colnames(y) <- c("y")
  list(X=X,y=y)
}
n <- 1000 ; p <- 3
z <- matrix(runif(n*p),ncol=p)
rho31 <- 0.8; rho32 <- 0.5;
Sigma <- matrix(c(1,0,rho31,0,1,rho32,rho31,rho32,1),ncol=3)
delta1 <- c(4,-1,2); delta2 <- c(-2,3,-1); betas1 <- .5; betas2 <- -1; beta2 <- 1; beta0 <- 2
simiv <- simIV(delta1,delta2,beta0,betas1,betas2,beta2,Sigma,n,z)
nW <- 18
W <- matrix(rnorm(nW*dim(z)[1]),dim(z)[1],nW)
YXW<-cbind(simiv$y, simiv$X, W)
y <- YXW[,1]; X <- YXW[,2:3]; W <- YXW[,-c(1:3)]
S <- 10000; burnin <- 1000
regivBMA <- ivbma::ivbma(Y = y, X = X, Z = z, W = W, s = S+burnin, b = burnin, odens = S, print.every = round(S/10), run.diagnostics = FALSE)
PIPmain <- regivBMA[["L.bar"]] # PIP outcome
PIPmain
1.0000 1.0000 1.0000 1.0000 0.0125 0.0382 0.0145 0.0148 0.0136 0.0102 0.0070 0.0527 0.0014 0.0077 0.0211 0.0081 0.0047 0.0141 0.0028 0.0063 0.0072 0.0220
EVmain <- regivBMA[["rho.bar"]] # Posterior mean outcome
EVmain
5.105361e-01 -9.828459e-01  1.996885e+00  1.005497e+00 -1.700857e-04  9.946613e-04  1.086717e-04 -1.448951e-04  1.532812e-04  1.356334e-04 -6.027285e-05  9.119699e-04 -1.581408e-05  1.050517e-04 2.488002e-04 -6.229493e-05  4.292825e-05  3.371366e-05  5.345760e-06  5.933764e-05  5.066236e-05 1.516718e-04
PIPaux <- regivBMA[["M.bar"]] # PIP auxiliary
EVaux <- regivBMA[["lambda.bar"]] # Posterior mean auxiliary
plot(EVaux[,1])
plot(EVaux[,2])
EVsigma <- regivBMA[["Sigma.bar"]] # Posterior mean variance matrix
```

Bayesian model averaging has been also extended to state-space models. The point of departure is the univariate random walk state-space model (see Equations 8.1 and 8.2 in Chapter 8) conditional on model $\mathcal{M}_m$, $m = 1, 2 \ldots, M$.

$$y_t = \boldsymbol{x}_{mt}^\top \boldsymbol{\beta}_{mt} + \mu_{mt} \tag{10.1}$$

$$\boldsymbol{\beta}_{mt} = \boldsymbol{\beta}_{mt-1} + \boldsymbol{w}_{mt}, \tag{10.2}$$

where $\mu_{mt} \sim N(0, \sigma^2)$ and $\boldsymbol{w}_{mt} \sim N(\boldsymbol{0}, \boldsymbol{\Omega}_{mt})$.

Given $\boldsymbol{\beta}_{mt-1}|\boldsymbol{y}_{1:t-1} \sim N(\boldsymbol{b}_{mt-1}, \boldsymbol{B}_{mt-1})$, then, we know from Chapter 8 that $\boldsymbol{\beta}_{mt}|\boldsymbol{y}_{1:t-1} \sim N(\boldsymbol{b}_{mt-1}, \boldsymbol{R}_{mt})$, $\boldsymbol{R}_{mt} = \boldsymbol{B}_{mt-1} + \boldsymbol{\Omega}_{mt}$.

Specification of $\boldsymbol{\Omega}_t$ can be highly demanding. Thus, a common approach is to express $\boldsymbol{\Omega}_{mt} = \frac{1-\lambda}{\lambda}\boldsymbol{B}_{mt-1}$, where λ is called the *forgetting parameter* or *discount factor*, because it discounts the matrix $\boldsymbol{B}_{mt-1}$ that we would have with a deterministic state evolution into the matrix $\boldsymbol{R}_{mt}$ [309, Chap. 4]. This parameter is typically slightly below 1, and implies that $\boldsymbol{R}_{mt} = \lambda^{-1}\boldsymbol{B}_{mt-1}$. ($\lambda^{-1} > 1$).

[323] assume that the model changes infrequently, and its evolution is given by the transition matrix $\boldsymbol{T} = [t_{ml}]$, where $t_{ml} = P(\mathcal{M}_t = \mathcal{M}_m|\mathcal{M}_{t-1} = \mathcal{M}_l)$.

Then, the aim is to calculate the filtering distribution $p(\boldsymbol{\beta}_{mt}, \mathcal{M}_t|y_t) = \sum_{m=1}^{M} p(\boldsymbol{\beta}_{mt}|\mathcal{M}_t = \mathcal{M}_m, y_t)p(\mathcal{M}_t = \mathcal{M}_m|y_t)$. Thus, given the conditional distribution of the state at time $t-1$, $p(\boldsymbol{\beta}_{mt-1}, \mathcal{M}_{t-1}|y_{t-1}) = \sum_{m=1}^{M} p(\boldsymbol{\beta}_{mt-1}|\mathcal{M}_{t-1} = \mathcal{M}_m, y_{t-1})p(\mathcal{M}_{t-1} = \mathcal{M}_m|y_{t-1})$, where the conditional distribution of $\boldsymbol{\beta}_{mt-1}$ is approximated by a Gaussian distribution, $\boldsymbol{\beta}_{mt-1}|\mathcal{M}_{t-1} = \mathcal{M}_m, y_{t-1} \sim N(\boldsymbol{b}_{mt-1}, \boldsymbol{B}_{mt-1})$, then the first step to get the one-step-ahead predictive distribution is getting the prediction of the model indicator,

$$\begin{aligned} p(\mathcal{M}_t = \mathcal{M}_l|y_{t-1}) &= \sum_{m=1}^{M} p(\mathcal{M}_{t-1} = \mathcal{M}_m|y_{t-1}) \times t_{lm} \\ &\approx \frac{p(\mathcal{M}_{t-1} = \mathcal{M}_l|y_{t-1})^\delta + c}{\sum_{m=1}^{M} p(\mathcal{M}_{t-1} = \mathcal{M}_m|y_{t-1})^\delta + c}, \end{aligned}$$

where the second equality is used to avoid dealing with the M^2 elements of the transition matrix $\boldsymbol{T}$ such that the forgetting parameter δ is used, this parameter is slightly less than 1, and $c = 0.001/M$ is introduced to handle a model probability being brought to computational zero by outliers.

Then, we get the one-step-ahead predictive distribution of the state vector, $\boldsymbol{\beta}_{mt}|\mathcal{M}_t = \mathcal{M}_m, y_{t-1} \sim N(\boldsymbol{b}_{mt-1}, \lambda^{-1}\boldsymbol{B}_{mt-1})$

Now, we consider the filtering stage, where the model filtering equation is

$$p(\mathcal{M}_t = \mathcal{M}_l|y_t) = \frac{p(\mathcal{M}_t = \mathcal{M}_l|y_{t-1})p_l(y_t|y_{t-1})}{\sum_{m=1}^{M} p(\mathcal{M}_t = \mathcal{M}_m|y_{t-1})p_m(y_t|y_{t-1})},$$

where $p_m(y_t|y_{t-1})$ is the one-step-ahead predictive distribution of $y_t|y_{t-1}$,

which is $N(f_t, Q_t)$, where $f_t = \boldsymbol{x}_t^\top \boldsymbol{b}_{t-1}$ and $Q_t = \boldsymbol{x}_{mt}^\top \lambda^{-1} \boldsymbol{B}_{mt-1} \boldsymbol{x}_{mt} + \sigma^2$ (see Chapter 8).

The states filtering equation is $\boldsymbol{\beta}_{mt} | \mathcal{M}_t = \mathcal{M}_m, y_t \sim N(\boldsymbol{b}_{mt}, \boldsymbol{B}_{mt})$ where $\boldsymbol{b}_{mt}$ and $\boldsymbol{B}_{mt}$ are given in the Kalman filtering recursion of Chapter 8.

[323] initiate their algorithm assuming equal prior model probabilities, and σ^2 is estimated using a recursive method of moments estimator.[7]

We implement dynamic Bayesian model averaging in our GUI using the function *dma* from the package *dma*. Algorithm A31 shows how to perform inference using our GUI.

Algorithm A31 Dynamic Bayesian model averaging

1: Select *Bayesian Model Averaging* on the top panel
2: Select *Normal data* model using the left radio button
3: Select *Dynamic Bayesian model averaging* using the right radio button under **Which type do you want to perform?**
4: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
5: Upload the matrix of models selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
6: Type the *forgetting parameters* in the boxes under **Lambda: Number slightly below 1** and **Delta: Number slightly below 1**. This is not necessary as by default there is 0.99 in both cases
7: Click the *Go!* button
8: Analyze results: After a few seconds or minutes, a table appears showing, for each regressor in the dataset, the dynamic Bayesian average filtering recursions for each state (**Mean** and **Standard deviation**), the posterior model probability (**PMP**), and the Bayesian model averaging prediction (**Prediction**)
9: Download posterior results using the *Download results DBMA*. There are two files, the first has the dynamic Bayesian average filtering recursions for each state, and the second file has the PMP of each model, and the dynamic Bayesian model averaging prediction

Example: Dynamic Bayesian model averaging

We perform a simulation exercise where there are 8 (2^3) competing models originating from 3 regressors: $x_{tk} \sim N(0.5, 0.8^2)$ for $k = 2, 3, 4$, with $\beta_1 = 0.5$. The sequence β_{2t} ranges from 1 to 2 in steps of $1/T$, and β_{3t} is given by:

$$\beta_{3t} = \begin{cases} -1, & 1 < t \leq 0.75T \\ 0, & 0.75T < t \leq T \end{cases}$$

and $\beta_4 = 1.2$.

[7] [329] extends this approach to Markov chain Monte Carlo model composition

Then, we have the model:

$$y_t = \beta_1 + \beta_{2t}x_{2t} + \beta_{3t}x_{3t} + \beta_4 x_{4t} + \mu_t,$$

where $\mu_t \sim N(0,1)$ for $t = 1, 2, \ldots, 500$. This setting implies that during the first 75% of the period, the model with all 3 regressors is the data-generating process, while after this, the model with regressors 2 and 4 is the data-generating process.

The following code shows the simulation exercise and the results of the dynamic Bayesian model averaging, setting $\lambda = \delta = 0.99$.

R code. Simulation exercise: Dynamic Bayesian model averaging

```
rm(list = ls()); set.seed(010101)
T <- 500; K <- 3
X <- matrix(rnorm(T*K, mean = 0.5, sd = 0.8), T, K)
combs <- expand.grid(c(0,1), c(0,1), c(0,1))
B1 <- 0.5; B2t <- seq(1, 2, length.out=T )
a <- 0.75; B3t <- c(rep(-1,round(a*T)), rep(0,round((1-a)*T)))
B4 <- 1.2; sigma <- 1; mu <- rnorm(T, 0, sigma)
y <- B1 + X[,1]*B2t + X[,2]*B3t + X[,3]*B4 + mu
T0 <- 50
dma.test <- dma::dma(X, y, combs, lambda=.99, gamma=.99, initialperiod = T0)
plot(dma.test[["pmp"]][-c(1:T0),8], type = "l", col = "green", main = "Posterior model probability: Model all regressors vs model regressors 2 and 4", xlab = "Time", ylab = "PMP")
lines(dma.test[["pmp"]][-c(1:T0),6], col = "red")
legend(x = 0, y = 1, legend = c("Model: All regressors", "Model: Regressors 2 and 4"), col = c("green", "red"), lty=1:1, cex=0.8)
require(latex2exp)
plot(dma.test[["thetahat.ma"]][-c(1:T0),1], type = "l", col = "green", main = "Bayesian model average filtering recursion", xlab = "Time", ylab = TeX("$\\beta_{1}$"))
abline(h = B1, col = "red")
legend(x = 0, y = 0.4, legend = c("State filtering", "State population"), col = c("green", "red"), lty=1:1, cex=0.8)
plot(dma.test[["thetahat.ma"]][-c(1:T0),2], type = "l", col = "green", main = "Bayesian model average filtering recursion", xlab = "Time", ylab = TeX("$\\beta_{2t}$"), ylim = c(0.5,2))
lines(B2t[-c(1:T0)], col = "red")
legend(x = 0, y = 0.8, legend = c("State filtering", "State population"), col = c("green", "red"), lty=1:1, cex=0.8)
plot(dma.test[["thetahat.ma"]][-c(1:T0),3], type = "l", col = "green", main = "Bayesian model average filtering recursion", xlab = "Time", ylab = TeX("$\\beta_{3t}$"))
lines(B3t[-c(1:T0)], col = "red")
legend(x = 0, y = -0.4, legend = c("State filtering", "State population"), col = c("green", "red"), lty=1:1, cex=0.8)
plot(dma.test[["thetahat.ma"]][-c(1:T0),4], type = "l", col = "green", main = "Bayesian model average filtering recursion", xlab = "Time", ylab = TeX("$\\beta_{4t}$"))
abline(h = B4, col = "red")
legend(x = 0, y = 1.3, legend = c("State filtering", "State population"), col = c("green", "red"), lty=1:1, cex=0.8)
```

Figure 10.1 shows the posterior model probabilities for the model with all the regressors (green line) and the model with regressors 2 and 4 (red line). On one hand, we see that the model with all regressors, which is the data-generating process in the first period ($t \leq 0.75T$), has a PMP close to 1, and then its PMP decreases. On the other hand, the model with regressors 2 and 4 has a PMP close to 0 in the first part of the period, and then its PMP increases to values higher than 60% on average, when this model becomes the data-generating process. These results suggest that, in this particular simulation exercise, the dynamic Bayesian model averaging method works relatively well in calculating the PMPs.

FIGURE 10.1
Posterior model probability: Dynamic Bayesian model averaging.

Figures 10.2, 10.3, 10.4, and 10.5 show a comparison between the Bayesian model averaging filtering recursions of the states (green lines) and their population values (red lines). We observe that the filtering recursions follow the general pattern of the population values. However, the values are not perfectly aligned. This discrepancy arises because the posterior model probabilities (PMPs) of the models that match the data-generating process are not equal to 1, which in turn affects the performance of the filtering recursions.

Dynamic Bayesian model averaging was extended to logit models by [278]. We ask in Exercise 12 to perform a simulation of this model, and perform BMA using the function *logistic.dma* from the *dma* package.

FIGURE 10.2
State β_1: Population versus dynamic Bayesian model averaging of the filtering recursion.

FIGURE 10.3
State β_{2t}: Population versus dynamic Bayesian model averaging of the filtering recursion.

10.3 Generalized linear models

Generalized linear models (GLMs) were introduced by [293], extending the concept of linear regression to a more general setting. These models are char-

FIGURE 10.4
State β_{3t}: Population versus dynamic Bayesian model averaging of the filtering recursion.

FIGURE 10.5
State β_4: Population versus dynamic Bayesian model averaging of the filtering recursion.

acterized by: i) a dependent variable y_i whose probability distribution func-

tion belongs to the exponential family (see Section 3.1), ii) a linear predictor $\eta = \boldsymbol{x}^\top \boldsymbol{\beta}$, and iii) a link function such that $\mathbb{E}[y|\boldsymbol{x}] = g^{-1}(\boldsymbol{x}^\top \boldsymbol{\beta})$, which implies that $g(\mathbb{E}[y|\boldsymbol{x}]) = \boldsymbol{x}^\top \boldsymbol{\beta}$. GLMs can be extended to the overdispersed exponential family [279].

As we know from Section 3.1, the Poisson distribution belongs to the exponential family, such that $p(y|\lambda) = \frac{\exp(-\lambda)\exp(y\log(\lambda))}{y!}$, or in the canonical form $p(y|\eta) = \frac{\exp(\eta y - \exp(\eta))}{y!}$, where $\eta = \log(\lambda)$, which means that $\boldsymbol{x}^\top \boldsymbol{\beta} = \log(\lambda)$. Consequently, $\mathbb{E}[y|\boldsymbol{x}] = \nabla(\exp(\eta)) = \exp(\eta) = \lambda = \exp(\boldsymbol{x}^\top \boldsymbol{\beta})$. Therefore, the link function in the Poisson case is the *log* function. In Exercise 6, we ask you to show that the link function in the Bernoulli case is the *logit* function. Other examples include the identity function in the case of the Gaussian distribution and the negative inverse in the case of the gamma distribution.

We can use the GLM framework to perform Bayesian model averaging (BMA) using the BIC approximation, following [320]. Specifically, the BIC is given by $BIC = k_m \log(N) - 2\log(p(\hat{\boldsymbol{\theta}}_m|\boldsymbol{y}))$, where $\hat{\boldsymbol{\theta}}_m$ is the maximum likelihood estimator. Thus, we simply need to calculate the likelihood function at the maximum likelihood estimator.

Example: Simulation exercises

Let's perform some simulation exercises to assess the performance of the BIC approximation using the Occam's window in GLMs. There are 27 regressors, where x_{i1} and x_{i2} are just the relevant regressors in all exercises, $i = 1, 2, \ldots, 1000$.

- Logit: $x_k \sim N(0,1)$, $k = 1, \ldots, 27$, and $p(y_i = 1|\boldsymbol{x}_i) = \exp(0.5 + 0.8x_{i1} - 1.2x_{i2})/(1 + \exp(0.5 + 0.8x_{i1} - 1.2x_{i2}))$.
- Gamma: $x_k \sim N(0, 0.5^2)$, $k = 1, \ldots, 27$, and $y_i \sim G(\alpha, \delta)$ where $\alpha = -(0.5 + 0.2x_{i1}0.1x_{i2})^{-1}$ and $\delta = 1$.
- Poisson: $x_k \sim N(0,1)$, $k = 1, \ldots, 27$, and $\mathbb{E}[y_i|\boldsymbol{x}_i] = \lambda_i = \exp(0.5 + 1.1x_{i1} + 0.7x_{i2})$.

Our GUI uses the command *bic.glm* from the *BMA* package to perform BMA using the BIC approximation with the Occam's window in GLMs. The Algorithm A32 shows how to do this in our GUI, and the following code shows how to perform BMA in logit models using the simulation setting.

The results show that the PIPs of x_{i1} and x_{i2} are equal 1 in all three settings, the data generating process gets the highest PMP, and the BMA posterior means are close to the population values in each simulation setting. The other variables get PIPs close to 0, except a few exceptions, and the BMA posterior means are also close to 0. This suggests that the BIC approximation does a good job finding the data generating process in generalized linear models.

We can take advantage of the *glm* function in **R** to perform BMA by programming an MC3 algorithm. The following code illustrates how to do this

Algorithm A32 Bayesian model averaging in generalized linear models using the Bayesian information criterion

1: Select *Bayesian Model Averaging* on the top panel
2: Select the generalized linear model using the left radio button. Options: *Binomial data (Logit)*, *Real positive data (Gamma)* and *Count data (Poisson)*
3: Upload the dataset selecting first if there is header in the file, and the kind of separator in the *csv* file of the dataset (comma, semicolon, or tab). Then, use the *Browse* button under the **Choose File** legend
4: Type the *OR* number of the Occam's window in the box under **OR: Number between 5 and 50**, this is not necessary as by default there is 50
5: Type the *OL* number of the Occam's window in the box under **OL: Number between 0.0001 and 1**, this is not necessary as by default there is 0.0025
6: Click the *Go!* button
7: Analyze results: After a few seconds or minutes, a table appears showing, for each regressor in the dataset, the PIP (posterior inclusion probability, **p!=0**), the BMA posterior mean (**EV**), and the BMA standard deviation (**SD**). User can also see the PMP for the most relevant models
8: Download posterior results using the *Download results using BIC*. There are two files, the first has the best models by row according to the PMP (last column) indicating with a 1 inclusion of the variable (0 indicates no inclusion), and the second file has the PIP, the BMA expected value and standard deviation for each variable in the dataset

in the Poisson simulation. First, we simulate the data; second, we define a function to compute the log marginal likelihood approximation using the results from the *glm* function. Then, we initialize the models to begin the MC3 algorithm. After that, we implement the MC3 algorithm, which involves small modifications of the code used for MC3 in Gaussian linear models.[8] We can calculate the posterior model probabilities (PMPs), posterior inclusion probabilities (PIPs), BMA means, and standard deviations as we did previously.

The simulation setting involves 2^{27} models, which corresponds to approximately 135 million models in the model space. We run our MC3 algorithm using the BIC approximation with 50,000 iterations. This takes considerably more time than the BIC approximation from the *BMA* package, but it seems to perform well in identifying the data-generating process, as the PMP of this model equals 1. The posterior inclusion probabilities (PIPs) for x_{i1} and x_{i2} are also 1, and the posterior means are 1.1 and 0.7, respectively, which are equal to the population values. The t-ratios are far greater than 2. However, running 50,000 iterations results in mass concentration in one model, in this case, the

[8]Note that we have split the *while* loop in the code into two pages.

data-generating process. If we run 25,000 MC3 iterations, the highest PMP is 0.8, but it is not associated with the data-generating process. Nonetheless, the PIP is equal to 1 for x_{i1} and x_{i2}, and other regressors also have high PIPs. The BMA means for x_{i1} and x_{i2} are equal to the population values, and the BMA means for the other regressors are equal to 0. The t-ratios of the regressors in the population statistical model are much greater than 2, whereas the t-ratios of the other regressors are equal to 0. This exercise demonstrates that 25,000 iterations were not sufficient to uncover the data-generating process. However, it also emphasizes an important point: we need to analyze all the relevant results from the BMA analysis, not just the PMPs and/or PIPs.

In Exercise 10, we ask you to use this approach to perform a BMA algorithm in the logit regression, using the simulation setting for logit models from this section.

R code. Simulation exercise: BMA for generalized linear models

```
### Logit ###
rm(list = ls()); set.seed(010101)
n<-1000; B<-c(0.5,0.8,-1.2)
X<-matrix(cbind(rep(1,n),rnorm(n,0,1),rnorm(n,0,1)),n,length(B))
p <- exp(X%*%B)/(1+exp(X%*%B)); y <- rbinom(n, 1, p)
nXgar<-25; Xgar<-matrix(rnorm(nXgar*n),n,nXgar)
df<-as.data.frame(cbind(y,X[,-1],Xgar))
colnames(df) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "x14", "x15", "x16", "x17", "x18", "x19", "x20", "x21", "x22", "x23", "x24", "x25", "x26", "x27")
BMAglmLogit <- BMA::bic.glm(y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23+x24+x25+x26+x27, data = df, glm.family = binomial(link="logit"), strict = FALSE, OR = 50)
summary(BMAglmLogit)
### Gamma ###
rm(list = ls()); set.seed(010101)
n<-1000; B<- c(0.5, 0.2, 0.1)
X<-matrix(cbind(rep(1,n),rnorm(n,0,0.5),rnorm(n,0,0.5)),n,length(B))
y1 <- (X%*%B)^(-1)
y <- rgamma(n,y1,scale=1)
nXgar<-25; Xgar<-matrix(rnorm(nXgar*n),n,nXgar)
df<-as.data.frame(cbind(y,X[,-1],Xgar))
colnames(df) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "x14", "x15", "x16", "x17", "x18", "x19", "x20", "x21", "x22", "x23", "x24", "x25", "x26", "x27")
BMAglmGamma <- BMA::bic.glm(y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23+x24+x25+x26+x27, data = df, glm.family = Gamma(link="inverse"), strict = FALSE, OR = 50)
summary(BMAglmGamma)
### Poisson ###
rm(list = ls()); set.seed(010101)
n<-1000; B<-c(2,1.1,0.7)
X<-matrix(cbind(rep(1,n),rnorm(n,0,1),rnorm(n,0,1)),n,length(B))
y1<-exp(X%*%B); y<-rpois(n,y1)
nXgar<-25; Xgar<-matrix(rnorm(nXgar*n),n,nXgar)
df<-as.data.frame(cbind(y,X[,-1],Xgar))
colnames(df) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "x14", "x15", "x16", "x17", "x18", "x19", "x20", "x21", "x22", "x23", "x24", "x25", "x26", "x27")
BMAglmPoisson <- BMA::bic.glm(y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23+x24+x25+x26+x27, data = df, glm.family = poisson(link="log"), strict = FALSE, OR = 50)
summary(BMAglmPoisson)

```

R code. Simulation exercise: BMA for generalized linear models using MC3 from scratch

```
rm(list = ls()); set.seed(010101)
n<-1000; B<-c(2,1.1,0.7)
X<-matrix(cbind(rep(1,n),rnorm(n,0,1),rnorm(n,0,1)),n,length(B))
y1<-exp(X%*%B); y<-rpois(n,y1)
nXgar<-25; Xgar<-matrix(rnorm(nXgar*n),n,nXgar)
df<-as.data.frame(cbind(y,X[,-1],Xgar))
colnames(df) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "x14", "x15", "x16", "x17", "x18", "x19", "x20", "x21", "x22", "x23", "x24", "x25", "x26", "x27")
Xnew <- apply(df[,-1], 2, scale)
BICfunt <- function(Model){
  indr <- Model == 1; kr <- sum(indr)
  if(kr > 0){
    Xr <- as.matrix(Xnew[ , indr])
    model <- glm(y ~ Xr, family = poisson(link = "log"))
    model_bic <- BIC(model)
    mllMod <- -model_bic/2
  }else{
    model <- glm(y ~ 1, family = poisson(link = "log"))
    model_bic <- BIC(model); mllMod <- -model_bic/2
  }
  return(mllMod)
}
M <- 500; K <- dim(df)[2] - 1
Models <- matrix(rbinom(K*M, 1, p = 0.5), ncol = K, nrow = M)
mllnew <- sapply(1:M, function(s){BICfunt(matrix(Models[s,], 1, K))})
oind <- order(mllnew, decreasing = TRUE)
mllnew <- mllnew[oind]; Models <- Models[oind, ]
# Hyperparameters MC3
iter <- 25000
pb <- winProgressBar(title = "progress bar", min = 0, max = iter, width = 300)
s <- 1
while(s <= iter){
  ActModel <- Models[M,]
  idK <- which(ActModel == 1)
  Kact <- length(idK)
  if(Kact < K & Kact > 1){
    CardMol <- K
    opt <- sample(1:3, 1)
    if(opt == 1){ # Same
      CandModel <- ActModel
    }else{
          if(opt == 2){ # Add
      All <- 1:K
      NewX <- sample(All[-idK], 1)
      CandModel <- ActModel
      CandModel[NewX] <- 1
    }else{ # Subtract
      LessX <- sample(idK, 1)
      CandModel <- ActModel
      CandModel[LessX] <- 0
    }
  }
```

R code. Simulation exercise: BMA for generalized linear models using MC3 from scratch

```
  }else{
    CardMol <- K + 1
    if(Kact == K){
      opt <- sample(1:2, 1)
      if(opt == 1){ # Same
        CandModel <- ActModel
      }else{ # Subtract
        LessX <- sample(1:K, 1)
        CandModel <- ActModel
        CandModel[LessX] <- 0
      }
    }else{
      if(K == 1){
        opt <- sample(1:3, 1)
        if(opt == 1){ # Same
          CandModel <- ActModel
        }else{
          if(opt == 2){ # Add
            All <- 1:K
            NewX <- sample(All[-idK], 1)
            CandModel <- ActModel
            CandModel[NewX] <- 1
          }else{ # Subtract
            LessX <- sample(idK, 1)
            CandModel <- ActModel
            CandModel[LessX] <- 0
          }
        }
      }else{ # Add
        NewX <- sample(1:K, 1)
        CandModel <- ActModel
        CandModel[NewX] <- 1
      }
    }
  }
  LogMLact <- BICfunt(matrix(ActModel, 1, K))
  LogMLcand <- BICfunt(matrix(CandModel, 1, K))
  alpha <- min(1, exp(LogMLcand-LogMLact))
  u <- runif(1)
  if(u <= alpha){
    mllnew[M] <- LogMLcand
    Models[M, ] <- CandModel
    oind <- order(mllnew, decreasing = TRUE)
    mllnew <- mllnew[oind]
    Models <- Models[oind, ]
  }else{
    mllnew <- mllnew
    Models <- Models
  }
  s <- s + 1
  setWinProgressBar(pb, s, title=paste( round(s/iter*100, 0)
    ,"% done"))
}
close(pb)
```

R code. Simulation exercise: BMA for generalized linear models using MC3 from scratch

```
ModelsUni <- unique(Models)
mllnewUni <- sapply(1:dim(ModelsUni)[1], function(s){BICfunt
    (matrix(ModelsUni[s,], 1, K))})
StMarLik <- exp(mllnewUni-mllnewUni[1])
PMP <- StMarLik/sum(StMarLik) # PMP based on unique selected
     models
plot(PMP)
ModelsUni[1,]
PIP <- NULL
for(k in 1:K){
  PIPk <- sum(PMP[which(ModelsUni[,k] == 1)])
  PIP <- c(PIP, PIPk)
}
plot(PIP)
Xnew <- df[,-1]
nModels <- dim(ModelsUni)[1]
Means <- matrix(0, nModels, K)
Vars <- matrix(0, nModels, K)
for(m in 1:nModels){
  idXs <- which(ModelsUni[m,] == 1)
  if(length(idXs) == 0){
    Regm <- glm(y ~ 1, family = poisson(link = "log"))
  }else{
    Xm <- as.matrix(Xnew[, idXs])
    Regm <- glm(y ~ Xm, family = poisson(link = "log"))
    SumRegm <- summary(Regm)
    Means[m, idXs] <- SumRegm[["coefficients"]][-1,1]
    Vars[m, idXs] <- SumRegm[["coefficients"]][-1,2]^2
  }
}
BMAmeans <- colSums(Means*PMP)
BMAsd <- (colSums(PMP*Vars)  + colSums(PMP*(Means-matrix(rep
    (BMAmeans, each = nModels), nModels, K))^2))^0.5
plot(BMAmeans)
plot(BMAsd)
plot(BMAmeans/BMAsd)
```

10.4 Calculating the marginal likelihood

The BIC is an asymptotic approximation of the marginal likelihood, and consequently, it is used to obtain the Bayes factors. However, this method has

limitations in applications with moderate and small sample sizes [147]. Therefore, other methods are available to calculate the Bayes factors when there is no analytical solution for the marginal likelihood.

Observe that calculating the Bayes factor with respect to a reference model ($\mathcal{M}_0$) helps to obtain the posterior model probabilities,

$$\begin{aligned}\pi(\mathcal{M}_j|\boldsymbol{y}) &= \frac{p(\boldsymbol{y}|\mathcal{M}_j)\pi(\mathcal{M}_j)}{\sum_{m=1}^{M} p(\boldsymbol{y}|\mathcal{M}_m)\pi(\mathcal{M}_m)} \\ &= \frac{p(\boldsymbol{y}|\mathcal{M}_j)\pi(\mathcal{M}_j)/p(\boldsymbol{y}|\mathcal{M}_0)}{\sum_{m=1}^{M} p(\boldsymbol{y}|\mathcal{M}_m)\pi(\mathcal{M}_m)/p(\boldsymbol{y}|\mathcal{M}_0)} \\ &= \frac{BF_{j0} \times \pi(\mathcal{M}_j)}{\sum_{m=1}^{M} BF_{l0} \times \pi(\mathcal{M}_l)}.\end{aligned}$$

Thus, $\pi(\mathcal{M}_j|\boldsymbol{y}) = \frac{BF_{j0}}{\sum_{m=1}^{M} BF_{l0}}$ assuming equal prior model probabilities.

In addition, it has been established in many settings that the Bayes factor is consistent. That is, the probability of identifying the true data generating process converges to 1 as the sample size increases to infinity. Alternatively, it asymptotically identifies the model that minimizes the Kullback-Leibler divergence with respect to the data generating process when this process is not part of the models under consideration [90, 410, 409].[9]

10.4.1 Savage-Dickey density ratio

The Savage-Dickey density ratio is a way to calculate the Bayes factors when we compare nested models with particular priors [111, 406]. In particular, given the parameter space $\boldsymbol{\theta} = (\boldsymbol{\omega}^\top, \boldsymbol{\psi}^\top)^\top \in \boldsymbol{\Theta} = \boldsymbol{\Omega} \times \boldsymbol{\Psi}$, where we wish to test the null hypothesis $H_0 : \boldsymbol{\omega} = \boldsymbol{\omega}_0$ (model $\mathcal{M}_1$) versus $H_1 : \boldsymbol{\omega} \neq \boldsymbol{\omega}_0$ (model $\mathcal{M}_2$), if $\pi(\boldsymbol{\psi}|\boldsymbol{\omega}_0, \mathcal{M}_2) = \pi(\boldsymbol{\psi}|\mathcal{M}_1)$,[10] then the Bayes factor comparing $\mathcal{M}_1$ versus $\mathcal{M}_2$ is

$$BF_{12} = \frac{\pi(\boldsymbol{\omega} = \boldsymbol{\omega}_0|\boldsymbol{y}, \mathcal{M}_2)}{\pi(\boldsymbol{\omega} = \boldsymbol{\omega}_0|\mathcal{M}_2)}, \tag{10.3}$$

where $\pi(\boldsymbol{\omega} = \boldsymbol{\omega}_0|\boldsymbol{y}, \mathcal{M}_2)$ and $\pi(\boldsymbol{\omega} = \boldsymbol{\omega}_0|\mathcal{M}_2)$ are the posterior and prior densities of $\boldsymbol{\omega}$ under $\mathcal{M}_2$ evaluated at $\boldsymbol{\omega}_0$ (see [406]).

Equation 10.3 is called the Savage-Dickey density ratio. A nice feature is that just requires estimation of model $\mathcal{M}_2$, and evaluation of the prior and posterior densities. This means no evaluation of the marginal likelihood [234,

[9] [213] highlight the important distinction between pairwise consistency and model selection consistency. The latter requires the consistency of a sequence of pairwise nested comparisons.

[10] Note that a sufficient condition for this assumption is to assume the same prior for the parameters that are the same in each model. [406] incorporate a correction factor when this assumption is not satisfied.

Chap. 4].

10.4.2 Chib's methods

Another popular method to calculate the marginal likelihood is given by [84] and [89]. The former is an algorithm to calculate the marginal likelihood from the posterior draws of the Gibbs sampling algorithm, and the latter calculates the marginal likelihood from the posterior draws of the Metropolis-Hastings algorithm.

The point of departure in [84] is the identity

$$\pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \mathcal{M}_m) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}^*, \mathcal{M}_m) \times \pi(\boldsymbol{\theta}^*|\mathcal{M}_m)}{p(\boldsymbol{y}|\mathcal{M}_m)},$$

where $\boldsymbol{\theta}^*$ is a particular value of $\boldsymbol{\theta}$ of high probability, for instance, the mode. This implies that

$$p(\boldsymbol{y}|\mathcal{M}_m) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}^*, \mathcal{M}_m) \times \pi(\boldsymbol{\theta}^*|\mathcal{M}_m)}{\pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \mathcal{M}_m)}.$$

We can easily calculate the numerator of this expression. However, the critical point in this expression is to calculate the denominator as we know $\pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \mathcal{M}_m)$ up to a normalizing constant. We can calculate this from the posterior draws. Assume that $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top \ \boldsymbol{\theta}_2^\top]^\top$, then $\pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \mathcal{M}_m) = \pi(\boldsymbol{\theta}_1^*|\boldsymbol{\theta}_2^*, \boldsymbol{y}, \mathcal{M}_m) \times \pi(\boldsymbol{\theta}_2^*|\boldsymbol{y}, \mathcal{M}_m)$. We have the first term because in the Gibbs sampling algorithm the posterior conditional distributions are available. The second is

$$\begin{aligned}
\pi(\boldsymbol{\theta}_2^*|\boldsymbol{y}, \mathcal{M}_m) &= \int_{\boldsymbol{\Theta}_1} \pi(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2^*|\boldsymbol{y}, \mathcal{M}_m) d\boldsymbol{\theta}_1 \\
&= \int_{\boldsymbol{\Theta}_1} \pi(\boldsymbol{\theta}_2^*|\boldsymbol{\theta}_1, \boldsymbol{y}, \mathcal{M}_m) \pi(\boldsymbol{\theta}_1|\boldsymbol{y}, \mathcal{M}_m) d\boldsymbol{\theta}_1 \\
&\approx \frac{1}{S} \sum_{s=1}^{S} \pi(\boldsymbol{\theta}_2^*|\boldsymbol{\theta}_1^{(s)}, \boldsymbol{y}, \mathcal{M}_m),
\end{aligned}$$

where $\boldsymbol{\theta}_1^{(s)}$ are the posterior draws of $\boldsymbol{\theta}_1$ from the Gibbs sampling algorithm.

The generalization to more blocks can be seen in [84] and [173, Chap. 7]. In addition, the extension to the Metropolis-Hastings algorithm can be seen in [89], and [173, Chap. 7].

10.4.3 Gelfand-Dey method

We can use the Gelfand-Dey method [147] when we want to calculate the Bayes factor to compare non-nested models, models where the Savage-Dickey density

ratio is hard to calculate, or the Chib's methods are difficult to implement. The Gelfand-Dey method is very general, and can be used in virtually any model [234, Chap. 5].

Given a probability density function $q(\boldsymbol{\theta})$, whose support is in $\boldsymbol{\Theta}$, then

$$\mathbb{E}\left[\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m,\mathcal{M}_m)}\middle|\boldsymbol{y},\mathcal{M}_m\right]=\frac{1}{p(\boldsymbol{y}|\mathcal{M}_m)},$$

where the expected value is with respect to the posterior distribution given the model $\mathcal{M}_m$ (see Exercise 12).

The critical point is to select a good $q(\boldsymbol{\theta})$. [161] recommends to use $q(\boldsymbol{\theta})$ equal to a truncated multivariate normal density function with mean and variance equal to the posterior mean ($\hat{\boldsymbol{\theta}}$) and variance ($\hat{\boldsymbol{\Sigma}}$) of $\boldsymbol{\theta}$. The truncation region is $\hat{\boldsymbol{\Theta}}=\left\{\boldsymbol{\theta}:(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^\top\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})\leq\chi^2_{1-\alpha}(K)\right\}$, where $\chi^2_{1-\alpha}(K)$ is the $(1-\alpha)$ percentile of the Chi-squared distribution with K degrees of freedom, K is the dimension of $\boldsymbol{\theta}$. We can pick small values of α, for instance, $\alpha=0.01$.

Observe that

$$\mathbb{E}\left[\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m,\mathcal{M}_m)}\middle|\boldsymbol{y},\mathcal{M}_m\right]\approx\frac{1}{S}\sum_{s=1}^{S}\left[\frac{q(\boldsymbol{\theta}^{(s)})}{\pi(\boldsymbol{\theta}^{(s)}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m^{(s)},\mathcal{M}_m)}\right],$$

where $\boldsymbol{\theta}_m^{(s)}$ are draws from the posterior distribution.

Observe that we can calculate the marginal likelihoods of the models in Chapters 6, 7, 8 and 9 using the Chib's methods and the Gelfand-Dickey method.

Example: Simulation exercise

Let's check the performance of the Savage-Dickey density ratio, Chib's method and the Gelfand-Dey method to calculate the Bayes factor in a setting where we can obtain the analytical solution for the marginal likelihood. In particular, we will consider the Gaussian linear model with a conjugate prior (see Section 3.3).

Assume that the data generating process is given by

$$y_i=0.7+0.3x_{i1}+0.7x_{i2}-0.2x_{i3}+0.2x_{i4}+\mu_i,$$

where $x_{i1}\sim B(0.3)$, $x_{ik}\sim N(0,1)$, for $k=2,\ldots,4$, and $\mu_i\sim N(0,1)$, for $i=1,2,\ldots,500$. Let us set $H_0:\beta_5=0$ (model $\mathcal{M}_1$) versus $H_1:\beta_5\neq 0$ (model $\mathcal{M}_2$).

We assume that $\boldsymbol{\beta}_{m0}=\mathbf{0}_{m0}$, $\boldsymbol{B}_{m0}=0.5\boldsymbol{I}_m$, $\alpha_0=\delta_0=4$. The dimensions of $\mathbf{0}_{m0}$ and $\boldsymbol{I}_m$ are 4 for model $\mathcal{M}_1$ and 5 for model $\mathcal{M}_2$. In addition, we assume equal prior probabilities for both models.

We know from Section 3.3 that the marginal likelihood is

$$p(\mathbf{y}|\mathcal{M}_\mathbf{m})=\frac{\delta_{m0}^{\alpha_{m0}/2}}{\delta_{mn}^{\alpha_{mn}/2}}\frac{|\mathbf{B}_{mn}|^{1/2}}{|\mathbf{B}_{m0}|^{1/2}}\frac{\Gamma(\alpha_{mn}/2)}{\Gamma(\alpha_{m0}/2)},$$

where $\boldsymbol{B}_{mn} = (\boldsymbol{B}_{m0}^{-1} + \boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1}$, $\boldsymbol{\beta}_{mn} = \mathbf{B}_{mn}(\boldsymbol{B}_{m0}^{-1}\boldsymbol{\beta}_{m0} + \boldsymbol{X}_m^\top \boldsymbol{X}_m \hat{\boldsymbol{\beta}}_m)$, $\alpha_{mn} = \alpha_{m0} + N$, and $\delta_{mn} = \delta_{m0} + (\boldsymbol{y} - \boldsymbol{X}_m\hat{\boldsymbol{\beta}}_m)^\top(\boldsymbol{y} - \boldsymbol{X}_m\hat{\boldsymbol{\beta}}_m) + (\hat{\boldsymbol{\beta}}_m - \boldsymbol{\beta}_{m0})^\top((\boldsymbol{X}_m{}^\top \boldsymbol{X}_m)^{-1} + \boldsymbol{B}_{m0})^{-1}(\hat{\boldsymbol{\beta}}_m - \boldsymbol{\beta}_{m0})$, $m = 1, 2$ are the indices of the models.

The log marginal likelihoods for models $\mathcal{M}_1$ and $\mathcal{M}_2$ are -751.72 and -740.79, respectively. This implies a $2\times\log(BF_{21}) = 21.85$ which means positive evidence against model $\mathcal{M}_1$ (see Table 1.1).

We have different ways to calculate the Bayes factor using the Savage-Dickey density ratio in this example because we know that the marginal prior and marginal posterior distributions of β_5 have analytical solutions. In addition, we can use the posterior draws of σ^2 to evaluate the conditional prior and conditional posterior distributions at $\beta_5 = 0$. We show in the following code the latter approach, as it is more general than using analytical solutions, which are not always available.

We know that the conditional posterior distribution of β_5 is $N(\beta_{5n}, \sigma\boldsymbol{B}_{55n})$, where β_{5n} is the 5th element of $\boldsymbol{\beta}_n$, and $\boldsymbol{B}_{55n}$ is the element 5,5 of $\boldsymbol{B}_n$. Then,

$$\begin{aligned}\pi(\beta_5 = 0|\boldsymbol{y}, \mathcal{M}_2) &= \int_{\mathcal{R}^+} \pi(\beta_5 = 0|\boldsymbol{y}, \sigma^2)\pi(\sigma^2|\boldsymbol{y})\, d\sigma^2 \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \pi(\beta_5 = 0|\boldsymbol{y}, \sigma^{2(s)}),\end{aligned}$$

where $\sigma^{2(s)}$ are draws from the posterior distribution of σ^2.

We can follow the same logic to obtain an approximation to $\pi(\beta_5 = 0|\mathcal{M}_2)$ by sampling draws from the prior distribution of σ^2.

We obtain $2 \times \log(BF_{21}) = 21.85$ using the Savage-Dickey density ratio, which is the same value as the analytic solution using the marginal likelihoods.

We calculate the log marginal likelihood using the Chib's method taking into account that

$$\log(p(\boldsymbol{y}|\mathcal{M}_m)) = \log(p(\boldsymbol{y}|\boldsymbol{\theta}^*, \mathcal{M}_m)) + \log(\pi(\boldsymbol{\theta}^*|\mathcal{M}_m)) - \log(\pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \mathcal{M}_m)),$$

where $p(\boldsymbol{y}|\boldsymbol{\theta}^*, \mathcal{M}_m)$ is the value of a normal density with mean $\boldsymbol{X}_m\boldsymbol{\beta}_m^*$ and variance $\sigma_m^{2*}\boldsymbol{I}_N$ evaluated at $\boldsymbol{y}$. In addition, $\log(\pi(\boldsymbol{\theta}^*|\mathcal{M}_m)) = \log(\pi(\boldsymbol{\beta}_m^*|\sigma_m^{2*})) + \log(\pi(\sigma_m^{2*}))$, where the first term is the density of a normal with mean $\boldsymbol{\beta}_{m0}$ and variance matrix $\sigma^{2*}\boldsymbol{B}_{m0}$ evaluated at $\boldsymbol{\beta}_m^*$, and the second term is the density of an inverse-gamma with parameters $\alpha_{m0}/2$ and $\delta_{m0}/2$ evaluated at σ_m^{2*}. Finally, the third term in the right hand of the previous expression is $\log(\pi(\boldsymbol{\theta}^*|\boldsymbol{y}, \mathcal{M}_m)) = \log(\pi(\boldsymbol{\beta}_m^*|\sigma_m^{2*}, \boldsymbol{y})) + \log(\pi(\sigma_m^{2*}|\boldsymbol{y}))$, where the first term is the density of a normal with mean $\boldsymbol{\beta}_{mn}$ and variance matrix $\sigma_m^{2*}\boldsymbol{B}_{mn}$ evaluated at $\boldsymbol{\beta}_m^*$, and the second term is the density of an inverse-gamma with parameters $\alpha_{mn}/2$ and $\delta_{mn}/2$ evaluated at σ_m^{2*}. We use the modes of the posterior draws of $\boldsymbol{\beta}_m$ and σ_m^2 as reference values.

We get the same value, up to two decimals, for the log marginal likelihood

of the restricted and unrestricted models using the Chib's method and the analytical expression. Thus, $2 \times \log(BF_{21}) = 21.85$, that is, positive evidence against model $\mathcal{M}_1$ (see Table 1.1).

We calculate the log marginal likelihood using the Gelfand-Dey method taking into account that

$$\log\left[\frac{q(\boldsymbol{\theta}^{(s)})}{\pi(\boldsymbol{\theta}^{(s)}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m^{(s)},\mathcal{M}_m)}\right] = \log(q(\boldsymbol{\theta}^{(s)})) - \log(\pi(\boldsymbol{\theta}^{(s)}|\mathcal{M}_m)) - \log(p(\boldsymbol{y}|\boldsymbol{\theta}_m^{(s)},\mathcal{M}_m)),$$

where $q(\boldsymbol{\theta}^{(s)})$ is the truncated multivariate normal density of Subsection 10.4.3 evaluated at $\boldsymbol{\theta}^{(s)} = [\boldsymbol{\beta}^{(s)\top}\ \sigma^{2(s)}]^\top$, which is the s-th posterior draw of the Gibbs sampling algorithm, such that $\boldsymbol{\theta}^{(s)}$ satisfies the truncation restriction. $\log(\pi(\boldsymbol{\theta}^{(s)}|\mathcal{M}_m)) = \log(\pi(\boldsymbol{\beta}_m^{(s)}|\sigma_m^{2(s)})) + \log(\pi(\sigma_m^{2(s)}))$, where the first term is the density of a normal with mean $\boldsymbol{\beta}_{m0}$ and variance matrix $\sigma^{2(s)}\boldsymbol{B}_{m0}$ evaluated at $\boldsymbol{\beta}_m^{(s)}$, and the second term is the density of an inverse-gamma with parameters $\alpha_{m0}/2$ and $\delta_{m0}/2$ evaluated at $\sigma_m^{2(s)}$. The third term $p(\boldsymbol{y}|\boldsymbol{\theta}^{(s)},\mathcal{M}_m)$ is the value of a normal density with mean $\boldsymbol{X}_m\boldsymbol{\beta}_m^{(s)}$ and variance $\sigma_m^{2(s)}\boldsymbol{I}_N$ evaluated at $\boldsymbol{y}$.

The log marginal likelihoods of the restricted and unrestricted models using the Gelfand-Dey method are -751.79 and -740.89, respectively. This implies $2 \times \log(BF_{21}) = 21.81$, which is positive evidence in favor of the unrestricted model.

We see in this example that these methods give very good approximations to the true marginal likelihoods. However, the Savage-Dickey density ratio and Chib's method performed slightly better than the Gelfand-Dey method. In addition, the computational demand of the Gelfand-Dey method is by far the largest. This is because the Gelfand-Dey method requires many evaluations based on the posterior draws. However, we should keep in mind that the Gelfand-Dey method is more general.

The following code shows how to do all these calculations.

R code. Simulation exercise: Bayes factors

```
rm(list = ls()); set.seed(010101)
N <- 500; K <- 5; K2 <- 3
B <- c(0.7, 0.3, 0.7, -0.2, 0.2)
X1 <- rbinom(N, 1, 0.3)
X2 <- matrix(rnorm(K2*N), N, K2)
X <- cbind(1, X1, X2)
Y <- X%*%B + rnorm(N, 0, sd = 1)
# Hyperparameters
d0 <- 4; a0 <- 4
b0 <- rep(0, K); c0pt <- 0.5
an <- N + a0; B0 <- c0pt*diag(K)
Bn <- solve(solve(B0)+t(X)%*%X); bhat <- solve(t(X)%*%X)%*%t(X)%*%Y
bn <- Bn%*%(solve(B0)%*%b0+t(X)%*%X%*%bhat)
dn <- as.numeric(d0 + t(Y-X%*%bhat)%*%(Y-X%*%bhat)+t(bhat - b0)%*%solve(solve(t(X)%*%X)+B0)%*%(bhat - b0))
Hn <- as.matrix(Matrix::forceSymmetric(dn*Bn/an))
S <- 10000
LogMarLikLM <- function(X, c0){
  K <- dim(X)[2]
  N <- dim(X)[1]
  # Hyperparameters
  B0 <- c0*diag(K)
  b0 <- rep(0, K)
  # Posterior parameters
  bhat <- solve(t(X)%*%X)%*%t(X)%*%Y
  # Force this matrix to be symmetric
  Bn <- as.matrix(Matrix::forceSymmetric(solve(solve(B0) + t(X)%*%X)))
  bn <- Bn%*%(solve(B0)%*%b0 + t(X)%*%X%*%bhat)
  dn <- as.numeric(d0 + t(Y)%*%Y+t(b0)%*%solve(B0)%*%b0-t(bn)%*%solve(Bn)%*%bn)
  an <- a0 + N
  # Log marginal likelihood
  logpy <- (N/2)*log(1/pi)+(a0/2)*log(d0)-(an/2)*log(dn) + 0.5*log(det(Bn)/det(B0)) + lgamma(an/2)-lgamma(a0/2)
  return(-logpy)
}
LogMarM2 <- -LogMarLikLM(X = X, c0 = c0pt)
LogMarM1 <- -LogMarLikLM(X = X[,1:4], c0 = c0pt)
BF12 <- exp(LogMarM1-LogMarM2)
BF12; 1/BF12
2*log(1/BF12)
# Savage-Dickey density ratio
# Posterior evaluation
Brest <- 0
sig2P <- invgamma::rinvgamma(S, shape = an/2, rate = dn/2)
PostRestCom <- mean(sapply(sig2P, function(x){dnorm(Brest, mean = bn[5], sd = (x*Bn[5,5])^0.5, log = FALSE)}))
# Prior evaluation
sig2 <- invgamma::rinvgamma(S, shape = a0/2, rate = d0/2)
PriorRestCom <- mean(sapply(sig2, function(x){dnorm(Brest, mean = 0, sd = (x*c0pt)^0.5, log = FALSE)}))
# Bayes factor
BF12SD <- PostRestCom/PriorRestCom
2*log(1/BF12SD)
```

R code. Simulation exercise: Bayes factors

```r
# Chib's method
sig2Post <- MCMCpack::rinvgamma(S,an/2,dn/2)
BetasGibbs <- sapply(1:S, function(s){MASS::mvrnorm(n = 1,
    mu = bn, Sigma = sig2Post[s]*Bn)})
# Mode function for continuous data
mode_continuous <- function(x){
  density_est <- density(x)
  mode_value <- density_est$x[which.max(density_est$y)]
  return(mode_value)
}
# Unrestricted model
BetasMode <- apply(BetasGibbs, 1, mode_continuous)
Sigma2Mode <- mode_continuous(sig2Post)
VarModel <- Sigma2Mode*diag(N)
MeanModel <- X%*%BetasMode
LogLik <- mvtnorm::dmvnorm(c(Y), mean = MeanModel, sigma =
    VarModel, log = TRUE, checkSymmetry = TRUE)
LogPrior <- mvtnorm::dmvnorm(BetasMode, mean = rep(0, K),
    sigma = Sigma2Mode*cOpt*diag(K), log = TRUE,
    checkSymmetry = TRUE)+log(MCMCpack::dinvgamma(Sigma2Mode
    , a0/2, d0/2))
LogPost1 <- mvtnorm::dmvnorm(BetasMode, mean = bn, sigma =
    Sigma2Mode*Bn, log = TRUE, checkSymmetry = TRUE)
LogPost2 <- log(MCMCpack::dinvgamma(Sigma2Mode, an/2, dn/2))
LogMarLikChib <- LogLik + LogPrior -(LogPost1 + LogPost2)
# Restricted model
anRest <- N + a0; XRest <- X[,-5]
KRest <- dim(XRest)[2]; B0Rest <- cOpt*diag(KRest)
BnRest <- solve(solve(B0Rest)+t(XRest)%*%XRest)
bhatRest <- solve(t(XRest)%*%XRest)%*%t(XRest)%*%Y
b0Rest <- rep(0, KRest)
bnRest <- BnRest%*%(solve(B0Rest)%*%b0Rest+t(XRest)%*%XRest%
    *%bhatRest)
dnRest <- as.numeric(d0 + t(Y-XRest%*%bhatRest)%*%(Y-XRest%*
    %bhatRest)+t(bhatRest - b0Rest)%*%solve(solve(t(XRest)%*
    %XRest)+B0Rest)%*%(bhatRest - b0Rest))
sig2PostRest <- MCMCpack::rinvgamma(S,anRest/2,dnRest/2)
BetasGibbsRest <- sapply(1:S, function(s){MASS::mvrnorm(n =
    1, mu = bnRest, Sigma = sig2PostRest[s]*BnRest)})
BetasModeRest <- apply(BetasGibbsRest, 1, mode_continuous)
Sigma2ModeRest <- mode_continuous(sig2PostRest)
VarModelRest <- Sigma2ModeRest*diag(N)
MeanModelRest <- XRest%*%BetasModeRest
LogLikRest <- mvtnorm::dmvnorm(c(Y), mean = MeanModelRest,
    sigma = VarModelRest, log = TRUE, checkSymmetry = TRUE)
LogPriorRest <- mvtnorm::dmvnorm(BetasModeRest, mean = rep
    (0, KRest), sigma = Sigma2ModeRest*cOpt*diag(KRest), log
     = TRUE, checkSymmetry = TRUE)+log(MCMCpack::dinvgamma(
    Sigma2ModeRest, a0/2, d0/2))
LogPost1Rest <- mvtnorm::dmvnorm(BetasModeRest, mean =
    bnRest, sigma = Sigma2ModeRest*BnRest, log = TRUE,
    checkSymmetry = TRUE)
LogPost2Rest <- log(MCMCpack::dinvgamma(Sigma2ModeRest,
    anRest/2, dnRest/2))
LogMarLikChibRest <- LogLikRest + LogPriorRest -(
    LogPost1Rest + LogPost2Rest)
BFChibs <- exp(LogMarLikChibRest-LogMarLikChib)
BFChibs; 1/BFChibs; 2*log(1/BFChibs)
```

R code. Simulation exercise: Bayes factors

```
# Gelfand-Dey method
GDmarglik <- function(ids, X, Betas, MeanThetas, VarThetas, sig2Post){
  K <- dim(X)[2]; Thetas <- c(Betas[ids,], sig2Post[ids])
  Lognom <- (1/(1-alpha))*mvtnorm::dmvnorm(Thetas, mean = MeanThetas, sigma = VarThetas, log = TRUE, checkSymmetry = TRUE)
  Logden1 <- mvtnorm::dmvnorm(Betas[ids,], mean = rep(0, K), sigma = sig2Post[ids]*cOpt*diag(K), log = TRUE, checkSymmetry = TRUE) + log(MCMCpack::dinvgamma(sig2Post[ids], a0/2, d0/2))
  VarModel <- sig2Post[ids]*diag(N)
  MeanModel <- X%*%Betas[ids,]
  Logden2 <- mvtnorm::dmvnorm(c(Y), mean = MeanModel, sigma = VarModel, log = TRUE, checkSymmetry = TRUE)
  LogGDid <- Lognom - Logden1 - Logden2
  return(LogGDid)
}
sig2Post <- MCMCpack::rinvgamma(S,an/2,dn/2)
Betas <- LaplacesDemon::rmvt(S, bn, Hn, an)
Thetas <- cbind(Betas, sig2Post)
MeanThetas <- colMeans(Thetas); VarThetas <- var(Thetas)
iVarThetas <- solve(VarThetas)
ChiSQ <- sapply(1:S, function(s){(Thetas[s,]-MeanThetas)%*%iVarThetas%*%(Thetas[s,]-MeanThetas)})
alpha <- 0.01; criticalval <- qchisq(1-alpha, K + 1)
idGoodThetas <- which(ChiSQ <= criticalval)
pb <- winProgressBar(title = "progress bar", min = 0, max = S, width = 300)
InvMargLik2 <- NULL
for(s in idGoodThetas){
  LogInvs <- GDmarglik(ids = s, X = X, Betas = Betas, MeanThetas = MeanThetas, VarThetas = VarThetas, sig2Post = sig2Post)
  InvMargLik2 <- c(InvMargLik2, LogInvs)
  setWinProgressBar(pb, s, title=paste( round(s/S*100, 0),"% done"))
}
close(pb); mean(InvMargLik2)
# Restricted model
anRest <- N + a0; XRest <- X[,-5]
KRest <- dim(XRest)[2]; B0Rest <- cOpt*diag(KRest)
BnRest <- solve(solve(B0Rest)+t(XRest)%*%XRest)
bhatRest <- solve(t(XRest)%*%XRest)%*%t(XRest)%*%Y
b0Rest <- rep(0, KRest)
bnRest <- BnRest%*%(solve(B0Rest)%*%b0Rest+t(XRest)%*%XRest%*%bhatRest)
dnRest <- as.numeric(d0 + t(Y-XRest%*%bhatRest)%*%(Y-XRest%*%bhatRest)+t(bhatRest - b0Rest)%*%solve(solve(t(XRest)%*%XRest)+B0Rest)%*%(bhatRest - b0Rest))
HnRest <- as.matrix(Matrix::forceSymmetric(dnRest*BnRest/anRest))
sig2PostRest <- MCMCpack::rinvgamma(S,anRest/2,dnRest/2)
BetasRest <- LaplacesDemon::rmvt(S, bnRest, HnRest, anRest)
ThetasRest <- cbind(BetasRest, sig2PostRest)
MeanThetasRest <- colMeans(ThetasRest)
VarThetasRest <- var(ThetasRest)
iVarThetasRest <- solve(VarThetasRest)
```

R code. Simulation exercise: Bayes factors

```
ChiSQRest <- sapply(1:S, function(s){(ThetasRest[s,]-
    MeanThetasRest)%*%iVarThetasRest%*%(ThetasRest[s,]-
    MeanThetasRest)})
idGoodThetasRest <- which(ChiSQRest <= criticalval)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    S, width = 300)
InvMargLik1 <- NULL
for(s in idGoodThetasRest){
  LogInvs <- GDmarglik(ids = s, X = XRest, Betas = BetasRest
    , MeanThetas = MeanThetasRest, VarThetas = VarThetasRest
    , sig2Post = sig2PostRest)
  InvMargLik1 <- c(InvMargLik1, LogInvs)
  setWinProgressBar(pb, s, title=paste( round(s/S*100, 0),"%
     done"))
}
close(pb); summary(coda::mcmc(InvMargLik1))
mean(InvMargLik1)
BFFD <- exp(mean(InvMargLik2)-mean(InvMargLik1))
BFFD; mean(1/BFFD); 2*log(1/BFFD)
```

10.5 Summary

In this chapter, we introduced Bayesian model averaging (BMA) in generalized linear models. For linear Gaussian models, we perform BMA using three approaches: the Bayesian Information Criterion (BIC) approximation with Occam's window, the Markov Chain Monte Carlo Model Composition (MC3) algorithm, and conditional Bayes factors, which account for endogeneity. Additionally, we show how to perform dynamic Bayesian model averaging in state-space models, where forgetting parameters are used to facilitate computation. For other generalized linear models, such as logit, gamma, and Poisson, we demonstrate how to use the BIC approximation to perform BMA. Finally, we present alternative methods for calculating the marginal likelihood: the Savage-Dickey density ratio, Chib's method, and the Gelfand-Dey method. These methods are particularly useful when the BIC approximation does not perform well due to small or moderate sample sizes.

However, a limitation of standard BMA is its implicit assumption that one of the candidate models is the true data-generating process. As Box famously noted, "Since all models are wrong, the scientist cannot obtain a 'correct' one

by excessive elaboration. On the contrary, following William of Occam, he should seek an economical description of natural phenomena" [52].

This perspective has motivated new developments in BMA that relax the "true model" assumption and instead treat all models as misspecified. In these approaches, the Bayesian average of predictive densities is constructed using leave-one-out (LOO) predictive performance, and model weights are chosen to minimize a predictive loss function. One prominent example is *stacking of predictive distributions*, which, rather than weighting models by marginal likelihood, uses cross-validation (LOO) to assign weights that maximize out-of-sample predictive accuracy [428]. For a comprehensive review of Bayesian methods for aggregating predictive distributions, see [427].

10.6 Exercises

1. The Gaussian linear model specifies $\mathbf{y} = \alpha \mathbf{i_N} + \mathbf{X_m}\boldsymbol{\beta}_\mathbf{m} + \boldsymbol{\mu}_\mathbf{m}$ such that $\boldsymbol{\mu}_m \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I}_n)$, and $\boldsymbol{X}_m$ does not have the column of ones. Assuming that $\pi(\sigma^2) \propto 1/\sigma^2$, $\pi(\alpha) \propto 1$, and $\boldsymbol{\beta}_m|\sigma^2 \sim N(\mathbf{0}_{k_m}, \sigma^2(g_m \boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1})$.

   - Show that the posterior conditional distribution of $\boldsymbol{\beta}_m$ is $N(\boldsymbol{\beta}_{mn}, \sigma^2 \boldsymbol{B}_{mn})$, where $\boldsymbol{\beta}_{mn} = \boldsymbol{B}_{mn}\boldsymbol{X}_m^\top \boldsymbol{y}$ and $\boldsymbol{B}_{mn} = ((1 + g_m)\boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1}$.
   - Show that the marginal likelihood associated with model $\mathcal{M}_m$ is proportional to

   $$p(\boldsymbol{y}|\mathcal{M}_m) \propto \left(\frac{g_m}{1+g_m}\right)^{k_m/2} \left[(\boldsymbol{y} - \bar{y}\boldsymbol{i}_N)^\top (\boldsymbol{y} - \bar{y}\boldsymbol{i}_N) - \frac{1}{1+g_m}(\boldsymbol{y}^\top \boldsymbol{P}_{X_m}\boldsymbol{y})\right]^{-(N-1)/2},$$

   where all parameter are indexed to model $\mathcal{M}_m$, $\boldsymbol{P}_{X_m} = \boldsymbol{X}_m(\boldsymbol{X}_m^\top \boldsymbol{X}_m)^{-1}\boldsymbol{X}_m$ is the projection matrix on the space generated by the columns of $\boldsymbol{X}_m$, and $\bar{y}$ is the sample mean of $\boldsymbol{y}$.

   Hint: Take into account that $\boldsymbol{i}_N^\top \boldsymbol{X}_m = \mathbf{0}_{k_m}$ due to all columns being centered with respect to their means.

2. **Determinants of export diversification I**

   [211] use BMA to study the determinants of export diversification. Use the dataset *10ExportDiversificationHHI.csv* to perform BMA using the BIC approximation and MC3 to check if these two approaches agree.

3. **Simulation exercise of the Markov chain Monte Carlo model composition continues**

Program an algorithm to perform MC3 where the final S models are unique. Use the simulation setting of Section 10.2 increasing the number of regressors to 40, this implies approximately 1.1e+12 models.

4. **Simulation exercise of IV BMA continues**

   Use the simulation setting with endogeneity in Section 10.2 to perform BMA based on the BIC approximation and MC3.

5. **Determinants of export diversification II**

   Use the datasets *11ExportDiversificationHHI.csv* and *12ExportDiversificationHHIInstr.csv* to perform IV BMA assuming that the log of per capita gross domestic product is endogenous (*avglgdpcap*). See [211] for details.

6. Show that the link function in the case of the Bernoulli distribution is $\log\left(\frac{\theta}{1-\theta}\right)$.

7. [329, 330] perform variable selection using the file *13InternetMed.csv*. In this dataset, the dependent variable is an indicator of Internet adoption (*internet*) for 5,000 households in Medellín (Colombia) during the period 2006–2014. This dataset contains information about 18 potential determinants, which implies 262,144 (2^{18}) potential models, considering only variable uncertainty (see these papers for details about the dataset). Perform BMA using the logit link function with this dataset.

8. [372] use the file *14ValueFootballPlayers.csv* to analyze the market value of soccer players in the most important leagues in Europe. In particular, there are 26 potential determinants of the market value (dependent variable) of a stratified sample of 335 soccer players from the five most important leagues in Europe (see [372] for details). Use this dataset to perform BMA using the gamma distribution, setting the default values for Occam's window.

9. Use the dataset *15Fertile2.csv* from [421, p. 547] to perform BMA using the Poisson model with the log link. This dataset contains information about 1,781 women from Botswana in 1988 (for details, see **https://rdrr.io/cran/wooldridge/man/fertil2.html**), with the caveat that we deleted some variables and omitted observations with NA values. The dependent variable is the number of children ever born (`ceb`), which is a count variable, as a function of 19 potential determinants.

10. Perform BMA in the logit model using MC3 and the BIC approximation using the simulation setting of Section 10.3.

11. Use the dataset *19ExchangeRateCOPUSD.csv* to perform dynamic BMA using four different *state-space models* to explain the annual variation of the COP to USD exchange rate:

- Interest rate parity
  $\Delta e_t = \beta_{1t}^{IRP} + \beta_{2t}^{IRP}(i_{t-1}^{Col} - i_{t-1}^{USA}) + \mu_t^{IRP}$
- Purchasing power parity
  $\Delta e_t = \beta_{1t}^{PPP} + \beta_{2t}^{PPP}(\pi_{t-1}^{Col} - \pi_{t-1}^{USA}) + \mu_t^{PPP}$
- Taylor rule
  $\Delta e_t = \beta_{1t}^{Taylor} + \beta_{2t}^{Taylor}(\pi_{t-1}^{Col} - \pi_{t-1}^{USA}) + \beta_{2t}^{Taylor}(g_{t-1}^{Col} - g_{t-1}^{USA}) + \mu_t^{IRP}$
- Money supply
  $\Delta e_t = \beta_{1t}^{Money} + \beta_{2t}^{Money}(g_{t-1}^{Col} - g_{t-1}^{USA}) + \beta_{2t}^{Money}(m_{t-1}^{Col} - m_{t-1}^{USA}) + \mu_t^{Money}$

where *varTRM* (Δe_t) represents the annual variation rate of the exchange rate from COP to USD, *TES_COL10* (i_t^{Col}) and *TES_USA10* (i_t^{USA}) denote the annual return rates of Colombian and U.S. public debts over 10 years, *inflation_COL* (π_t^{Col}) and *inflation_USA* (π_t^{USA}) are the annual inflation rates for Colombia and the U.S., *varISE_COL* (g_t^{Col}) and *varISE_USA* (g_t^{USA}) represent the annual variations of economic activity indices, and *varCOL_M3* (m_t^{Col}) and *varUSA_M3* (m_t^{USA}) are the annual variations of the money supply. In addition, $\mu_t^{\cdot}$ is the stochastic error. The dataset includes monthly variations from January 2006 to November 2023.

Perform Bayesian model averaging using these four models to explain the annual variation of the exchange rate, calculate the posterior model probabilities, and plot the posterior mean and credible interval of β_{2t}^{Money}.

12. Perform a simulation of the dynamic logistic model, where there are 7 ($2^3 - 1$, excluding the model without regressors) competing models originating from 3 regressors: $x_{tk} \sim N(0.5, 0.8^2)$, $k = 2, 3, 4$, and $\beta_1 = 0.5$, β_{2t} is a sequence from 1 to 2 in steps given by $1/T$, and $\beta_{3t} = \begin{Bmatrix} -1, & 1 < t \leq 0.5T \\ 0, & 0.5T < t \leq T \end{Bmatrix}$, with $\beta_4 = 1.2$. Then, $\boldsymbol{x}_t^\top \boldsymbol{\beta}_t = \beta_1 + \beta_{2t}x_{2t} + \beta_{3t}x_{3t} + \beta_4 x_{4t}$, where

$$P[Y_t = 1|\boldsymbol{x}_t, \boldsymbol{\beta}_t] = \frac{\exp(\boldsymbol{x}_t^\top \boldsymbol{\beta}_t)}{1 + \exp(\boldsymbol{x}_t^\top \boldsymbol{\beta}_t)}, \quad t = 1, 2, \ldots, 1100.$$

Use the function *logistic.dma* from the *dma* package to obtain the posterior model probabilities, first setting the forgetting parameter of the models to 0.99, and then to 0.95. Compare the results.

13. Show that

$$\mathbb{E}\left[\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m, \mathcal{M}_m)}\middle|\boldsymbol{y}, \mathcal{M}_m\right] = \frac{1}{p(\boldsymbol{y}|\mathcal{M}_m)},$$

where the expected value is with respect to the posterior distribution given the model $\mathcal{M}_m$, and $q(\boldsymbol{\theta})$ is the proposal distribution whose support is $\boldsymbol{\Theta}$.

Part III

Advanced methods: A brief introduction

11

Semi-parametric and non-parametric models

Non-parametric models are characterized by making minimal assumptions about the data-generating process. Unlike parametric models, which have a finite-dimensional parameter space, non-parametric models often involve infinite-dimensional parameter spaces. A major challenge in non-parametric modeling is the *curse of dimensionality*, as these models require dense data coverage, necessitating large datasets to achieve reliable estimates.

Semi-parametric methods, on the other hand, combine parametric assumptions for part of the model with non-parametric assumptions for the rest. This approach offers a balance between flexibility, tractability and applicability.

In this chapter, we introduce finite Gaussian mixture models (GMM) and Dirichlet mixture processes (DMP), the latter representing an infinite mixture. Both can be used to specify an entire statistical model (nonparametric specification) or to model stochastic error distributions in a semiparametric framework.

Additionally, we present spline models, where the outcome depends linearly on smooth nonparametric functions. To address the curse of dimensionality, we introduce partially linear models, which mitigate this issue while remaining interpretable and flexible for practical applications.

We let other useful Bayesian non-parametric approaches like Bayesian additive random trees (BART) and Gaussian process (GP) for Chapter 12.

11.1 Mixture models

Mixture models naturally arise in situations where a sample consists of draws from different *subpopulations* (*clusters*) that cannot be easily distinguished based on observable characteristics. However, performing inference on specific identified subpopulations can be misleading if the assumed distribution for each cluster is misspecified.

Even when distinct subpopulations do not exist, finite and infinite mixture models provide a useful framework for semi-parametric inference. They

effectively approximate distributions with skewness, excess kurtosis, and multimodality, making them useful for modeling stochastic errors.

In addition, mixture models help capture unobserved heterogeneity. That is, as data modelers, we may observe individuals with identical sets of observable variables but entirely different response variables. These differences cannot be explained solely by sampling variability; rather, they suggest the presence of an unobserved underlying process, independent of the observable features, that accounts for this pattern.

11.1.1 Finite Gaussian mixtures

A finite Gaussian mixture model for regression with H known components assumes that a sample $\boldsymbol{y} = [y_1\ y_2\ \dots\ y_N]^\top$ consists of observations y_i, for $i = 1, 2, \dots, N$, where each y_i is generated from one of the H components, $h = 1, 2, \dots, H$, conditional on the regressors $\boldsymbol{x}_i$. Specifically, we assume

$$y_i \mid \boldsymbol{x}_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2).$$

Thus, the sampling distribution of y_i is given by

$$p(y_i \mid \{\lambda_h, \boldsymbol{\beta}_h, \sigma_h^2\}_{h=1}^H, \boldsymbol{x}_i) = \sum_{h=1}^H \lambda_h \phi(y_i \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2),$$

where $\phi(y_i \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2)$ is the Gaussian density with mean $\boldsymbol{x}_i^\top \boldsymbol{\beta}_h$ and variance σ_h^2, $0 < \lambda_h < 1$ represents the proportion of the population belonging to subpopulation h, and the weights satisfy $\sum_{h=1}^H \lambda_h = 1$.

Then, we allow cross-sectional units to differ according to unobserved clusters (subpopulations) that exhibit homogeneous behavior within each cluster.

To model a finite Gaussian mixture, we introduce an individual cluster indicator or latent class ψ_{ih} such that

$$\psi_{ih} = \begin{cases} 1, & \text{if the } i\text{-th unit is drawn from the } h\text{-th cluster,} \\ 0, & \text{otherwise.} \end{cases}$$

Thus, $P(\psi_{ih} = 1) = \lambda_h$ for all clusters $h = 1, 2, \dots, H$ and units $i = 1, 2, \dots, N$. Note that a high probability of individuals belonging to the same cluster suggests that these clusters capture similar sources of unobserved heterogeneity.

This setting implies that

$$\boldsymbol{\psi}_i = [\psi_{i1}\ \psi_{i2}\ \dots\ \psi_{iH}]^\top \sim \text{Categorical}(\boldsymbol{\lambda}),$$

where $\boldsymbol{\lambda} = [\lambda_1\ \lambda_2\ \dots\ \lambda_H]^\top$ represents the event probabilities.

We know from Subsection 3.2.1 that the Dirichlet prior distribution is

conjugate to the multinomial distribution, where the categorical distribution is a special case in which the number of trials is one. Thus, we assume that

$$\pi(\boldsymbol{\lambda}) \sim \mathrm{Dir}(\boldsymbol{\alpha}_0),$$

where $\boldsymbol{\alpha}_0 = [\alpha_{10}\ \alpha_{20}\ \ldots\ \alpha_{H0}]^\top$, $\alpha_{h0} = 1/H$ is recommended by [150, p. 535].

Observe that we are using a hierarchical structure, as we specify a prior on $\boldsymbol{\lambda}$, which serves as the hyperparameter for the cluster indicators. In addition, we can assume conjugate families for the location and scale parameters to facilitate computation, that is, $\boldsymbol{\beta}_h \sim N(\boldsymbol{\beta}_{h0}, \boldsymbol{B}_{h0})$ and $\sigma_h^2 \sim IG(\alpha_{h0}/2, \delta_{h0}/2)$.

This setting allows to obtain standard conditional posterior distributions:

$$\boldsymbol{\beta}_h \sim N(\boldsymbol{\beta}_{hn}, \boldsymbol{B}_{hn}),$$

where $\boldsymbol{B}_{hn} = (\boldsymbol{B}_{h0}^{-1} + \sigma_h^{-2} \sum_{\{i:\psi_{ih}=1\}} \boldsymbol{x}_i \boldsymbol{x}_i^\top)^{-1}$ and $\boldsymbol{\beta}_{hn} = \boldsymbol{B}_{hn}(\boldsymbol{B}_{h0}^{-1}\boldsymbol{\beta}_{h0} + \sigma_h^{-2} \sum_{\{i:\psi_{ih}=1\}} \boldsymbol{x}_i y_i)$.

$$\sigma_h^2 \sim IG(\alpha_{hn}/2, \delta_{hn}/2),$$

where $\alpha_{hn} = \alpha_{h0} + N_h$, $\delta_{hn} = \delta_{h0} + \sum_{\{i:\psi_{ih}=1\}} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_h)^2$, and N_h is the number of units in cluster h.

$$\boldsymbol{\lambda} \sim \mathrm{Dir}(\boldsymbol{\alpha}_n),$$

where $\boldsymbol{\alpha}_n = [\alpha_{1n}\ \alpha_{2n}\ \ldots\ \alpha_{Hn}]^\top$, and $\alpha_{hn} = \alpha_{h0} + N_h$.

$$\boldsymbol{\psi}_{in} \sim \mathrm{Categorical}(\boldsymbol{\lambda}_n),$$

where $P(\psi_{ih} = 1) = \frac{\lambda_h \phi(y_i | \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2)}{\sum_{j=1}^{H} \lambda_j \phi(y_i | \boldsymbol{x}_i^\top \boldsymbol{\beta}_j, \sigma_j^2)}$.

In general, it is always safer to perform inference in mixture models using informative priors, as non-informative priors may have unintended consequences on posterior inference. One way to facilitate prior elicitation is to work with standardized data, as this removes dependence on measurement units. Another useful approach is to specify the model in log-log form so that the coefficients can be interpreted as elasticities or semi-elasticities. As always in Bayesian inference, it makes sense to perform a sensitivity analysis with respect to the hyperparameters.

Mixture models have the label-switching identification problem, meaning they are nonidentifiable because the distribution remains unchanged if the group labels are permuted [405]. For instance, a mixture model with two components can be characterized by $\{\lambda_1, \boldsymbol{\beta}_1, \sigma_1^2\}$ for the first cluster and $\{1 - \lambda_1, \boldsymbol{\beta}_2, \sigma_2^2\}$ for the second. However, an alternative characterization is $\{1 - \lambda_1, \boldsymbol{\beta}_2, \sigma_2^2\}$ for cluster 1 and $\{\lambda_1, \boldsymbol{\beta}_1, \sigma_1^2\}$ for cluster 2. This parametrization yields exactly the same likelihood as the first one, meaning any permutation of the cluster labels leaves the likelihood unchanged. Consequently, the posterior draws of each component-specific parameter target the same distribution.

Label switching may pose challenges when performing inference on specific mixture components, such as in the regression analysis presented here. However, it is not an issue when inference on specific components is unnecessary, as in cases where mixtures are used to model stochastic errors in semi-parametric settings. In the former case, post-processing strategies can mitigate the issue, such as *random permutation of latent classes* (see the simulation exercise below, [150, p. 534] and Algorithm 3.5 in [140, p. 82]).

A semi-parametric regression imposes a specific structure in part of the model and uses flexible assumptions in another part, for instance

$$y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \mu_i,$$
$$p(\mu_i \mid \{\lambda_h, \mu_h, \sigma_h^2\}_{h=1}^H) = \sum_{h=1}^H \lambda_h \phi(\mu_i \mid \mu_h, \sigma_h^2).$$

Thus, the distribution of the stochastic error is a finite Gaussian mixture. Note that the mean of the stochastic error is not equal to zero; consequently, the intercept in the regression should be removed, as these two parameters are not separately identifiable [405]. Additionally, this approach allows for multiple modes and asymmetric distributions of the stochastic errors, providing greater flexibility.

We can use a Gibbs sampling algorithm in this semi-parametric specification if we assume conjugate families. The difference from the previous setting is that we have the same slope parameters; thus, $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$. Additionally, we must specify the prior distribution for the means of the stochastic errors, given by $\mu_h \sim N(\mu_{h0}, \sigma_{\mu 0}^2)$. Then, the posterior distributions are:

$$\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{h=1}^H \sum_{\{i:\psi_{ih}=1\}} \sigma_h^{-2} \boldsymbol{x}_i \boldsymbol{x}_i^\top)^{-1}$ and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{h=1}^H \sum_{\{i:\psi_{ih}=1\}} \sigma_h^{-2} \boldsymbol{x}_i (y_i - \mu_h))$.

$$\sigma_h^2 \sim IG(\alpha_{hn}/2, \delta_{hn}/2),$$

where $\alpha_{hn} = \alpha_{h0} + N_h$, $\delta_{hn} = \delta_{h0} + \sum_{\{i:\psi_{ih}=1\}} (y_i - \mu_h - \boldsymbol{x}_i^\top \boldsymbol{\beta}_h)^2$, and N_h is the number of units in cluster h.

$$\mu_h \sim N(\mu_{hn}, \sigma_{hn}^2),$$

where $\sigma_{hn}^2 = \left(\frac{1}{\sigma_h^2} + \frac{N_h}{\sigma_h^2}\right)^{-1}$ and $\mu_{hn} = \sigma_{hn}^2 \left(\frac{\mu_{h0}}{\sigma_{\mu 0}^2} + \frac{\sum_{\{i:\psi_{ih}=1\}} (y_i - \boldsymbol{x}_i \boldsymbol{\beta})}{\sigma_h^2}\right)$.

$$\boldsymbol{\lambda} \sim \mathrm{Dir}(\boldsymbol{\alpha}_n),$$

where $\boldsymbol{\alpha}_n = [\alpha_{1n}\ \alpha_{2n}\ \ldots\ \alpha_{Hn}]^\top$, and $\alpha_{hn} = \alpha_{h0} + N_h$.

$$\boldsymbol{\psi}_{in} \sim \mathrm{Categorical}(\boldsymbol{\lambda}_n),$$

where $P(\psi_{ih} = 1) = \frac{\lambda_h \phi(y_i - \mu_h - \boldsymbol{x}_i^\top \boldsymbol{\beta} | \mu_h, \sigma_h^2, \boldsymbol{\beta})}{\sum_{j=1}^{H} \lambda_j \phi(y_i - \mu_j - \boldsymbol{x}_i^\top \boldsymbol{\beta} | \mu_j, \sigma_j^2, \boldsymbol{\beta})}$.

A potential limitation of finite mixture models is the need to specify the number of components in advance. One approach is to estimate the model for different values of H and then compute the marginal likelihood to select the model best supported by the data. However, this procedure can be tedious. A simpler strategy is to set H large enough (e.g., 10 components), assign $\alpha_{h0} = 1/H$, and perform an initial run of the algorithm. If we are not interested in the specific composition of clusters, this approach is sufficient. Otherwise, the posterior distribution of H can be obtained by tracking the number of nonempty clusters in each iteration. In a second run, H can then be fixed at the mode of this posterior distribution.

However, the previous approaches ultimately fix the number of components. Consequently, finite mixtures cannot be considered a non-parametric method [349], as they lack an automatic mechanism to increase H as the sample size grows. An alternative is to avoid pre-specifying the number of components altogether by using a Dirichlet process mixture (DPM). This is the topic of the next section.

Example: Simulation exercises

First, let's illustrate the label-switching issue using a simple model without regressors, assuming the same known variance. Consider the following distribution:

$$p(y_i) = 0.75\phi(y_i \mid \beta_{01}, 1^2) + 0.25\phi(y_i \mid \beta_{02}, 1^2), \quad i = 1, 2, \ldots, 500.$$

Initially, we set $\beta_{01} = 0.5$ and $\beta_{02} = 2.5$. We perform 1,000 MCMC iterations, with a burn-in period of 500 and a thinning factor of 2. The following code demonstrates how to implement the Gibbs sampler using a prior normal distribution with mean 0 and variance 10, with the hyperparameters of the Dirichlet distribution set to 1/2.

R code. Simulation exercise: Label switching issue

```
      ####### Simulation exercise: Label swithching issue ##############
rm(list = ls()); set.seed(010101); library(ggplot2)
# Simulate data from a 2-component mixture model
n <- 500
z <- rbinom(n, 1, 0.75)  # Latent class indicator
y <- ifelse(z == 0, rnorm(n, 0.5, 1), rnorm(n, 2.5, 1))
data <- data.frame(y)
# Plot
ggplot(data, aes(x = y)) +
geom_density(fill = "blue", alpha = 0.3) + labs(title = "Density Plot", x = "y", y = "Density") + theme_minimal()
# Hyperparameters
mu0 <- 0; sig2mu0 <- 10; H <- 2; a0h <- rep(1/H, H)
# MCMC parameters
mcmc <- 1000; burnin <- 500; tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
Postmu <- function(yh){
  Nh <- length(yh)
  sig2mu <- (1/sig2mu0 + Nh)^(-1)
  mun <- sig2mu*(mu0/sig2mu0 + sum(yh))
  mu <- rnorm(1, mun, sig2mu^0.5)
  return(mu)
}
PostPsi <- matrix(NA, tot, n); PostMu <- matrix(NA, tot, H)
PostLambda <- rep(NA, tot)
Id1 <- which(y <= 1) # 1 is from inspection of the density plot of y
Id2 <- which(y > 1)
N1 <- length(Id1); N2 <- length(Id2)
Lambda <- c(N1/n, N2/n)
MU <- c(mean(y[Id1]), mean(y[Id2])); Psi <- rep(NA, n)
pb <- winProgressBar(title = "progress bar", min = 0, max = tot, width = 300)
for(s in 1:tot){
  for(i in 1:n){
    lambdai <- NULL
    for(h in 1:H){
      lambdaih <- Lambda[h]*dnorm(y[i], MU[h], 1)
      lambdai <- c(lambdai, lambdaih)
    }
    Psi[i] <- sample(1:H, 1, prob = lambdai)
  }
  PostPsi[s, ] <- Psi
  for(h in 1:H){
    idh <- which(Psi == h);     MU[h] <- Postmu(yh = y[idh])
  }
  PostMu[s,] <- MU;
  Lambda <- sort(MCMCpack::rdirichlet(1, a0h + table(Psi)), decreasing = TRUE)
  PostLambda[s] <- Lambda[1]
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0), "% done"))
}
close(pb)
keep <- seq(burnin, tot, thin)
PosteriorMUs <- coda::mcmc(PostMu[keep,])
summary(PosteriorMUs); plot(PosteriorMUs)
```

R code. Simulation exercise: Label switching issue

```
dfMU <- data.frame(mu1 = PostMu[keep,1], mu2 = PostMu[keep
    ,2])
# Plot
require(latex2exp)
ggplot(dfMU) + geom_density(aes(x = mu1, color = "mu1"),
    linewidth = 1) + geom_density(aes(x = mu2, color = "mu2"
    ), linewidth = 1) + labs(title = "Density Plot", x = TeX
    ("$\\mu$"), y = "Density", color = "Variable") + theme_
    minimal() + scale_color_manual(values = c("mu1" = "blue"
    , "mu2" = "red"))
```

Figure 11.1 shows the posterior densities of the location parameters. The posterior means are 0.42 and 2.50, with 95% credible intervals of (0.07, 0.71) and (2.34, 2.65), respectively. The posterior mean of the probability is 0.27, with a 95% credible interval of (0.19, 0.35). Note that in this simple simulation exercise, we did not observe unintended consequences from using non-informative priors and not standardizing the data. However, real-world applications should take these aspects into account.

FIGURE 11.1
Posterior distributions: Mean parameters, population values $\beta_{01} = 0.5$ and $\beta_{02} = 2.5$.

We perform the same exercise assuming $\beta_{01} = 0.5$ and $\beta_{02} = 1$. Figure 11.2 shows the posterior densities, where we observe significant overlap. The posterior means are 0.77 in both cases, with 95% credible intervals of (0.40, 1.05) and (-0.44, 1.71). The posterior mean of the probability is 0.84.

FIGURE 11.2
Posterior distributions: Mean parameters, population values $\beta_{01} = 0.5$ and $\beta_{02} = 1$.

In the second setting, the posterior draws of the Gibbs sampler can switch between the two means because they are relatively close. This situation contrasts with the first example, where there is a relatively large separation between the means, resulting in a region of the parameter space with zero probability (see the flat region between the two posterior distributions in Figure 11.1). The key point is that, given a sufficiently large number of Gibbs sampler iterations, the algorithm should eventually explore the entire parameter space and encounter the label-switching issue. This occurs because both posterior chains should exhibit similar behavior, as they are targeting the same distribution.

We can implement *random permutation of latent classes* to address this issue. This involves sampling a random permutation of the labels at each iteration of the MCMC algorithm. For example, with three clusters, there are $3! = 6$ possible label permutations. Let the permutations be labeled as $\boldsymbol{p}_k = \{p_k(1), p_k(2), \ldots, p_k(H)\}, k = 1, 2, \ldots, H!$. At the end of each iteration in the MCMC algorithm, we randomly select one of the permutations $\boldsymbol{p}_k$ and replace the cluster probabilities $\lambda_1^{(s)}, \ldots, \lambda_H^{(s)}$ with $\lambda_{p_k(1)}^{(s)}, \ldots, \lambda_{p_k(H)}^{(s)}$. We

apply the same permutation to $\boldsymbol{\beta}^{(s)}$, $\sigma^{2(s)}$, and $\boldsymbol{\psi}_i^{(s)}$, for $i = 1, 2, \ldots, n$. The following algorithm illustrates how to implement this in our simple example.

R code. Simulation exercise: Random permutation of latent classes

```
###### Permutations ######
rm(list = ls()); set.seed(010101); library(ggplot2)
# Simulate data from a 2-component mixture model
n <- 500
z <- rbinom(n, 1, 0.75)  # Latent class indicator
y <- ifelse(z == 0, rnorm(n, 0.5, 1), rnorm(n, 2.5, 1))
# Hyperparameters
mu0 <- 0; sig2mu0 <- 10; H <- 2; a0h <- rep(1/H, H)
# MCMC parameters
mcmc <- 2000; burnin <- 500
tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
Postmu <- function(yh){
  Nh <- length(yh)
  sig2mu <- (1/sig2mu0 + Nh)^(-1)
  mun <- sig2mu*(mu0/sig2mu0 + sum(yh))
  mu <- rnorm(1, mun, sig2mu^0.5)
  return(mu)
}
PostPsi <- matrix(NA, tot, n); PostMu <- matrix(NA, tot, H)
PostLambda <- rep(NA, tot)
Id1 <- which(y <= 1); Id2 <- which(y > 1)
N1 <- length(Id1); N2 <- length(Id2)
Lambda <- c(N1/n, N2/n); MU <- c(mean(y[Id1]), mean(y[Id2]))
Psi <- rep(NA, n); per1 <- c(1,2); per2 <- c(2,1)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  for(i in 1:n){
    lambdai <- NULL
    for(h in 1:H){
      lambdaih <- Lambda[h]*dnorm(y[i], MU[h], 1)
      lambdai <- c(lambdai, lambdaih)
    }
    Psi[i] <- sample(1:H, 1, prob = lambdai)
  }
  for(h in 1:H){
    idh <- which(Psi == h)
    MU[h] <- Postmu(yh = y[idh])
  }
  Lambda <- MCMCpack::rdirichlet(1, a0h + table(Psi))
  # Permutations
  labels <- sample(1:2, 1, prob = c(0.5, 0.5))
  if(labels == 2){
    Lambda <- Lambda[per2]
    MU <- MU[per2]
    for(i in 1:n){
      if(Psi[i] == 1){Psi[i] <- 2
      }else{Psi[i] <- 1}
    }
  }
  PostPsi[s, ] <- Psi; PostMu[s,] <- MU
  PostLambda[s] <- Lambda[1]
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
```

R code. Simulation exercise: Random permutation of latent classes

```
keep <- seq(burnin, tot, thin)
PosteriorMUs <- coda::mcmc(PostMu[keep,])
summary(PosteriorMUs)
plot(PosteriorMUs)
dfMU <- data.frame(mu1 = PostMu[keep,1], mu2 = PostMu[keep
    ,2])
# Plot
require(latex2exp)
ggplot(dfMU) +
geom_density(aes(x = mu1, color = "mu1"), linewidth = 1) +
    # First density plot
geom_density(aes(x = mu2, color = "mu2"), linewidth = 1) +
    # Second density plot
labs(title = "Density Plot", x = TeX("$\\mu$"), y = "Density
    ", color = "Variable") +
theme_minimal() +
scale_color_manual(values = c("mu1" = "blue", "mu2" = "red")
    )  # Custom colors
PosteriorLAMBDA <- coda::mcmc(PostLambda[keep])
summary(PosteriorLAMBDA)
plot(PosteriorLAMBDA)

```

Figure 11.3 shows the posterior distributions from the random permutation of latent classes in the first simulation setting. We observe that both posterior distributions look similar, as both are targeting a bimodal distribution given by two clusters.

In the following setting we simulate a simple regression mixture with two components such that $\psi_{i1} \sim \text{Ber}(0.5)$, consequently, $\psi_{i2} = 1 - \psi_{i1}$, and assume one regressor, $x_i \sim N(0, 1)$, $i = 1, 2, \ldots, 1,000$. Then,

$$p(y_i \mid \boldsymbol{x}_i) = 0.5\phi(y_i \mid 2 + 1.5x_i, 1^2) + 0.5\phi(y_i \mid -1 + 0.5x_i, 0.8^2).$$

The following code shows how to perform inference in this model, assuming $N(0, 5)$ and $N(0, 2)$ priors for the intercepts and slopes, respectively. Additionally, we use a $Cauchy(0, 2)$ prior truncated at 0 for the standard deviations, and a $Dirichlet(1, 1)$ prior for the probabilities. We use the *brms* package in **R**, which in turn uses *Stan*, setting number of MCMC iterations 2,000, a burn-in (warm-up) equal to 1,000, and 4 chains. Remember that *Stan* software uses Hamiltonian Monte Carlo.

The following code performs inference in this simulation from scratch using Gibbs sampling. We do not implement the random permutation of latent classes algorithm for facilitating exposition and comparability with the results

FIGURE 11.3
Posterior distributions: Mean parameters, population values $\beta_{01} = 0.5$ and $\beta_{02} = 2.5$.

from the package *brms*. We use non-informative priors, setting $\alpha_{h0} = \delta_{h0} = 0.01$, $\boldsymbol{\beta}_{h0} = \mathbf{0}_2$, $\boldsymbol{B}_{h0} = \boldsymbol{I}_2$, and $\boldsymbol{\alpha}_0 = [1/2\ 1/2]^\top$. The number of MCMC iterations is 5,000, the burn-in is 1,000, and the thinning parameter is 2. In general, the Gibbs sampler appears to yield good posterior results as all 95% intervals encompass the population parameters.

R code. Simulation exercise: Gaussian mixture with 2 components using brms package

```r
####### Simulation exercise: Gaussian mixture: 2 components ##############
rm(list = ls())
set.seed(010101)
library(brms)
library(ggplot2)

# Simulate data from a 2-component mixture model
n <- 1000
x <- rnorm(n)
z <- rbinom(n, 1, 0.5)  # Latent class indicator
y <- ifelse(z == 0, rnorm(n, 2 + 1.5*x, 1), rnorm(n, -1 + 0.5*x, 0.8))
data <- data.frame(y, x)

# Plot
ggplot(data, aes(x = y)) +
geom_density(fill = "blue", alpha = 0.3) +  # Density plot with fill color
labs(title = "Density Plot", x = "y", y = "Density") +
theme_minimal()

# Define priors
priors <- c(
set_prior("normal(0, 5)", class = "Intercept", dpar = "mu1"),  # First component intercept
set_prior("normal(0, 5)", class = "Intercept", dpar = "mu2"),  # Second component intercept
set_prior("normal(0, 2)", class = "b", dpar = "mu1"),  # First component slope
set_prior("normal(0, 2)", class = "b", dpar = "mu2"),  # Second component slope
set_prior("cauchy(0, 2)", class = "sigma1", lb = 0),  # First component sigma
set_prior("cauchy(0, 2)", class = "sigma2", lb = 0),  # Second component sigma
set_prior("dirichlet(1, 1)", class = "theta")  # Mixing proportions
)

# Fit a 2-component Gaussian mixture regression model
fit <- brm(
bf(y ~ 1 + x, family = mixture(gaussian, gaussian)),  # Two normal distributions
data = data,
prior = priors,
chains = 4, iter = 2000, warmup = 1000, cores = 4
)
prior_summary(fit) # Summary of priors
summary(fit) # Summary of posterior draws
plot(fit) # Plots of posterior draws

```

R code. Simulation exercise: Gaussian mixture with 2 components using Gibbs sampling

```
########### Perform inference from scratch ###############
rm(list = ls()); set.seed(010101)
library(brms); library(ggplot2)
# Simulate data from a 2-component mixture model
n <- 1000
x <- rnorm(n)
z <- rbinom(n, 1, 0.5)  # Latent class indicator
y <- ifelse(z == 0, rnorm(n, 2 + 1.5*x, 1), rnorm(n, -1 +
    0.5*x, 0.8))
# Hyperparameters
d0 <- 0.001; a0 <- 0.001
b0 <- rep(0, 2); B0 <- diag(2); B0i <- solve(B0)
a01 <- 1/2; a02 <- 1/2
# MCMC parameters
mcmc <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
PostSig2 <- function(Betah, Xh, yh){
  Nh <- length(yh); an <- a0 + Nh
  dn <- d0 + t(yh - Xh%*%Betah)%*%(yh - Xh%*%Betah)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2h, Xh, yh){
  Bn <- solve(B0i + sig2h^(-1)*t(Xh)%*%Xh)
  bn <- Bn%*%(B0i%*%b0 + sig2h^(-1)*t(Xh)%*%yh)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostBetas1 <- matrix(0, mcmc+burnin, 2)
PostBetas2 <- matrix(0, mcmc+burnin, 2)
PostSigma21 <- rep(0, mcmc+burnin)
PostSigma22 <- rep(0, mcmc+burnin)
PostPsi <- matrix(0, mcmc+burnin, n)
PostLambda <- rep(0, mcmc+burnin)
Id1 <- which(y<1) # 1 is from inspection of the density plot
    of y
N1 <- length(Id1); Lambda1 <- N1/n
Id2 <- which(y>=1)
N2 <- length(Id2); Lambda2 <- N2/n
Reg1 <- lm(y ~ x, subset = Id1)
SumReg1 <- summary(Reg1); Beta1 <- Reg1$coefficients
sig21 <- SumReg1$sigma^2
Reg2 <- lm(y ~ x, subset = Id2); SumReg2 <- summary(Reg2)
Beta2 <- Reg2$coefficients
sig22 <- SumReg2$sigma^2
X <- cbind(1, x); Psi <- rep(NA, n)
```

R code. Simulation exercise: Gaussian mixture with 2 components using Gibbs sampling

```
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  for(i in 1:n){
    lambdai1 <- Lambda1*dnorm(y[i], X[i,]%*%Beta1, sig21
    ^0.5)
    lambdai2 <- Lambda2*dnorm(y[i], X[i,]%*%Beta2, sig22
    ^0.5)
    Psi[i] <- sample(c(1,2), 1, prob = c(lambdai1, lambdai2)
    )
  }
  PostPsi[s, ] <- Psi
  Id1 <- which(Psi == 1); Id2 <- which(Psi == 2)
  N1 <- length(Id1); N2 <- length(Id2)
  sig21 <- PostSig2(Betah = Beta1, Xh = X[Id1, ], yh = y[Id1
    ])
  sig22 <- PostSig2(Betah = Beta2, Xh = X[Id2, ], yh = y[Id2
    ])
  PostSigma21[s] <- sig21; PostSigma22[s] <- sig22
  Beta1 <- PostBeta(sig2h = sig21, Xh = X[Id1, ], yh = y[Id1
    ])
  Beta2 <- PostBeta(sig2h = sig22, Xh = X[Id2, ], yh = y[Id2
    ])
  PostBetas1[s,] <- Beta1; PostBetas2[s,] <- Beta2
  Lambda <- sort(MCMCpack::rdirichlet(1, c(a01 + N1, a02 +
    N2)), decreasing = TRUE)
  Lambda1 <- Lambda[1]; Lambda2 <- Lambda[2]
  PostLambda[s] <- Lambda1
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorBetas1 <- coda::mcmc(PostBetas1[keep,])
summary(PosteriorBetas1)
plot(PosteriorBetas1)
PosteriorBetas2 <- coda::mcmc(PostBetas2[keep,])
summary(PosteriorBetas2)
plot(PosteriorBetas2)
PosteriorSigma21 <- coda::mcmc(PostSigma21[keep])
summary(PosteriorSigma21)
plot(PosteriorSigma21)
PosteriorSigma22 <- coda::mcmc(PostSigma22[keep])
summary(PosteriorSigma22)
plot(PosteriorSigma22)
```

Let's perform another simulation exercise in which we conduct a semi-parametric analysis where the stochastic error follows a Student's t-distribution with 3 degrees of freedom. Specifically,

$$y_i = 1 - 0.5x_{i1} + 1.5x_{i2} + \mu_i,\ i = 1, 2, \ldots, 500.$$

The variables x_{i1} and x_{i2} are standard normally distributed. Let's set $H = 5$, and use non-informative priors setting $\alpha_{h0} = \delta_{h0} = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_2$, $\boldsymbol{B}_0 = \boldsymbol{I}_2$, $\mu_{h0} = 0$, $\sigma^2_{\mu 0} = 10$ and $\boldsymbol{\alpha}_0 = [1/H\ \ \ldots\ \ 1/H]^\top$. Use 6,000 MCMC iterations, burn-in equal to 4,000, and thinning parameter equal to 2. In this exercise, there is no need to address the label-switching issue, as we are not specifically interested in the individual components of the posterior distributions of the clusters. Exercise 1 asks how to get the posterior density of the stochastic errors in semi-parametric specifications.

We can see from the posterior estimates that three components disappear after the burn-in iterations. The 95% credible intervals encompass the population values of the slope parameters. The 95% credible intervals for the probabilities are (0.70, 0.89) and (0.11, 0.30), and the 95% credible interval for the weighted average of the intercepts encompasses the population parameter.

R code. Simulation exercise: Semi-parametric model using Gibbs sampling

```r
rm(list = ls()); set.seed(010101)
library(ggplot2)
# Simulate data from a 2-component mixture model
n <- 500
x1 <- rnorm(n); x2 <- rnorm(n)
X <- cbind(x1,x2); B <- c(-0.5, 1.5)
u <- rt(n, 3); y <- 1 + X%*%B + u
Reg <- lm(y ~ X)
Res <- Reg$residuals
data <- data.frame(Res)
# Plot
ggplot(data, aes(x = Res)) +
geom_density(fill = "blue", alpha = 0.3) +  # Density plot with fill color
labs(title = "Density Plot", x = "Residuals", y = "Density") +
theme_minimal()
# Hyperparameters
d0 <- 0.001; a0 <- 0.001; b0 <- rep(0, 2)
B0 <- diag(2); B0i <- solve(B0)
mu0 <- 0; sig2mu0 <- 10; H <- 5; a0h <- rep(1/H, H)
# MCMC parameters
mcmc <- 2000; burnin <- 4000
tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
PostSig2 <- function(Beta, muh, Xh, yh){
  Nh <- length(yh); an <- a0 + Nh
  dn <- d0 + t(yh - muh - Xh%*%Beta)%*%(yh - muh - Xh%*%Beta)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2, mu, X, y, Psi){
  XtX <- matrix(0, 2, 2); Xty <- matrix(0, 2, 1)
  Hs <- length(mu)
  for(h in 1:Hs){
    idh <- which(Psi == h)
    if(length(idh) == 1){
      Xh <- matrix(X[idh,], 1, 2)
      XtXh <- sig2[h]^(-1)*t(Xh)%*%Xh
      yh <- y[idh]
      Xtyh <- sig2[h]^(-1)*t(Xh)%*%(yh - mu[h])
    }else{
      Xh <- X[idh,]
      XtXh <- sig2[h]^(-1)*t(Xh)%*%Xh
      yh <- y[idh]
      Xtyh <- sig2[h]^(-1)*t(Xh)%*%(yh - mu[h])
    }
    XtX <- XtX + XtXh; Xty <- Xty + Xtyh
  }
  Bn <- solve(B0i + XtX); bn <- Bn%*%(B0i%*%b0 + Xty)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
```

R code. Simulation exercise: Semi-parametric model using Gibbs sampling

```
Postmu <- function(sig2h, Beta, Xh, yh){
  Nh <- length(yh)
  sig2mu <- (1/sig2mu0 + Nh/sig2h)^(-1)
  mun <- sig2mu*(mu0/sig2mu0 + sum((yh - Xh%*%Beta))/sig2h)
  mu <- rnorm(1, mun, sig2mu^0.5)
  return(mu)
}
PostBetas <- matrix(0, mcmc+burnin, 2)
PostPsi <- matrix(0, mcmc+burnin, n)
PostSigma2 <- list(); PostMu <- list()
PostLambda <- list()
Resq <- quantile(Res, c(0.2, 0.4, 0.6, 0.8))
Id1 <- which(Res <= Resq[1])
Id2 <- which(Res > Resq[1] & Res <= Resq[2])
Id3 <- which(Res > Resq[2] & Res <= Resq[3])
Id4 <- which(Res > Resq[3] & Res <= Resq[4])
Id5 <- which(Res > Resq[4])
Nh <- rep(n/H, H); Lambda <- rep(1/H, H)
MU <- c(mean(Res[Id1]), mean(Res[Id2]), mean(Res[Id3]), mean(Res[Id4]), mean(Res[Id5]))
Sig2 <- c(var(Res[Id1]), var(Res[Id2]), var(Res[Id3]), var(Res[Id4]), var(Res[Id5]))
Beta <- Reg$coefficients[2:3]
Psi <- rep(NA, n); Hs <- length(MU)
pb <- winProgressBar(title = "progress bar", min = 0, max = tot, width = 300)
for(s in 1:tot){
  for(i in 1:n){
    lambdai <- NULL
    for(h in 1:Hs){
      lambdaih <- Lambda[h]*dnorm(y[i] - X[i,]%*%Beta, MU[h], Sig2[h]^0.5)
      lambdai <- c(lambdai, lambdaih)
    }
    Psi[i] <- sample(1:Hs, 1, prob = lambdai)
  }
  PostPsi[s, ] <- Psi
  Hs <- length(table(Psi))
  for(h in 1:Hs){
    idh <- which(Psi == h)
    Sig2[h] <- PostSig2(Beta = Beta, muh = MU[h], Xh = X[idh,], yh = y[idh])
    MU[h] <- Postmu(sig2h = Sig2[h], Beta = Beta, Xh = X[idh,], yh = y[idh])
  }
  PostSigma2[[s]] <- Sig2
  PostMu[[s]] <- MU
  Beta <- PostBeta(sig2 = Sig2, mu = MU, X = X, y = y, Psi = Psi)
  PostBetas[s,] <- Beta
  Lambda <- sort(MCMCpack::rdirichlet(1, a0h[1:Hs] + table(Psi)), decreasing = TRUE)
  PostLambda[[s]] <- Lambda
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0), "% done"))
}
close(pb)
```

R code. Simulation exercise: Semi-parametric model using Gibbs sampling

```
keep <- seq(burnin, tot, thin)
PosteriorBetas <- coda::mcmc(PostBetas[keep,])
summary(PosteriorBetas)
plot(PosteriorBetas)
PosteriorPsi <- PostPsi[keep,]
Clusters <- sapply(1:length(keep), function(i){length(table(
    PosteriorPsi[i,]))})
NClus <- 2
PosteriorSIGMA <- matrix(NA, length(keep), NClus)
PosteriorMU <- matrix(NA, length(keep), NClus)
PosteriorLAMBDA <- matrix(NA, length(keep), NClus)
l <- 1
for (s in keep){
  PosteriorSIGMA[l,] <- PostSigma2[[s]][1:NClus]
  PosteriorMU[l,] <- PostMu[[s]][1:NClus]
  PosteriorLAMBDA[l,] <- PostLambda[[s]][1:NClus]
  l <- l + 1
}

summary(coda::mcmc(PosteriorSIGMA))
summary(coda::mcmc(PosteriorMU))
summary(coda::mcmc(PosteriorLAMBDA))
```

11.1.2 Direchlet processes

A Dirichlet process (DP) is a probability distribution over probability distributions, and a generalization of the Dirichlet distribution. It was introduced by [128], and it is commonly used as a prior for unknown distributions, making it particularly useful in non-parametric settings. Unlike finite Gaussian mixture models, a Dirichlet process does not require pre-specifying the number of components, allowing for greater flexibility in modeling complex data structures. In this sense, DPs can be viewed as the limiting case of finite mixtures when the number of components approaches infinity.

A Dirichlet process $G \sim DP(\alpha G_0)$ is defined by a precision parameter $\alpha > 0$, and a base probability measure G_0 on a probability space Ω.[1] The DP assigns probability $G(B)$ to any (measurable) set B in Ω such that for any finite (measurable) partition $\{B_1, B_2, \ldots, B_k\}$ of Ω,

[1] Measurability ensures that we can compute probabilities and expectations properly. If a set is not measurable, we cannot define its probability meaningfully.

$$G(B_1), G(B_2), \ldots, G(B_k) \sim \text{Dirichlet}(\alpha G_0(B_1), \alpha G_0(B_2), \ldots, \alpha G_0(B_k)).$$

In particular, $\mathbb{E}\left[G(B)\right] = G_0(B)$ and $\mathbb{V}ar\left[G(B)\right] = G_0(B)\left[1 - G_0(B)\right]/(1+\alpha)$ [288, p. 8]. Observe that as $\alpha \to \infty$, G concentrates at G_0, which is why α is called the precision parameter.

A key property of the DP is its discrete nature, which allows it to be expressed as

$$G(\cdot) = \sum_{h=1}^{\infty} \lambda_h \delta_{\boldsymbol{\theta}_h}(\cdot),$$

where λ_h is the probability mass at $\boldsymbol{\theta}_h$, and $\delta_{\boldsymbol{\theta}_h}(\cdot)$ denotes the Dirac measure that assigns mass one to the atom $\boldsymbol{\theta}_h$.[2] Given this property, a particularly useful construction of the DP is the *stick-breaking* representation [373], which is given by

$$\begin{aligned} G(\cdot) &= \sum_{h=1}^{\infty} \lambda_h \delta_{\boldsymbol{\theta}_h}(\cdot), \\ \lambda_h &= V_h \prod_{m<h} (1 - V_m), \quad V_h \sim \text{Beta}(1, \alpha), \\ \boldsymbol{\theta}_h &\overset{iid}{\sim} G_0. \end{aligned}$$

The intuition behind this representation is straightforward. We begin with a stick of length 1 and break off a random proportion V_1 drawn from a $\text{Beta}(1, \alpha)$ distribution. This assigns a probability mass of $\lambda_1 = V_1$ to $\boldsymbol{\theta}_1$, which is sampled from G_0. Next, from the remaining stick of length $(1 - V_1)$, we break off a fraction proportional to $V_2 \sim \text{Beta}(1, \alpha)$, assigning a probability mass of $\lambda_2 = V_2(1 - V_1)$ to a new draw $\boldsymbol{\theta}_2$ from G_0. This process continues indefinitely.

Since $\mathbb{E}(V_h) = \frac{1}{1+\alpha}$, as $\alpha \to \infty$, we have $\mathbb{E}(V_h) \to 0$. Consequently, the DP places mass on a large number of atoms, leading to convergence to the base distribution G_0.

Note that DPs give as realizations discrete distributions, this poses challenges when working with continuous distributions. One way to overcome this limitation is to use the DP as a mixing distribution for simple parametric distributions, such as the normal distribution [127]. This leads to Dirichlet process mixtures (DPM), which are defined as

$$f_G(y) = \int f_{\boldsymbol{\theta}}(y) dG(\boldsymbol{\theta}).$$

Observe that the mixing measure G is discrete when a DP is the prior,

[2] An atom is a set that has positive probability but cannot be further decomposed into smaller sets with strictly smaller probability.

with mass concentrated at an infinite number of atoms $\boldsymbol{\theta}_h$. Consequently, if $f_{\boldsymbol{\theta}}(y)$ follows a Gaussian distribution, the resulting mixture resembles a finite Gaussian mixture. However, unlike finite mixtures, the DP-based approach eliminates the need to pre-specify the number of components, as it provides an automatic mechanism for determining them.

Thus, if we assume $y \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2)$, then the DPM is

$$p(y_i \mid \{\lambda_h, \boldsymbol{\beta}_h, \sigma_h^2\}_{h=1}^{\infty}, \boldsymbol{x}_i) = \sum_{h=1}^{\infty} \lambda_h \phi(y_i \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2),$$

where λ_h are drawn from the stick-breaking representation of the DP, and $\boldsymbol{\theta}_h = [\boldsymbol{\beta}_h^\top \ \sigma_h^2]^\top$.

This mixture model can be expressed in a hierarchical structure:

$$\begin{aligned} y_i \mid \boldsymbol{\theta}_i &\overset{ind}{\sim} f_{\boldsymbol{\theta}_i} \\ \boldsymbol{\theta}_i \mid G &\overset{iid}{\sim} G \\ G \mid \alpha, G_0 &\sim DP(\alpha G_0). \end{aligned}$$

Note that the hierarchical representation induces specific unit parameters, leading to a probabilistic clustering model [19], similar to a finite Gaussian mixture. However, the DPM is not consistent in estimating the number of clusters, it tends to overestimate the number of clusters (see simulation exercise below); although there is posterior asymptotic concentration in the other model components [285].

The hierarchical representation implies that there are latent assignment variables $s_i = h$, such that when $\boldsymbol{\theta}_i$ is equal to the h-th unique $\boldsymbol{\theta}_h^*$ — that is, $\boldsymbol{\theta}_i = \boldsymbol{\theta}_h^*$— then $s_i = h$. Then,

$$\begin{aligned} s_i &\sim \sum_{h=0}^{\infty} \lambda_h \delta_h, \\ y_i \mid s_i, \boldsymbol{\theta}_{s_i} &\sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}_{s_i}, \sigma_{s_i}^2), \end{aligned}$$

where $\lambda_h = P(\boldsymbol{\theta}_i = \boldsymbol{\theta}_h^*)$.

This latent assignment structure, the *Pólya urn* representation of the DP [45], which is obtained when G is marginalized out to avoid infinite atoms, and the use of conjugate priors allow for convenient computational inference in DPM.

Specifically, we assume $\boldsymbol{\beta}_i \mid \sigma_i^2 \sim N(\boldsymbol{\beta}_0, \sigma_i^2 \boldsymbol{B}_0)$ and $\sigma_i^2 \sim IG(\alpha_0/2, \delta_0/2)$. In addition, we can add a layer in the hierarchical representation, $\alpha \sim G(a, b)$ such that introducing the latent variable $\xi|\alpha, N \sim Be(\alpha+1, N)$, allows to easily sample the posterior draws of $\alpha|\xi, H, \pi_\xi \sim \pi_\xi G(a+H, b-log(\xi)) + (1-\pi_\xi)G(a+H-1, b-log(\xi))$, where $\frac{\pi_\xi}{1-\pi_\xi} = \frac{a+H-1}{N(b-log(\xi))}$, H is the number of atoms (mixture components) [127].

The conditional posterior distribution of $\boldsymbol{\theta}_i$ is

$$\boldsymbol{\theta}_i | \left\{\boldsymbol{\theta}_{i'}, \boldsymbol{s}_{i'} : i' \neq i\right\}, y_i, \alpha \sim \sum_{i' \neq i} \frac{N_h^{(i)}}{\alpha + N - 1} f_N(y_i | \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2)$$
$$+ \frac{\alpha}{\alpha + N - 1} \int_{\mathcal{R}^K} \int_0^\infty f_N(y_i | \boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) f_N\left(\boldsymbol{\beta} \middle| \boldsymbol{\beta}_0, \sigma^2 \boldsymbol{B}_0\right) f_{IG}(\sigma^2 | \alpha_0, \delta_0) d\sigma^2 d\boldsymbol{\beta},$$

where $N_h^{(i)}$ is the number of observations such that $s_{i'} = h$, $i' \neq i$.

Observe that the probability of belonging to a particular cluster has a reinforcement property, as it increases with the cluster size; therefore, a DPM exhibits a self-reinforcing property, the more often a given value has been sampled in the past, the more likely it is to be sampled again.

Observe that the integral in the previous equation has exactly the same form as in the marginal likelihood presented in Section 3.3. Thus,

$$p(y_i) = \int_{\mathcal{R}^K} \int_0^\infty f_N(y_i | \boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) f_N\left(\boldsymbol{\beta} \middle| \boldsymbol{\beta}_0, \sigma^2 \boldsymbol{B}_0\right) f_{IG}(\sigma^2 | \alpha_0, \delta_0) d\sigma^2 d\boldsymbol{\beta}$$
$$= \frac{1}{\pi^{1/2}} \frac{\delta_0^{\alpha_0/2}}{\delta_n^{\alpha_n/2}} \frac{|\boldsymbol{B}_n|^{1/2}}{|\boldsymbol{B}_0|^{1/2}} \frac{\Gamma(\alpha_n/2)}{\Gamma(\alpha_0/2)},$$

where $\alpha_n = 1 + \alpha_0$, $\delta_n = \delta_0 + y^\top y + \boldsymbol{\beta}_0^\top \boldsymbol{B}_0^{-1} \boldsymbol{\beta}_0 - \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1} \boldsymbol{\beta}_n$, $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{x}\boldsymbol{x}^\top)^{-1}$ and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{x}y)$.

Therefore, we sample s_i as follows,

$$s_i | \left\{\boldsymbol{\beta}_{i'}, \sigma_{i'}^2, \boldsymbol{s}_{i'} : i' \neq i\right\}, y_i, \alpha \sim \left\{\begin{array}{c} P(s_i = 0|\cdot) = q_0^* \\ P(s_i = h|\cdot) = q_h^*, h = 1, 2, \ldots, H^{(i)} \end{array}\right\},$$

where $H^{(i)}$ is the number of clusters excluding i, which may have its own cluster (singleton cluster), $q_c^* = \frac{q_c}{q_0 + \sum_h q_h}$, $q_c = \{q_0, q_h\}$, $q_h = \frac{N_h^{(i)}}{\alpha + N - 1} f_N(y_i | \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2)$ and $q_0 = \frac{\alpha}{\alpha + N - 1} p(y_i)$.

If $s_i = 0$ is sampled, then $s_i = H + 1$, and a new σ_h^2 is sampled from $IG\left(\alpha_n/2, \delta_n/2\right)$, a new $\boldsymbol{\beta}_h$ is sample from $N(\boldsymbol{\beta}_n, \sigma_h^2 \boldsymbol{B}_n)$.

Discarding $\boldsymbol{\theta}_h$'s from last step, we use $\boldsymbol{s}$ and the total number of components to sample σ_h^2 from

$$IG\left(\frac{\alpha_0 + N_m}{2}, \frac{\delta_0 + \boldsymbol{y}_h^\top \boldsymbol{y}_h + \boldsymbol{\beta}_0^\top \boldsymbol{B}_0^{-1} \boldsymbol{\beta}_0 - \boldsymbol{\beta}_{hn}^\top \boldsymbol{B}_{hn}^{-1} \boldsymbol{\beta}_{hn}}{2}\right),$$

where $\boldsymbol{B}_{hn} = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}_h^\top \boldsymbol{X}_h)^{-1}$ and $\boldsymbol{\beta}_{hn} = \boldsymbol{B}_{hn}(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}_h^\top \boldsymbol{y}_h)$, $\boldsymbol{X}_h$ and $\boldsymbol{y}_h$ have the $\boldsymbol{x}_i$ and y_i of individuals in component h. We sample $\boldsymbol{\beta}_h$ from

$$N\left(\boldsymbol{\beta}_{hn}, \sigma_h^2 \boldsymbol{B}_{hn}\right),$$

$h = 1, 2, \ldots, H$.

We can also use DPMs in a semi-parametric setting, as we did in the finite Gaussian mixtures case. In Exercise 3, we ask to get the posterior sampler in this setting, that is,

$$y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + e_i$$
$$e_i \mid \mu_i, \sigma_i^2 \overset{iid}{\sim} N(\mu_i, \sigma_i^2).$$

We should not include the intercept in $\boldsymbol{\beta}$, such that μ_i allows to get flexibility in the distribution of the stochastic errors.

Note that mixture models can be easily extended to non-linear models, where posterior inference is based on data augmentation, such as the probit and tobit models in Chapter 6. The basic idea is to incorporate the mixture (finite or infinite) into the specification of the latent variable implicit in the data-generating process. For instance, [26] perform inference in the probit model using DPMs. Furthermore, mixtures can also be used in the multivariate models presented in Chapter 7 and the hierarchical models presented in Chapter 9. [332] perform semi-parametric inference in the exact affine demand system [249] using DPMs, while [26] apply them in hierarchical models.

To sum up, a Dirichlet process mixture is a flexible way to model non-parametrically a distribution using an infinite weighted sum of discrete distributions, where each individual weight increases with the number of observations that belongs to it.

Example: Simulation exercises

Let's simulate again the simple regression mixture with two components such that $\psi_{i1} \sim \text{Ber}(0.5)$, consequently, $\psi_{i2} = 1 - \psi_{i1}$, and assume one regressor, $x_i \sim N(0, 1)$, $i = 1, 2, \ldots, 1,000$. Then,

$$p(y_i \mid \boldsymbol{x}_i) = 0.5\phi(y_i \mid 2 + 1.5x_i, 1^2) + 0.5\phi(y_i \mid -1 + 0.5x_i, 0.8^2).$$

We use non-informative priors again, setting $\alpha_0 = \delta_0 = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_2$, $\boldsymbol{B}_0 = \boldsymbol{I}_2$, and $a = b = 0.1$. The number of MCMC iterations is 5,000, the burn-in is 1,000, and the thinning parameter is 2. However, we do not have to fix in advance the number of clusters using the DPM.

The following code demonstrates how to perform inference using the stated Gibbs sampler for the DPM in the **R** package.

We see from the results that the DPM overestimates the number of clusters. After the burn-in period, there are typically three clusters. Additionally, the posterior plots illustrate the label-switching problem. Eventually, the posterior chains of clusters reach different posterior modes, causing a high variability of the posterior draws. In Exercise 5, we ask to fix this issue using *random permutation of latent classes*.

R code. Simulation exercise: Dirichlet process mixture using Gibbs sampling

```
rm(list = ls())
set.seed(010101)
# Simulate data from a 2-component mixture model
N <- 1000; x <- rnorm(N); z <- rbinom(N, 1, 0.5)
y <- ifelse(z == 0, rnorm(N, 2 + 1.5*x, 1), rnorm(N, -1 +
    0.5*x, 0.8))
X <- cbind(1, x)
k <- 2
data <- data.frame(y, x); Reg <- lm(y ~ x)
SumReg <- summary(Reg)
# Hyperparameters
a0 <- 0.001; d0 <- 0.001
b0 <- rep(0, k); B0 <- diag(k)
B0i <- solve(B0)
a <- 0.1; b <- 0.1
# MCMC parameters
mcmc <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
PostSig2 <- function(Xh, yh){
  Nh <- length(yh)
  yh <- matrix(yh, Nh, 1)
  if(Nh == 1){
    Xh <- matrix(Xh, k, 1)
    Bn <- solve(Xh%*%t(Xh) + B0i)
    bn <- Bn%*%(B0i%*%b0 + Xh%*%yh)
  }else{
    Xh <- matrix(Xh, Nh, k)
    Bn <- solve(t(Xh)%*%Xh + B0i)
    bn <- Bn%*%(B0i%*%b0 + t(Xh)%*%yh)
  }
  Bni <- solve(Bn)
  an <- a0 + Nh
  dn <- d0 + t(yh)%*%yh + t(b0)%*%B0i%*%b0 - t(bn)%*%Bni%*%
    bn
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2h, Xh, yh){
  Nh <- length(yh)
  yh <- matrix(yh, Nh, 1)
  if(Nh == 1){
    Xh <- matrix(Xh, k, 1)
    Bn <- solve(Xh%*%t(Xh) + B0i)
    bn <- Bn%*%(B0i%*%b0 + Xh%*%yh)
  }else{
    Xh <- matrix(Xh, Nh, k)
    Bn <- solve(t(Xh)%*%Xh + B0i)
    bn <- Bn%*%(B0i%*%b0 + t(Xh)%*%yh)
  }
  Beta <- MASS::mvrnorm(1, bn, sig2h*Bn)
  return(Beta)
}
```

R code. Simulation exercise: Dirichlet process mixture using Gibbs sampling

```
PostAlpha <- function(s, alpha){
  H <- length(unique(s))
  psi <- rbeta(1, alpha + 1, N)
  pi.ratio <- (a + H - 1) / (N * (b - log(psi)))
  pi <- pi.ratio / (1 + pi.ratio)
  components <- sample(1:2, prob = c(pi, (1 - pi)), size =
    1)
  cs <- c(a + H, a + H - 1)
  ds <- b - log(psi)
  alpha <- rgamma(1, cs[components], ds)
  return(alpha)
}
LogMarLikLM <- function(xh, yh){
  xh <- matrix(xh, k, 1)
  Bn <- solve(xh%*%t(xh) + B0i)
  Bni <- solve(Bn)
  bn <- Bn%*%(B0i%*%b0 + xh%*%yh)
  an <- a0 + 1
  dn <- d0 + yh^2 + t(b0)%*%B0i%*%b0 - t(bn)%*%Bni%*%bn
  # Log marginal likelihood
  logpy <- (1/2)*log(1/pi)+(a0/2)*log(d0)-(an/2)*log(dn) +
    0.5*log(det(Bn)/det(B0)) + lgamma(an/2)-lgamma(a0/2)
  return(logpy)
}
PostS <- function(BETA, SIGMA, Alpha, s, i){
  Nl <- table(s[-i]); H <- length(Nl)
  qh <- sapply(1:H, function(h){(Nl[h]/(N+Alpha-1))*dnorm(y[
    i], mean = t(X[i,])%*%BETA[,h], sd = SIGMA[h])})
  q0 <- (Alpha/(N+Alpha-1))*exp(LogMarLikLM(xh = X[i,], yh =
     y[i]))
  qh <- c(q0, qh)
  Clust <- as.numeric(names(Nl))
  si <- sample(c(0, Clust), 1, prob = qh)
    if(si == 0){
      si <- Clust[H] + 1
      Sig2New <- PostSig2(Xh = X[i,], yh = y[i])
      SIGMA <- c(SIGMA, Sig2New^0.5)
      BetaNew <- PostBeta(sig2h = Sig2New, Xh = X[i,], yh =
    y[i])
      BETA <- cbind(BETA, BetaNew)
    }else{si == si
    }
    return(list(si = si, BETA = BETA, SIGMA = SIGMA))
  }
PostBetas <- list(); PostSigma <- list()
Posts <- matrix(0, tot, N); PostAlphas <- rep(0, tot)
S <- sample(1:3, N, replace = T, prob = c(0.5, 0.3, 0.2))
BETA <- cbind(Reg$coefficients, Reg$coefficients, Reg$
    coefficients)
SIGMA <- rep(SumReg$sigma, 3)
Alpha <- rgamma(1, a, b)
```

R code. Simulation exercise: Dirichlet process mixture using Gibbs sampling

```
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  for(i in 1:N){
    Rests <- PostS(BETA = BETA, SIGMA = SIGMA, Alpha = Alpha
    , s = S, i = i)
    S[i] <- Rests$si
    BETA <- Rests$BETA; SIGMA <- Rests$SIGMA
  }
    sFreq <- table(S)
  lt <- 1
  for(li in as.numeric(names(sFreq))){
    Index <- which(S == li)
    if(li == lt){S[Index] <- li
    } else {S[Index] <- lt
    }
    lt <- lt + 1
  }
  Alpha <- PostAlpha(s = S, alpha = Alpha)
  Nl <- table(S); H <- length(Nl)
  SIGMA <- rep(NA, H)
  BETA <- matrix(NA, k, H)
  l <- 1
  for(h in unique(S)){
    Idh <- which(S == h)
    SIGMA[l] <- (PostSig2(Xh = X[Idh, ], yh = y[Idh]))^0.5
    BETA[,l] <- PostBeta(sig2h = SIGMA[l]^2, Xh = X[Idh, ],
    yh = y[Idh])
    l <- l + 1
  }
  PostBetas[[s]] <- BETA
  PostSigma[[s]] <- SIGMA
  Posts[s, ] <- S
  PostAlphas[s] <- Alpha
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorS<- Posts[keep,]
Clusters <- sapply(1:length(keep), function(i){length(table(
    PosteriorS[i,]))})
table(Clusters)
PosteriorBeta1 <- matrix(NA, length(keep), k); j <- 1
for(s in keep){
  PosteriorBeta1[j,] <- PostBetas[[s]][,1]; j <- j + 1
}
print(summary(coda::mcmc(PosteriorBeta1)))
PosteriorBeta2 <- matrix(NA, length(keep), k);j <- 1
for(s in keep){
  PosteriorBeta2[j,] <- PostBetas[[s]][,2]; j <- j + 1
}
print(summary(coda::mcmc(PosteriorBeta2)))
PosteriorBeta3 <- matrix(NA, length(keep), k); j <- 1
for(s in keep){
  PosteriorBeta3[j,] <- PostBetas[[s]][,3]; j <- j + 1
}
print(summary(coda::mcmc(PosteriorBeta3)))
```

Example: Consumption of marijuana in Colombia

Let's use the dataset *MarijuanaColombia.csv* from our GitHub repository to perform inference on the demand for marijuana in Colombia. This dataset contains information on the (log) monthly demand in 2019 from the National Survey of the Consumption of Psychoactive Substances. It includes variables such as the presence of a drug dealer in the neighborhood (*Dealer*), gender (*Female*), indicators of good physical and mental health (*PhysicalHealthGood* and *MentalHealthGood*), age (*Age* and *Age2*), years of schooling (*YearsEducation*), and (log) prices of marijuana, cocaine, and crack by individual (*LogPriceMarijuana*, *LogPriceCocaine*, and *LogPriceCrack*). The sample size is 1,156.

We are interested in the own-price and cross-price elasticities of marijuana demand. However, there is a potential endogeneity issue between price and demand due to strategic provider search (omission of a relevant regressor) or measurement errors in self-reported prices. Endogeneity should be properly addressed for rigorous scientific analysis. This example is purely illustrative.

In Exercise 2 we ask to perform inference using a finite Gaussian mixture regression starting with five potential clusters. Here we perform inference using a Dirichlet process mixture.

The following code shows how to perform this analysis using non-informative priors, setting $\alpha_0 = \delta_0 = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_K$, $\boldsymbol{B}_0 = \boldsymbol{I}_K$, and $a = b = 0.1$, K is the number of regressors, 11 including the intercept. The number of MCMC iterations is 5,000, the burn-in is 1,000, and the thinning parameter is 2.

R code. Demand of marijuana in Colombia: Dirichlet process mixture

```
rm(list = ls()); set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/MarijuanaColombia.csv")
attach(Data)
y <- LogMarijuana; X <- as.matrix(cbind(1, Data[,-1]))
Reg <- lm(y ~ X - 1); SumReg <- summary(Reg)
k <- dim(X)[2]
N <- dim(X)[1]
library(ggplot2)
ggplot(Data, aes(x = LogMarijuana)) + geom_density(fill = "blue", alpha = 0.3) + labs(title = "Density Plot: Marijuana (log) monthly consumption in Colombia", x = "y", y = "Density") + theme_minimal()
a0 <- 0.001; d0 <- 0.001
b0 <- rep(0, k); B0 <- diag(k)
B0i <- solve(B0)
a <- 0.1; b <- 0.1
# MCMC parameters
mcmc <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
PostSig2 <- function(Xh, yh){
  Nh <- length(yh)
  yh <- matrix(yh, Nh, 1)
  if(Nh == 1){
    Xh <- matrix(Xh, k, 1)
    Bn <- solve(Xh%*%t(Xh) + B0i)
    bn <- Bn%*%(B0i%*%b0 + Xh%*%yh)
  }else{
    Xh <- matrix(Xh, Nh, k)
    Bn <- solve(t(Xh)%*%Xh + B0i)
    bn <- Bn%*%(B0i%*%b0 + t(Xh)%*%yh)
  }
  Bni <- solve(Bn)
  an <- a0 + Nh
  dn <- d0 + t(yh)%*%yh + t(b0)%*%B0i%*%b0 - t(bn)%*%Bni%*%bn
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2h, Xh, yh){
  Nh <- length(yh)
  yh <- matrix(yh, Nh, 1)
  if(Nh == 1){
    Xh <- matrix(Xh, k, 1)
    Bn <- solve(Xh%*%t(Xh) + B0i)
    bn <- Bn%*%(B0i%*%b0 + Xh%*%yh)
  }else{
    Xh <- matrix(Xh, Nh, k)
    Bn <- solve(t(Xh)%*%Xh + B0i)
    bn <- Bn%*%(B0i%*%b0 + t(Xh)%*%yh)
  }
  Beta <- MASS::mvrnorm(1, bn, sig2h*Bn)
  return(Beta)
}
```

R code. Demand of marijuana in Colombia: Dirichlet process mixture

```
PostAlpha <- function(s, alpha){
  H <- length(unique(s))
  psi <- rbeta(1, alpha + 1, N)
  pi.ratio <- (a + H - 1) / (N * (b - log(psi)))
  pi <- pi.ratio / (1 + pi.ratio)
  components <- sample(1:2, prob = c(pi, (1 - pi)), size =
    1)
  cs <- c(a + H, a + H - 1)
  ds <- b - log(psi)
  alpha <- rgamma(1, cs[components], ds)
  return(alpha)
}
LogMarLikLM <- function(xh, yh){
  xh <- matrix(xh, k, 1)
  Bn <- solve(xh%*%t(xh) + B0i)
  Bni <- solve(Bn)
  bn <- Bn%*%(B0i%*%b0 + xh%*%yh)
  an <- a0 + 1
  dn <- d0 + yh^2 + t(b0)%*%B0i%*%b0 - t(bn)%*%Bni%*%bn
  logpy <- (1/2)*log(1/pi)+(a0/2)*log(d0)-(an/2)*log(dn) +
    0.5*log(det(Bn)/det(B0)) + lgamma(an/2)-lgamma(a0/2)
  return(logpy)
}
PostS <- function(BETA, SIGMA, Alpha, s, i){
  Nl <- table(s[-i]); H <- length(Nl)
  qh <- sapply(1:H, function(h){(Nl[h]/(N+Alpha-1))*dnorm(y[
    i], mean = t(X[i,])%*%BETA[,h], sd = SIGMA[h])})
  q0 <- (Alpha/(N+Alpha-1))*exp(LogMarLikLM(xh = X[i,], yh =
     y[i]))
  qh <- c(q0, qh)
  Clust <- as.numeric(names(Nl))
  si <- sample(c(0, Clust), 1, prob = qh)
  if(si == 0){
    si <- Clust[H] + 1
    Sig2New <- PostSig2(Xh = X[i,], yh = y[i])
    SIGMA <- c(SIGMA, Sig2New^0.5)
    BetaNew <- PostBeta(sig2h = Sig2New, Xh = X[i,], yh = y[
    i])
    BETA <- cbind(BETA, BetaNew)
  }else{si == si
  }
  return(list(si = si, BETA = BETA, SIGMA = SIGMA))
}
PostBetas <- list(); PostSigma <- list()
Posts <- matrix(0, tot, N); PostAlphas <- rep(0, tot)
S <- sample(1:3, N, replace = T, prob = c(0.5, 0.3, 0.2))
BETA <- cbind(Reg$coefficients, Reg$coefficients, Reg$
    coefficients)
SIGMA <- rep(SumReg$sigma, 3)
Alpha <- rgamma(1, a, b)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
```

R code. Demand of marijuana in Colombia: Dirichlet process mixture

```
for(s in 1:tot){
  for(i in 1:N){
    Rests <- PostS(BETA = BETA, SIGMA = SIGMA, Alpha = Alpha, s = S, i = i)
    S[i] <- Rests$si
    BETA <- Rests$BETA; SIGMA <- Rests$SIGMA
  }
  sFreq <- table(S)
  lt <- 1
  for(li in as.numeric(names(sFreq))){
    Index <- which(S == li)
    if(li == lt){S[Index] <- li
    } else {S[Index] <- lt
    }
    lt <- lt + 1
  }
  Alpha <- PostAlpha(s = S, alpha = Alpha)
  Nl <- table(S); H <- length(Nl)
  SIGMA <- rep(NA, H)
  BETA <- matrix(NA, k, H)
  l <- 1
  for(h in unique(S)){
    Idh <- which(S == h)
    SIGMA[l] <- (PostSig2(Xh = X[Idh, ], yh = y[Idh]))^0.5
    BETA[,l] <- PostBeta(sig2h = SIGMA[l]^2, Xh = X[Idh, ], yh = y[Idh])
    l <- l + 1
  }
  PostBetas[[s]] <- BETA
  PostSigma[[s]] <- SIGMA
  Posts[s, ] <- S
  PostAlphas[s] <- Alpha
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0), "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorS<- Posts[keep,]
Clusters <- sapply(1:length(keep), function(i){length(table(PosteriorS[i,]))})
table(Clusters)
NClus <- 3
sapply(1:length(keep), function(i){print(table(PosteriorS[i,]))})
PosteriorSIGMA <- matrix(NA, length(keep), NClus)
l <- 1
for (s in keep){
  PosteriorSIGMA[l,] <- PostSigma[[s]][1:NClus]
  l <- l + 1
}
summary(coda::mcmc(PosteriorSIGMA))
plot(coda::mcmc(PosteriorSIGMA))
```

R code. Demand of marijuana in Colombia: Dirichlet process mixture

```
l <- 1
for (s in keep){
  PosteriorSIGMA[l,] <- PostSigma[[s]][1:NClus]
  l <- l + 1
}
summary(coda::mcmc(PosteriorSIGMA))
plot(coda::mcmc(PosteriorSIGMA))
PosteriorBeta1 <- matrix(NA, length(keep), k)
j <- 1
for(s in keep){
  PosteriorBeta1[j,] <- PostBetas[[s]][,1]
  j <- j + 1
}
colnames(PosteriorBeta1) <- c("Ct", names(Data[,-1]))
HDI <- HDInterval::hdi(PosteriorBeta1, credMass = 0.95)
print(summary(coda::mcmc(PosteriorBeta1)))
plot(coda::mcmc(PosteriorBeta1))
PosteriorBeta2 <- matrix(NA, length(keep), k)
j <- 1
for(s in keep){
  PosteriorBeta2[j,] <- PostBetas[[s]][,2]
  j <- j + 1
}
colnames(PosteriorBeta2) <- c("Ct", names(Data[,-1]))
HDI <- HDInterval::hdi(PosteriorBeta2, credMass = 0.95)
print(summary(coda::mcmc(PosteriorBeta2)))
plot(coda::mcmc(PosteriorBeta2))
PosteriorBeta3 <- matrix(NA, length(keep), k)
j <- 1
for(s in keep){
  PosteriorBeta3[j,] <- PostBetas[[s]][,3]
  j <- j + 1
}
colnames(PosteriorBeta3) <- c("Ct", names(Data[,-1]))
HDI <- HDInterval::hdi(PosteriorBeta3, credMass = 0.95)
print(summary(coda::mcmc(PosteriorBeta3)))
plot(coda::mcmc(PosteriorBeta3))
DataElast <- data.frame(Marijuana = PosteriorBeta1[,2],
    Cocaine = PosteriorBeta1[,3],
Crack = PosteriorBeta1[,4])
dens1 <- ggplot(DataElast, aes(x = Marijuana)) + geom_
    density(fill = "blue", alpha = 0.3) + labs(x = "
    Marijuana", y = "Density") + theme_minimal()
dens2 <- ggplot(DataElast, aes(x = Cocaine)) + geom_density(
    fill = "blue", alpha = 0.3) +  labs(x = "Cocaine", y = "
    Density") + theme_minimal()
dens3 <- ggplot(DataElast, aes(x = Crack)) + geom_density(
    fill = "blue", alpha = 0.3) +
labs(x = "Crack", y = "Density") + theme_minimal()
library(ggpubr)
ggarrange(dens1, dens2, dens3, labels = c("A", "B", "C"),
    ncol = 3, nrow = 1, legend = "bottom", common.legend =
    TRUE)
```

In this application, there are three clusters after the burn-in period, with individuals in the sample relatively evenly allocated between them. The posterior density plots of the elasticities associated with the three clusters appear very similar, which implies that the posterior draws associated with each cluster are targeting the same posterior distribution. Consequently, Figure 11.4 presents the posterior density estimates of the demand price elasticities of marijuana in Colombia for the first cluster.

Panel A of Figure 11.4 shows that the own-price elasticity of marijuana demand is bimodal. The first mode suggests that some individuals have an elasticity near -0.5, meaning that a 10% increase in the price of marijuana leads to a 5% decrease in their demand. The second mode is near 0, indicating that these individuals do not adjust their marijuana consumption in response to price changes.

Panel B presents the cross-price elasticity of marijuana demand with respect to the price of cocaine. This posterior distribution is also bimodal. The first mode suggests that some individuals do not change their marijuana demand in response to cocaine price variations. The second mode indicates that marijuana and cocaine are substitutes: a 10% increase in the price of cocaine leads to an approximately 2.5% increase in marijuana demand.

Panel C shows that marijuana and crack are also substitutes. This posterior distribution is unimodal, indicating that most individuals increase their marijuana consumption when the price of crack rises. On average, a 10% increase in the price of crack results in an approximately 3% increase in marijuana demand.

These results seem plausible. However, they should be interpreted with caution, as potential endogeneity issues may affect the estimates.

11.2 Splines

Another common approach to performing non-parametric or semi-parametric inference is using splines. These are piecewise polynomial functions that allow for approximating complex relationships in data.

To clarify ideas, let's begin with a simple univariate case. The starting point is to assume that

$$y_i = f(x_i) + \mu_i,$$

where $\mu_i \sim N(0, \sigma^2)$ is independent of the regressor x_i, and $f(x_i)$ is a smooth function such that nearby points in the support of x_i should have similar values. Observe that this means we must always sort the data in increasing order according to x_i. This has no effect, as we assume a random sample. The index $i = 1, 2, \ldots, N$ represents the observations.

FIGURE 11.4
Posterior distributions: Elasticities of marijuana consumption in Colombia.

The spline approach defines

$$f(x_i) = \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_H b_H(x_i) = \boldsymbol{b}(x_i)^\top \boldsymbol{\beta}, \tag{11.1}$$

where $\boldsymbol{\beta} = [\beta_1 \ \beta_2 \ \ldots \ \beta_H]^\top$ and $\boldsymbol{b}(x_i) = [b_1(x_i) \ b_2(x_i) \ \ldots \ b_H(x_i)]^\top$ are the vector of parameters and the vector of spline basis (B-splines) function values at x_i, respectively.

Note that in this setting, there is only one regressor; however, there are H location parameters. Given N observations, the design matrix is:

$$\boldsymbol{W} = \begin{bmatrix} b_1(x_1) & b_2(x_1) & \ldots & b_H(x_1) \\ b_1(x_2) & b_2(x_2) & \ldots & b_H(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_N) & b_2(x_N) & \ldots & b_H(x_N) \end{bmatrix}.$$

Thus, we have a standard linear model where we can proceed as in Chapter 3.

Actually, Equation 11.1 represents the basis function approach, where $\boldsymbol{b}(x_i)$ is a specified set of basis functions, such as B-splines. However, other functions derived from the Fourier transform, Gaussian radial basis functions, wavelets, etc., can also be used. Here, we focus on the most popular case: cubic B-splines. However, the same ideas apply to other basis functions. In particular, cubic B-splines provide the lowest degree necessary to generate sufficiently smooth curves; the first and second derivatives are nonzero, which is not the case with constant and linear splines [276].

Thus, from Equation 11.1, we see that a spline is a linear combination of H B-splines. The latter are polynomials of a certain degree, meaning that splines are piecewise polynomial functions. Any spline function of order n can be expressed as a linear combination of B-splines of order n, and B-splines serve as the basis functions for the spline function space. The cubic B-spline is

$$b_{h,3}(x_i) = \left\{ \begin{array}{ccc} \frac{u^3}{6} & x_h \leq x_i < x_{h+1}, & u = (x_i - x_h)/\delta \\ \frac{1+3u+3u^2-3u^3}{6} & x_{h+1} \leq x_i < x_{h+2}, & u = (x_i - x_{h+1})/\delta \\ \frac{4-6u^2+3u^3}{6} & x_{h+2} \leq x_i < x_{h+3}, & u = (x_i - x_{h+2})/\delta \\ \frac{1-3u+3u^2-3u^3}{6} & x_{h+3} \leq x_i < x_{h+4}, & u = (x_i - x_{h+3})/\delta \\ 0 & \text{otherwise} & \end{array} \right\},$$

where x_{h+k} are called the knots, for $k = 0, 1, \ldots, 4$. These are the points in the domain of the function where the pieces of the spline join, satisfying $x_{h+k} \leq x_{h+k+1}$. The knots control the shape and flexibility of the spline curve; more knots imply greater flexibility but less smoothness. Note that δ determines the distance between the knots, and that this expression clearly shows that the cubic B-spline is a piecewise polynomial function.

The following code implements this function in **R** using a delta of 1.5 and 5 knots $\{2, 3.5, 5, 6.5, 8\}$. The knots $\{2, 8\}$ are called boundary knots, which can be given by the range of x, and the knots $\{3.5, 5, 6.5\}$ are called internal knots. B-splines with evenly distributed knots are called uniform B-splines. However, this is not always the case.

Figure 11.5 compares the results of our own function (black circles) with those obtained using the *bs* function from the *splines* package (red line).

R code. B-spline

```
SplineOwn <- function(x, knots, delta){
  if(knots[1] <= x & x < knots[2]){
    u <- (x - knots[1])/delta
    b <- u^3/6
  }else{
    if(knots[2] <= x & x < knots[3]){
      u <- (x - knots[2])/delta
      b <- (1/6)*(1 + 3*u + 3*u^2 - 3*u^3)
    }else{
      if(knots[3] <= x & x < knots[4]){
        u <- (x - knots[3])/delta
        b <- (1/6)*(4 - 6*u^2 + 3*u^3)
      }else{
        if(knots[4] <= x & x < knots[5]){
          u <- (x - knots[4])/delta
          b <- (1/6)*(1 - 3*u + 3*u^2 - u^3)
        }else{
          b <- 0
        }
      }
    }
  }
  return(b)
}
delta <- 1.5
knotsA <- seq(2, 8, delta)
xA <- seq(2, 8, 0.1)
Ens <- sapply(xA, function(xi) {SplineOwn(xi, knots = knotsA
    , delta = delta)})
plot(xA, Ens, xlab = "x", ylab = "B-spline", main = "Cubic B
    -spline comparison: own function vs bs")
require(splines)
BSfunc <- bs(xA, knots = knotsA, degree = 3)
lines(xA, BSfunc[,4], col = "red")
```

Figure 11.5 shows a particular cubic B-spline, but $f(x_i)$ in Equation 11.1 involves H splines over the range of potential values of x, as we need to evaluate the splines centered at different knots. Here, we should clarify that to avoid issues with the use of splines at boundary regions, it is essential to artificially increase the number of boundary knots by repeating these knots as many times as the degree of the polynomial. For instance, in our example, the final set of knots is $\{2, 2, 2, 2, 3.5, 5, 6.5, 8, 8, 8, 8\}$.

The set of B-splines can be constructed using the Cox–de Boor recursion

FIGURE 11.5
Cubic B-spline comparison.

formula. The starting point is the B-spline of degree 0,

$$b_{h,0}(x_i) = \begin{Bmatrix} 1 & x_h \leq x_i < x_{h+1}, \quad u = (x_i - x_h)/\delta \\ 0 & \text{otherwise} \end{Bmatrix},$$

and then, the other B-splines are defined by the recursion

$$b_{h,p}(x_i) = \frac{x_i - x_h}{x_{h+p} - x_h} b_{h,p-1}(x_i) + \frac{x_{h+p+1} - x_i}{x_{h+p+1} - x_{h+1}} b_{h+1,p-1}(x_i). \tag{11.2}$$

Note that B-splines are defined by their degree and their knots.

The following code demonstrates how to perform this recursion using the same settings as in the previous code and compares the results with the *bs* function from the *splines* package. As shown in Figure 11.6, we obtain the same results. We will continue using the *bs* function from the *splines* package for convenience in the process, but it seems that we have gained some understanding of the underlying process.

R code. Splines

```
cubic_bspline <- function(x, knots, degree = 3) {
  extended_knots <- c(rep(knots[1], degree), knots, rep(knots[length(knots)], degree))
  num_basis <- length(knots) + degree - 1
  basis_matrix <- matrix(0, nrow = length(x), ncol = num_basis)
  # Function to compute B-spline basis recursively
  b_spline_basis <- function(x, degree, i, knots) {
    if (degree == 0) {
      return(ifelse(x >= knots[i] & x < knots[i + 1], 1, 0))
    } else {
      left_num <- (x - knots[i])
      left_den <- (knots[i + degree] - knots[i])
      left <- ifelse(left_den != 0, (left_num / left_den) * b_spline_basis(x, degree - 1, i, knots), 0)

      right_num <- (knots[i + degree + 1] - x)
      right_den <- (knots[i + degree + 1] - knots[i + 1])
      right <- ifelse(right_den != 0, (right_num / right_den) * b_spline_basis(x, degree - 1, i + 1, knots), 0)

      return(left + right)
    }
  }

  for (i in 1:num_basis) {
    basis_matrix[, i] <- sapply(x, function(xi) b_spline_basis(xi, degree, i, extended_knots))
  }
  if(x[length(x)] == knots[length(knots)]){
    basis_matrix[length(x), num_basis] <- 1
  }
  return(basis_matrix)
}
delta <- 1.5
knotsA <- seq(2, 8, delta)
xA <- seq(2, 8, 0.1)
basis_matrix <- cubic_bspline(xA, knots = knotsA, degree = 3)
library(splines)
bs_matrix <- bs(xA, knots = knotsA[-c(1, length(knotsA))], degree = 3, intercept = TRUE, Boundary.knots = range(knotsA))
bs_matrix_matrix <- as.matrix(bs_matrix)
par(mfrow = c(1,2))
matplot(xA, basis_matrix, type = "l", lty = 1, col = rainbow(ncol(basis_matrix)), ylab = "B-spline Basis", xlab = "x", main = "Own function")
matplot(xA, bs_matrix_matrix, type = "l", lty = 1, col = rainbow(ncol(basis_matrix)), ylab = "B-spline Basis", xlab = "x", main = "bs function")
```

FIGURE 11.6
Sequence of cubic B-splines comparison.

The idea in splines is to calculate the linear combination of the H B-splines, as those shown in Figure 11.6, that bets fit the dependent variable. Let's perform a simulation exercise to fix ideas.

Splines can also be extended to non-linear models based on data augmentation, such as the probit and tobit models. Again, the basic idea is to incorporate splines into the implicit latent variables. [234, Chap. 10] presents these extensions.

Simulation exercise: Splines

Let's assume that the data generating process is

$$y_i = 0.4 + 0.25\sin(8x_i - 5) + 0.4\exp(-16(4x_i - 2.5)^2) + \mu_i,$$

where $\mu_i \sim N(0, 0.15^2)$, and x_i is a sequence in $(0, 1)$, with 100 random draws of the pairs (x_i, y_i). In addition, we choose the knots $\{0, 0.25, 0.5, 0.75, 1\}$. We then calculate the cubic B-splines; however, we should take into account that the last two B-spline columns generate multicollinearity issues. Therefore, we exclude them when generating random realizations of the splines by drawing the coefficients from $N(0, 0.35^2)$. The intercept is fixed to maintain the scale in Figure 11.7, where we observe the data points, $f(x)$, and the different splines associated with various random realizations of the coefficients. The following code demonstrates how to perform this simulation.

R code. Realizations of splines

```
########### Simulation: B-splines ###########
rm(list = ls())
library(ggplot2); library(splines)
set.seed(010101)
x <- seq(0, 1, 0.001)
ysignal <- 0.4 + 0.25*sin(8*x - 5) + 0.4*exp(-16*(4*x - 2.5)^2)
sig <- 0.15
e <- rnorm(length(ysignal), 0, sd = sig)
y <- ysignal + e
N <- 100
ids <- sort(sample(1:length(ysignal), N))
xobs <- x[ids]
yobs <- y[ids]
knots <- seq(0, 1, 0.25)
BS <- bs(xobs, knots = knots, degree = 3, Boundary.knots = range(x), intercept = FALSE)
# Splines
Spline1 <- 0.56 + BS[,-c(7:8)] %*% rnorm(6, 0, 0.35)
Spline2 <- 0.56 + BS[,-c(7:8)] %*% rnorm(6, 0, 0.35)
Spline3 <- 0.56 + BS[,-c(7:8)] %*% rnorm(6, 0, 0.35)
Spline4 <- 0.56 + BS[,-c(7:8)] %*% rnorm(6, 0, 0.35)
# Create data frames for the true signal, observed data, and Splines
data_true_signal <- data.frame(x = x, y = ysignal, Type = "True Signal")
data_obs <- data.frame(x = xobs, y = yobs, Type = "Observed Data")
# Create separate data frames for each Spline
data_Spline1 <- data.frame(x = xobs, y = Spline1, Type = "Spline 1")
data_Spline2 <- data.frame(x = xobs, y = Spline2, Type = "Spline 2")
data_Spline3 <- data.frame(x = xobs, y = Spline3, Type = "Spline 3")
data_Spline4 <- data.frame(x = xobs, y = Spline4, Type = "Spline 4")
data <- rbind(data_true_signal, data_obs, data_Spline1, data_Spline2, data_Spline3, data_Spline4)
ggplot(data, aes(x = x, y = y)) + geom_line(data = subset(data, Type == "True Signal"), aes(color = "True Signal"), linewidth = 1) + geom_point(data = subset(data, Type == "Observed Data"), aes(color = "Observed Data"), shape = 16) + geom_line(data = subset(data, Type == "Spline 1"), aes(color = "Splines"), linewidth = 1, linetype = "solid") + geom_line(data = subset(data, Type == "Spline 2"), aes(color = "Splines"), linewidth = 1, linetype = "solid") + geom_line(data = subset(data, Type == "Spline 3"), aes(color = "Splines"), linewidth = 1, linetype = "solid") + geom_line(data = subset(data, Type == "Spline 4"), aes(color = "Splines"), linewidth = 1, linetype = "solid") + scale_color_manual(values = c("True Signal" = "black", "Observed Data" = "red", "Splines" = "blue")) + labs(y = "y", color = "Legend") + theme_minimal() +
theme(legend.position = "top")
```

FIGURE 11.7
Different realizations of splines.

We can also use this simulation to examine the effect of changing the knots. The following code performs the fit using least squares, which is equivalent to using a non-informative prior in a Bayesian linear regression framework.

Figure 11.8 illustrates the fit of different splines based on varying numbers of knots. We observe that increasing the number of knots enhances the flexibility of the spline, providing better local control. However, too many knots can result in a wiggly, overly complex fit that captures noise rather than the true underlying trend and may also increase multicollinearity. Therefore, selecting an appropriate number of knots is a crucial aspect of spline modeling. To address this issue, we can apply the strategies discussed in Chapter 10, and the strategies that will be discussed in Section 12.2 in the next chapter.

R code. Demand of marijuana in Colombia: Splines in age

```r
rm(list = ls())
library(ggplot2); library(splines)
# Data generation
set.seed(10101)
x <- seq(0, 1, 0.001)
ysignal <- 0.4 + 0.25*sin(8*x - 5) + 0.4*exp(-16*(4*x - 2.5)^2)
sig <- 0.15; e <- rnorm(length(ysignal), 0, sd = sig)
y <- ysignal + e; N <- 100
ids <- sort(sample(1:length(ysignal), N))
xobs <- x[ids]
yobs <- y[ids]
# Generate Fits with different knot placements
knots_list <- list(seq(0, 1, 0.33), seq(0, 1, 0.25), seq(0, 1, 0.2), seq(0, 1, 0.1))
Fits <- list()
for (i in 1:4) {
  BS <- bs(xobs, knots = knots_list[[i]], degree = 3, Boundary.knots = range(x), intercept = FALSE)
  fm <- lm(yobs ~ BS)
  Fits[[i]] <- predict(fm)
}
# Create data frames
data_true_signal <- data.frame(x = x, y = ysignal, Type = "True Signal")
data_obs <- data.frame(x = xobs, y = yobs, Type = "Observed Data")
data_preds <- data.frame(
x = rep(xobs, 4),
y = c(Fits[[1]], Fits[[2]], Fits[[3]], Fits[[4]]),
Type = rep(c("Fit 1", "Fit 2", "Fit 3", "Fit 4"), each = length(xobs))
)
data <- rbind(data_true_signal, data_obs, data_preds)

# Create ggplot
ggplot(data, aes(x = x, y = y, color = Type)) + geom_line(data = subset(data, Type == "True Signal"), linewidth = 1) + geom_point(data = subset(data, Type == "Observed Data"), shape = 16, size = 2) + geom_line(data = subset(data, grepl("Fit", Type)), linewidth = 1, linetype = "solid") + scale_color_manual(values = c("True Signal" = "black", "Observed Data" = "red", "Fit 1" = "blue", "Fit 2" = "green", "Fit 3" = "orange", "Fit 4" = "purple")) + labs(y = "y", color = "Legend") +
theme_minimal() + theme(legend.position = "top")
```

FIGURE 11.8
Different fits of splines.

Note that splines use local information to perform the fit. This implies the *curse of dimensionality* issue, where increasing the number of variables leads to more dispersed regions in the regressor space. In other words, the observations become increasingly further apart as we have more regressors. As a result, splines become less reliable in high-dimensional spaces.

A strategy to overcome the curse of dimensionality is to specify the *partial linear model*:

$$y_i = \boldsymbol{z}_i^\top \boldsymbol{\gamma} + f(x_i) + \mu_i. \tag{11.3}$$

In this specification, some regressors enter in a linear way, while potentially the most relevant regressor — the one under primary investigation — enters in a non-parametric way. Note that we only need to increase the dimension of the matrix $\boldsymbol{W}$ by adding the columns from $\boldsymbol{z}$, and proceed as we did previously.

Example: Consumption of marijuana in Colombia continues

Let's continue using the dataset *MarijuanaColombia.csv* to perform a partial linear model as in Equation 11.3, where y_i is the (log) marijuana monthly consumption, $\boldsymbol{z}_i$ represents the presence of a drug dealer in the neighborhood (*Dealer*), gender (*Female*), indicators of good physical and mental health (*PhysicalHealthGood* and *MentalHealthGood*), years of education (*YearsEducation*), and the (log) prices of marijuana, cocaine, and crack by individual.

We set the knots as the percentiles $\{0, 0.05, \ldots, 0.95, 1\}$ of age, use cubic B-splines, non-informative conjugate priors in the linear regression model, 5,000 MCMC iterations, and 5,000 burn-in iterations.

R code. Fit using splines: Different knots

```r
rm(list = ls()); set.seed(010101)
library(splines); library(ggplot2)
Data <- read.csv("https://raw.githubusercontent.com/BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/MarijuanaColombia.csv")
attach(Data)
IdOrd <- order(Age)
y <- LogMarijuana[IdOrd]
Z <- as.matrix(cbind(Data[IdOrd,-c(1, 6, 7)]))
x <- Age[IdOrd]
knots <- quantile(x, seq(0, 1, 0.05))
BS <- bs(x, knots = knots, degree = 3, Boundary.knots = range(x), intercept = FALSE)
matplot(x, BS, type = "l", lty = 1, col = rainbow(ncol(BS)))
X <- cbind(1, BS[,1:22], Z) # Get B-splines without multicollinearity issues
k <- dim(X)[2]; N <- dim(X)[1]
# Hyperparameters
d0 <- 0.001; a0 <- 0.001
b0 <- rep(0, k); c0 <- 1000; B0 <- c0*diag(k); B0i <- solve(B0)
# MCMC parameters
mcmc <- 5000; burnin <- 5000
tot <- mcmc + burnin; thin <- 1
posterior  <- MCMCpack::MCMCregress(y~X-1, b0=b0, B0 = B0i, c0 = a0, d0 = d0, burnin = burnin, mcmc = mcmc, thin = thin)
summary(coda::mcmc(posterior))
# Predict values with 95% credible intervals
xfit <- seq(min(x), max(x), 0.2)
H <- length(xfit)
i <- sample(1:N, 1)
idfit <- sample(1:N, H)
BSfit <- bs(xfit, knots = knots, degree = 3, Boundary.knots = range(x), intercept = FALSE)
Xfit <- cbind(1, BSfit[,1:22], Z[rep(i, H),]) # Relevant regressors, PIP > 0.5
Fit <- matrix(NA, mcmc, H)
# posterior[posterior > 0] <- 0
for(s in 1:mcmc){
  Fit[s,] <- Xfit%*%posterior[s,1:31]
}
# Create a data frame for ggplot
plot_data <- data.frame(x = xfit, fit = colMeans(Fit), liminf = apply(Fit, 2, quantile, 0.025), limsup = apply(Fit, 2, quantile, 0.975))
ggplot() + geom_line(data = plot_data, aes(x, fit), color = "blue", linewidth = 1) + geom_ribbon(data = plot_data, aes(x, ymin = liminf, ymax = limsup), fill = "blue", alpha = 0.2) + labs(title = "B-Spline Regression with 95% Confidence Interval", x = "Age", y = "Log Marijuana") + theme_minimal()
```

FIGURE 11.9
Spline: Marijuana monthly consumption vs age.

The previous code illustrates the procedure. The posterior 95% credible intervals indicate that marijuana is an inelastic good, there is substitution between marijuana and crack, more educated female individuals consume less, and the presence of a drug dealer in the neighborhood increases consumption.

Figure 11.9 shows the fitted spline, specifically the mean (blue line) and the 95% credible intervals (shaded blue). Notice that there is a lot of wiggliness due to the use of many knots, suggesting that this relationship could be expressed in a more parsimonious way. In Exercise 7, we ask to use variable selection methods, such as the BIC approximation presented in Chapter 10, to perform regressor selection, particularly for the splines.

In general, the selection of H is a problem of variable selection (regressor uncertainty) and can be handled using other approaches, such as those discussed in the regularization section 12.2 of the following chapter.

11.3 Summary

In this chapter, we introduce the most common Bayesian methods for inference in non-parametric and semi-parametric models. Finite Gaussian and Dirichlet process mixtures, as well as splines, are highly flexible methods; however, they also have limitations, such as the label-switching issue in mixtures and potential multicollinearity in splines. Additionally, it is wise to elicit informative

priors and conduct sensitivity analyses to assess the robustness of the results in real-world applications.

11.4 Exercises

1. Simulate a semi-parametric regression where

$$y_i = 0.5x_{i1} - 1.2x_{i2} + \mu_i,$$
$$p(\mu_i) = 0.3\phi(\mu_i \mid -0.5, 0.5^2) + 0.7\phi(\mu_i \mid 1, 0.8^2).$$

   Assume that x_{i1} and x_{i2} follow a standard normal distribution and that the sample size is 1,000. Perform inference in this model assuming that the number of components is unknown. Start with $H = 5$ and use non-informative priors, setting $\alpha_{h0} = \delta_{h0} = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_2$, $\boldsymbol{B}_0 = \boldsymbol{I}_2$, $\mu_{h0} = 0$, $\sigma^2_{\mu 0} = 10$, and $\boldsymbol{\alpha}_0 = [1/H \ \ldots \ 1/H]^\top$. Use 6,000 MCMC iterations, a burn-in period of 4,000, and a thinning parameter of 2. Compare the population parameters with the posterior estimates and plot the population density along with the posterior density estimate of $\boldsymbol{\mu}$ (the mean, and the 95% credible interval).

2. **Example: Consumption of marijuana in Colombia continues I**

   Use the dataset *MarijuanaColombia.csv* from our GitHub repository to perform inference on the demand for marijuana in Colombia. This dataset contains information on the (log) monthly demand in 2019 from the National Survey of the Consumption of Psychoactive Substances. It includes variables such as the presence of a drug dealer in the neighborhood (*Dealer*), gender (*Female*), indicators of good physical and mental health (*PhysicalHealthGood* and *MentalHealthGood*), age (*Age* and *Age2*), years of schooling (*YearsEducation*), and (log) prices of marijuana, cocaine, and crack by individual (*LogPriceMarijuana*, *LogPriceCocaine*, and *LogPriceCrack*). The sample size is 1,156.

   Estimate a finite Gaussian mixture regression using non-informative priors, that is, $\alpha_0 = \delta_0 = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_K$, $\boldsymbol{B}_0 = \boldsymbol{I}_K$, and and $\boldsymbol{\alpha}_0 = [1/H \ \ldots \ 1/H]^\top$, K is the number of regressors, 11 including the intercept. The number of MCMC iterations is 5,000, the burn-in is 1,000, and the thinning parameter is 2. Start with five potential clusters. Obtain the posterior distribution of the own-price elasticity of marijuana and the cross-price elasticities of marijuana demand with respect to the prices of cocaine and crack.

3. Get the posterior sampler in the semi-parametric setting using a Dirichlet process mixture:

$$y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + e_i$$
$$e_i \mid \mu_i, \sigma_i^2 \overset{iid}{\sim} N(\mu_i, \sigma_i^2),$$

Do not include the intercept in $\boldsymbol{\beta}$ to get flexibility in the distribution of the stochastic errors.

Let's assume $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$, $\sigma_i^2 \sim IG(\alpha_0/2, \delta_0/2)$, $\mu_i \sim N(\mu_0, \sigma_i^2/\beta_0)$, $\alpha \sim G(a, b)$ such that introducing the latent variable $\xi|\alpha, N \sim Be(\alpha+1, N)$, allows to easily sample the posterior draws of $\alpha|\xi, H, \pi_\xi \sim \pi_\xi G(a+H, b-log(\xi)) + (1-\pi_\xi)G(a+H-1, b-log(\xi))$, where $\frac{\pi_\xi}{1-\pi_\xi} = \frac{a+H-1}{N(b-log(\xi))}$, H is the number of atoms (mixture components).

4. **Example: Exercise 1 and 3 continue**

Perform inference in the simulation of the semi-parametric model of Exercise 1 using the sampler of Exercise 3. Use non-informative priors, setting $\alpha_0 = \delta_0 = 0.01$, $\boldsymbol{\beta}_0 = \boldsymbol{0}_2$, $\boldsymbol{B}_0 = \boldsymbol{I}_2$, and $a = b = 0.1$. The number of MCMC iterations is 5,000, the burn-in is 1,000, and the thinning parameter is 2.

5. **Example: Simulation exercise continues I**

Fix the label-switching problem of the simulation exercise of the DPM using *random permutation of latent classes*.

6. **Example: Simulation exercise continues II**

Obtain the density estimate of y in the simulation exercise of the DPM, evaluated at $x = 0$.

7. **Example: Consumption of marijuana in Colombia continues II**

Perform the application of marijuana consumption with the following specification:

$$y_i = \boldsymbol{z}_i^\top \boldsymbol{\gamma} + f(Age_i) + \mu_i,$$

where y_i is the (log) marijuana monthly consumption, $\boldsymbol{z}_i$ represents the presence of a drug dealer in the neighborhood (*Dealer*), gender (*Female*), indicators of good physical and mental health (*PhysicalHealthGood* and *MentalHealthGood*), years of education (*YearsEducation*), and the (log) prices of marijuana, cocaine, and crack by individual.

Initially, set the knots as the percentiles $\{0, 0.05, \ldots, 0.95, 1\}$ of age and use cubic B-splines. Then, apply the BIC approximation to perform variable selection in this model with non-informative conjugate priors, 5,000 MCMC iterations, and 5,000 burn-in iterations.

Do you think that using a linear regression with a second-degree polynomial in age provides a good approximation to the relationship found using splines in this application?

12

Bayesian machine learning

In this chapter, we focus on Bayesian approaches to *supervised Machine Learning* (ML) problems, where the outcome variable is observed and used to guide prediction or inference. In contrast, *unsupervised ML* refers to settings in which the outcome variable is not observed, such as in clustering or dimensionality reduction.

Machine learning methods are often characterized by high-dimensional parameter spaces, particularly in the context of nonparametric inference. It is important to note that nonparametric inference does not imply the absence of parameters, but rather models with potentially infinitely many parameters. This setting is often referred to as the *wide* problem, where the number of input variables, and consequently parameters, can exceed the sample size. Another common challenge in ML is the *tall* problem, which occurs when the sample size is extremely large, necessitating scalable algorithms.

Specifically, we introduce Bayesian ML tools for regression, including regularization techniques, regression trees, and Gaussian processes. Extensions of these methods for binary classification are explored in some of the exercises. The section begins with a discussion on the relationship between cross-validation and Bayes factors, and concludes with Bayesian approaches for addressing large-scale data challenges.

12.1 Cross-validation and Bayes factors

Prediction is central to machine learning, particularly in supervised learning. The starting point is a set of raw regressors or features, denoted by $\mathbf{x}$, which are used to construct a set of input variables fed into the model: $\mathbf{w} = T(\mathbf{x})$, where $T(\cdot)$ represents a *dictionary of transformations* such as polynomials, interactions between variables, or the application of functions like exponentials, and so on. These inputs are then used to predict y through a model $f(\mathbf{w})$:

$$y_i = f(\mathbf{w}_i) + \mu_i,$$

where $\mu_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$.

We predict y using $\hat{f}(\mathbf{w})$, a model trained on the data. The expected squared error (ESE) at a fixed input $\mathbf{w}_i$ is given by:

$$
\begin{aligned}
\text{ESE} &= \mathbb{E}_{\mathcal{D},y}\left[(y_i - \hat{f}(\mathbf{w}_i))^2\right] \\
&= \mathbb{E}_{\mathcal{D},\boldsymbol{\mu}}\left[\left(f(\mathbf{w}_i) + \mu_i - \hat{f}(\mathbf{w}_i)\right)^2\right] \\
&= \mathbb{E}_{\mathcal{D},\boldsymbol{\mu}}\left[(f(\mathbf{w}_i) - \hat{f}(\mathbf{w}_i))^2 + 2\mu_i(f(\mathbf{w}_i) - \hat{f}(\mathbf{w}_i)) + \mu_i^2\right] \\
&= \mathbb{E}_{\mathcal{D}}\left[(f(\mathbf{w}_i) - \hat{f}(\mathbf{w}_i))^2\right] + \mathbb{E}_{\boldsymbol{\mu}}\left[\mu_i^2\right] \\
&= \mathbb{E}_{\mathcal{D}}\left[\left((f(\mathbf{w}_i) - \bar{f}(\mathbf{w}_i)) - (\hat{f}(\mathbf{w}_i) - \bar{f}(\mathbf{w}_i))\right)^2\right] + \sigma^2 \\
&= \underbrace{\mathbb{E}_{\mathcal{D}}\left[(f(\mathbf{w}_i) - \bar{f}(\mathbf{w}_i))^2\right]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[(\hat{f}(\mathbf{w}_i) - \bar{f}(\mathbf{w}_i))^2\right]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible Error}} .
\end{aligned}
$$

Here, $\mathcal{D}$ denotes the distribution over datasets defined by the feature space. Independence between the noise $\mu_i \sim \mathcal{N}(0, \sigma^2)$ and the estimator ensures that $\mathbb{E}_{\mathcal{D},\boldsymbol{\mu}}\left[\mu_i(f(\mathbf{w}_i) - \hat{f}(\mathbf{w}_i))\right] = 0$. We also define $\bar{f}(\mathbf{w}_i) = \mathbb{E}_{\mathcal{D}}[\hat{f}(\mathbf{w}_i)]$ as the expected predictor across datasets.

Thus, the ESE is composed of the squared bias, the variance of the prediction (which is a random variable due to data variation $\mathcal{D}$), and the irreducible error. The key insight is that increasing model complexity, such as by including more inputs, typically reduces bias but increases variance. This trade-off can lead to overfitting, where models perform well on the training data but poorly on new, unseen data. There is, therefore, an optimal point at which the predictive error is minimized (see Figure 12.1).[1]

To avoid overfitting in machine learning, an important step called *cross-validation* is often employed. This involves splitting the dataset into multiple parts (called *folds*) and systematically training and testing models on these parts [183, 124]. The main goal is to evaluate how well models generalize to "unseen data".

There is a compelling justification for cross-validation within Bayesian inference proposed by [38] in their section 6.1.6. The point of departure is to assume an $\mathcal{M}$-open view of nature, in which there exists a set of models $\mathcal{M} = \{M_j : j \in J\}$ under comparison, but none of them represents the true data-generating process, which is assumed to be unknown. Nevertheless, we can compare the models in $\mathcal{M}$ based on their posterior risk functions (see Chapter 1) without requiring the specification of a true model. In this framework, we select the model that minimizes the posterior expected loss. Unfortunately, we cannot explicitly compute this posterior expected loss due

[1] However, recent developments show that powerful modern machine learning methods, such as deep neural networks, often overfit and yet generalize remarkably well on unseen data. This phenomenon is known as *double descent* or *benign overfitting* [32, 25, 181].

Notes: Curves for training squared error (dashed line) and test squared error (solid line). The classical U-shaped error curve arising from the bias–variance trade-off.

FIGURE 12.1
Bias-variance trade-off in machine learning.

to the lack of knowledge of the true posterior distribution under the $\mathcal{M}$-open assumption.[2]

Given the expected loss conditional on model j for a predictive problem, where the action a is chosen to minimize the posterior expected loss:

$$\mathbb{E}_{y_0}[L(Y_0, a) \mid M_j, \mathbf{y}] = \int_{\mathcal{Y}_0} L(y_0, a \mid M_j, \mathbf{y})\, \pi(y_0 \mid \mathbf{y})\, dy_0,$$

we note that there are N possible partitions of the dataset $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$ following a leave-one-out strategy, i.e., $\mathbf{y} = \{\mathbf{y}_{-k}, y_k\}$, for $k = 1, \ldots, N$, where $\mathbf{y}_{-k}$ denotes the dataset excluding observation y_k. Assuming that $\mathbf{y}$ is exchangeable, meaning its joint distribution is invariant to reordering, and that N is large, then $\mathbf{y}_{-k}$ and y_k are good proxies for $\mathbf{y}$ and y_0, respectively. Consequently,

$$\frac{1}{K} \sum_{k=1}^{K} L(y_k, a \mid M_j, \mathbf{y}_{-k}) \xrightarrow{p} \int_{\mathcal{Y}_0} L(y_0, a \mid M_j, \mathbf{y})\, \pi(y_0 \mid \mathbf{y})\, dy_0,$$

by the law of large numbers, as $N \to \infty$ and $K \to \infty$.

Thus, we can select a model by minimizing the expected squared error

[2]This is not the case under an $\mathcal{M}$-closed view of nature, where one of the candidate models is assumed to be the true data-generating process. In that setting, the posterior distribution becomes a mixture distribution with mixing probabilities given by the posterior model probabilities (see Chapter 10).

based on its expected predictions $\mathbb{E}[y_k \mid M_j, \mathbf{y}_{-k}]$; that is, we select the model with the lowest value of

$$\frac{1}{K}\sum_{k=1}^{K}\left(\mathbb{E}[y_k \mid M_j, \mathbf{y}_{-k}] - y_k\right)^2.$$

Note that if we want to compare two models based on their relative predictive accuracy using the log-score function [275], we select model j if

$$\int_{\mathcal{Y}_0} \log \frac{p(y_0 \mid M_j, \mathbf{y})}{p(y_0 \mid M_l, \mathbf{y})}\, \pi(y_0 \mid \mathbf{y})\, dy_0 > 0.$$

However, we know that

$$\frac{1}{K}\sum_{k=1}^{K} \log \frac{p(y_k \mid M_j, \mathbf{y}_{-k})}{p(y_k \mid M_l, \mathbf{y}_{-k})} \xrightarrow{p} \int_{\mathcal{Y}_0} \log \frac{p(y_0 \mid M_j, \mathbf{y})}{p(y_0 \mid M_l, \mathbf{y})}\, \pi(y_0 \mid \mathbf{y})\, dy_0,$$

by the law of large numbers as $N \to \infty$ and $K \to \infty$.

This implies that we select model j over model l if

$$\prod_{k=1}^{K}\left(\frac{p(y_k \mid M_j, \mathbf{y}_{-k})}{p(y_k \mid M_l, \mathbf{y}_{-k})}\right)^{1/K} = \prod_{k=1}^{K}\left(B_{jl}(y_k, \mathbf{y}_{-k})\right)^{1/K} > 1,$$

where $B_{jl}(y_k, \mathbf{y}_{-k})$ is the Bayes factor comparing model j to model l, based on the posterior $\pi(\boldsymbol{\theta}_m \mid M_m, \mathbf{y}_{-k})$ and the predictive $\pi(y_k \mid \boldsymbol{\theta}_m)$, for $m \in \{j, l\}$.

Thus, under the log-score function, cross-validation reduces to the geometric average of Bayes factors that evaluate predictive performance based on the leave-one-out samples $\mathbf{y}_{-k}$.

12.2 Regularization

In this century, the amount of available data continues to grow. This means we have access to more covariates for prediction, and we can also generate additional inputs to enhance the predictive power of our models. As a result, we often encounter *wide* datasets, where the number of inputs may exceed the number of observations. Even in modest settings, we might have hundreds of inputs, and we can use them to identify which ones contribute to accurate predictions. However, we generally avoid using all inputs at once due to the risk of overfitting. Thus, we require a class of input selection or *regularization*.

In the standard linear regression setting,

$$\mathbf{y} = \mathbf{1}_N \beta_0 + \mathbf{W}\boldsymbol{\beta} + \boldsymbol{\mu},$$

where $\mathbf{1}_N$ is an N-dimensional vector of ones, $\mathbf{W}$ is the $N \times K$ design matrix of inputs, and $\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$, there has been extensive development of techniques aimed at regularization within the Frequentist inferential framework. These include methods such as Ridge regression [193]; discrete subset selection techniques like best subset selection [142], forward selection, and backward stepwise selection [183]; as well as continuous subset selection approaches such as the LASSO [397], the elastic net [433], and OSCAR [50].

It is important to note, however, that Ridge regression does not perform variable selection; rather, it shrinks coefficient estimates toward zero without setting them exactly to zero.

Ridge regression and the LASSO can be viewed as special cases of a more general class of estimators known as *Bridge regression* [141], which also admits a Bayesian interpretation. Consider the following penalized least squares criterion in the linear regression setting:

$$\hat{\boldsymbol{\beta}}^{\text{Bridge}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{k=1}^{K} \tilde{w}_{ik} \beta_k \right)^2 + \lambda \sum_{k=1}^{K} |\beta_k|^q \right\}, \quad q \geq 0,$$

where $\tilde{w}_{ik}$ denotes the standardized inputs. Standardizing inputs is important in variable selection problems to avoid issues caused by differences in scale; otherwise, variables with larger magnitudes will be penalized less and disproportionately influence the regularization path.

Interpreting $|\beta_k|^q$ as proportional to the negative log-prior density of β_k, the penalty shapes the contours of the prior distribution on the parameters. Specifically:

- $q = 0$ corresponds to best subset selection, where the penalty counts the number of nonzero coefficients.
- $q = 1$ yields the LASSO, which corresponds to a Laplace (double-exponential) prior.
- $q = 2$ yields ridge regression, which corresponds to a Gaussian prior.

In this light, best subset selection, the LASSO, and ridge regression can be viewed as maximum a posteriori (MAP) estimators under different priors centered at zero [304]. However, they are not Bayes estimators in the strict sense, since Bayes estimators are typically defined as the posterior *mean*. While ridge regression coincides with the posterior mean under a Gaussian prior [205], the LASSO and best subset selection yield posterior modes.

This distinction is important because the Bayesian framework naturally incorporates regularization through the use of proper priors, which helps mitigate overfitting. Specifically, when proper shrinkage priors are used, the posterior balances data likelihood and prior information, thus controlling model complexity.

Furthermore, empirical Bayes methods, where the marginal likelihood is

optimized, or cross-validation can be used to estimate the scale parameter of the prior covariance matrix for the regression coefficients. This scale parameter, in turn, determines the strength of regularization in ridge regression.

Note that regularization introduces bias into the parameter estimates because it constrains the model, shrinking the location parameters toward zero. However, it substantially reduces variance, as the estimates are prevented from varying excessively across samples. As a result, the mean squared error (MSE) of the estimates, which equals the sum of the squared bias and the variance, is often lower for regularization methods compared to ordinary least squares (OLS), which remains unbiased under the classical assumptions. This trade-off is particularly important when the goal is to identify causal effects, where unbiasedness may be preferred over predictive accuracy (see Chapter 13).

12.2.1 Bayesian LASSO

Given the popularity of the LASSO as a variable selection technique, we present its Bayesian formulation in this subsection [304]. The Gibbs sampler for the Bayesian LASSO exploits the representation of the Laplace distribution as a scale mixture of normals. This leads to the following hierarchical representation of the model:

$$\begin{aligned}
\mathbf{y} \mid \beta_0, \boldsymbol{\beta}, \sigma^2, \mathbf{W} &\sim N(\mathbf{1}_N \beta_0 + \mathbf{W}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_N), \\
\boldsymbol{\beta} \mid \sigma^2, \tau_1^2, \ldots, \tau_K^2 &\sim N(\mathbf{0}_K, \sigma^2 \mathbf{D}_\tau), \\
\tau_1^2, \ldots, \tau_K^2 &\sim \prod_{k=1}^{K} \frac{\lambda^2}{2} \exp\left\{-\frac{\lambda^2}{2}\tau_k^2\right\}, \\
\sigma^2 &\sim \frac{1}{\sigma^2}, \\
\beta_0 &\sim c,
\end{aligned}$$

where $\mathbf{D}_\tau = \mathrm{diag}(\tau_1^2, \ldots, \tau_K^2)$ and c is a constant.

After integrating out $\tau_1^2, \ldots, \tau_K^2$, the conditional prior of $\boldsymbol{\beta} \mid \sigma^2$ is:

$$\pi(\boldsymbol{\beta} \mid \sigma^2) = \prod_{k=1}^{K} \frac{\lambda}{2\sqrt{\sigma^2}} \exp\left\{-\frac{\lambda}{\sqrt{\sigma^2}}|\beta_k|\right\},$$

which implies that the log-prior is proportional to $\lambda \sum_{k=1}^{K} |\beta_k|$, matching the penalty term in the LASSO optimization problem.

The conditional posterior distributions for the Gibbs sampler are [304]:

$$\begin{aligned}
\boldsymbol{\beta} \mid \sigma^2, \tau_1^2, \ldots, \tau_K^2, \tilde{\mathbf{W}}, \tilde{\mathbf{y}} &\sim N(\boldsymbol{\beta}_n, \sigma^2 \mathbf{B}_n), \\
\sigma^2 \mid \boldsymbol{\beta}, \tau_1^2, \ldots, \tau_K^2, \tilde{\mathbf{W}}, \tilde{\mathbf{y}} &\sim \text{Inverse-Gamma}(\alpha_n/2, \delta_n/2), \\
1/\tau_k^2 \mid \boldsymbol{\beta}, \sigma^2 &\sim \text{Inverse-Gaussian}(\mu_{kn}, \lambda_n), \\
\beta_0 \mid \sigma^2, \tilde{\mathbf{W}}, \tilde{\mathbf{y}} &\sim N(\bar{y}, \sigma^2/N)
\end{aligned}$$

where:

$$
\begin{aligned}
\boldsymbol{\beta}_n &= \mathbf{B}_n \tilde{\mathbf{W}}^\top \tilde{\mathbf{y}}, \\
\mathbf{B}_n &= \left(\tilde{\mathbf{W}}^\top \tilde{\mathbf{W}} + \mathbf{D}_\tau^{-1}\right)^{-1}, \\
\alpha_n &= (N-1) + K, \\
\delta_n &= (\tilde{\mathbf{y}} - \tilde{\mathbf{W}}\boldsymbol{\beta})^\top (\tilde{\mathbf{y}} - \tilde{\mathbf{W}}\boldsymbol{\beta}) + \boldsymbol{\beta}^\top \mathbf{D}_\tau^{-1} \boldsymbol{\beta}, \\
\mu_{kn} &= \sqrt{\frac{\lambda^2 \sigma^2}{\beta_k^2}}, \\
\lambda_n &= \lambda^2,
\end{aligned}
$$

where $\tilde{\mathbf{W}}$ is the matrix of standardized inputs and $\tilde{\mathbf{y}}$ is the centered response vector.

Note that the posterior distribution of $\boldsymbol{\tau}$ depends on the data through $\boldsymbol{\beta}$ and σ^2, which is a typical feature of hierarchical models. In this formulation, we can interpret τ_k as local shrinkage parameters, while λ acts as a global shrinkage parameter. Higher values of τ_k and λ imply stronger shrinkage toward zero. [304] propose two approaches for specifying the global shrinkage parameter: empirical Bayes estimation or a fully Bayesian hierarchical specification, where λ^2 is assigned a Gamma prior.

We should acknowledge that the Bayesian LASSO is more computationally expensive than the Frequentist LASSO. However, it provides credible intervals for the parameters automatically. In contrast, obtaining standard errors in the Frequentist LASSO is more challenging, particularly for parameters estimated to be exactly zero [239].

Example: Simulation exercise to study the Bayesian LASSO performance

We simulate the process $y_i = \beta_0 + \sum_{k=1}^{10} \beta_k w_{ik} + \mu_i$, where $\beta_k \sim \mathcal{U}(-3, 3)$, $\mu_i \sim \mathcal{N}(0, 1)$, and $w_{ik} \sim \mathcal{N}(0, 1)$, $i = 1, 2, \ldots, 500$. Additionally, we generate 90 extra potential inputs from a standard normal distribution, which are included in the model specification. Our goal is to assess whether the Bayesian LASSO can successfully identify the truly relevant inputs.

We use the *bayesreg* package in **R** to perform the Bayesian LASSO, using 5,000 MCMC draws and 1,000 burn-in iterations. The following code illustrates the simulation exercise and compares the posterior means with the true population values.

The summary of the fit and the plot comparing the population parameters with the posterior means show that the Bayesian LASSO is able to identify the variables that are relevant in the data generating process. In Exercise 1, we propose programming the Gibbs sampler from scratch, assuming a hierarchical structure for the global shrinkage parameter, and comparing the results with those obtained using the *monomvn* package.

R code. The Bayesian LASSO

```
rm(list = ls()); set.seed(10101)
library(bayesreg)
# Parameters
n <- 500  # sample size
p <- 100  # number of predictors
s <- 10   # number of non-zero coefficients
# Generate design matrix
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
# True beta: first s coefficients are non-zero, rest are zero
beta_true <- c(runif(s, -3, 3), rep(0, p - s))
# Generate response with some noise
sigma <- 1
y <- X %*% beta_true + rnorm(n, sd = sigma)
df <- data.frame(X,y)
### Using bayesreg ###
# Fit the model
fit <- bayesreg::bayesreg(y ~ X, data = df, model = "gaussian", prior = "lasso",
n.samples = 5000, burnin = 1000)
# Check summary
summary(fit)
# Extract posterior means of beta
beta_post_mean <- rowMeans(fit$beta)
# Compare true vs estimated
plot(beta_true, beta_post_mean, pch = 19, col = "steelblue",
xlab = "True beta", ylab = "Posterior mean of beta",
main = "Bayesian LASSO Shrinkage")
abline(0, 1, col = "red", lty = 2)
```

12.2.2 Stochastic search variable selection

Another well-known Bayesian strategy for variable selection in the presence of a large set of regressors (inputs) is *stochastic search variable selection* (SSVS) [158, 157]. SSVS is a particular case of the broader class of *spike-and-slab* priors, in which the prior distribution for the location parameters is specified as a hierarchical mixture that captures the uncertainty inherent in variable

selection problems [205]. The hierarchical structure of the model is given by

$$\begin{aligned}
\mathbf{y} \mid \beta_0, \boldsymbol{\beta}, \sigma^2, \mathbf{W} &\sim N(\mathbf{1}_N \beta_0 + \mathbf{W}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_N), \\
\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{\gamma} &\sim N(\mathbf{0}_K, \sigma^2 \mathbf{D}_\gamma \mathbf{R} \mathbf{D}_\gamma), \\
\sigma^2 &\sim \text{Inverse-Gamma}\left(\frac{v}{2}, \frac{v\lambda_\gamma}{2}\right), \\
\gamma_k &\sim \text{Bernoulli}(p_k),
\end{aligned}$$

where p_k is the prior inclusion probability of regressor w_k, that is, $P(\gamma_k = 1) = 1 - P(\gamma_k = 0) = p_k$, $\mathbf{R}$ is a correlation matrix, and $\mathbf{D}_\gamma$ is a diagonal matrix whose kk-th element is defined as

$$(\mathbf{D}_\gamma)_{kk} = \begin{Bmatrix} v_{0k}, & \text{if } \gamma_k = 0, \\ v_{1k}, & \text{if } \gamma_k = 1 \end{Bmatrix}.$$

This formulation implies that $\beta_k \sim (1-\gamma_k)N(0, v_{0k}) + \gamma_k N(0, v_{1k})$, where v_{0k} and v_{1k} are chosen such that v_{0k} is small and v_{1k} is large. Therefore, when the data favors $\gamma_k = 0$ over $\gamma_k = 1$, the corresponding β_k is shrunk toward zero, effectively excluding input k from the model. In this sense, $N(0, v_{0k})$ is a spike prior concentrated at zero, while $N(0, v_{1k})$ is a diffuse slab prior. A critical aspect of SSVS is the choice of the hyperparameters v_{0k} and v_{1k}, as they determine the amount of shrinkage applied to the regression coefficients (see [158, 157] for details).

The assumption $\gamma_k \sim \text{Bernoulli}(p_k)$ implies that the prior on the inclusion indicators is given by $\pi(\boldsymbol{\gamma}) = \prod_{k=1}^{K} p_k^{\gamma_k}(1-p_k)^{1-\gamma_k}$. This means that the inclusion of input k is independent of the inclusion of any other input $j \neq k$. A common choice is the uniform prior $\pi(\boldsymbol{\gamma}) = 2^{-K}$, which corresponds to setting $p_k = 1/2$, giving each regressor an equal chance of being included [205].

A practical choice for the correlation matrix is to set $\mathbf{R} \propto (\tilde{\mathbf{W}}^\top \tilde{\mathbf{W}})^{-1}$ [158]. Regarding the hyperparameters v and λ_γ, it is helpful to interpret λ_γ as a prior estimate of σ^2, and v as the prior sample size associated with this estimate. In the absence of prior information, [159] recommend setting λ_γ equal to the least squares estimate of the variance from the *saturated model*, that is, the model including all regressors, and v a small number, for instance, 0.01.

The conditional posterior distributions for the Gibbs sampler are [158]:

$$\begin{aligned}
\boldsymbol{\beta} \mid \sigma^2, \gamma_1, \ldots, \gamma_K, \tilde{\mathbf{W}}, \tilde{\mathbf{y}} &\sim N(\boldsymbol{\beta}_n, \mathbf{B}_n), \\
\sigma^2 \mid \boldsymbol{\beta}, \gamma_1, \ldots, \gamma_K, \tilde{\mathbf{W}}, \tilde{\mathbf{y}} &\sim \text{Inverse-Gamma}(\alpha_n/2, \delta_n/2), \\
\gamma_k \mid \boldsymbol{\beta}, \sigma^2 &\sim \text{Bernoulli}(p_{kn}),
\end{aligned}$$

where:

$$
\begin{aligned}
\boldsymbol{\beta}_n &= \sigma^{-2}\mathbf{B}_n\tilde{\mathbf{W}}^\top\tilde{\mathbf{y}}, \\
\mathbf{B}_n &= \left(\sigma^{-2}\tilde{\mathbf{W}}^\top\tilde{\mathbf{W}} + \mathbf{D}_\gamma^{-1}\mathbf{R}^{-1}\mathbf{D}_\gamma^{-1}\right)^{-1}, \\
\alpha_n &= N + v, \\
\delta_n &= (\tilde{\mathbf{y}} - \tilde{\mathbf{W}}\boldsymbol{\beta})^\top(\tilde{\mathbf{y}} - \tilde{\mathbf{W}}\boldsymbol{\beta}) + v\lambda_\gamma, \\
p_{kn} &= \frac{\pi(\boldsymbol{\beta} \mid \boldsymbol{\gamma}_{-k}, \gamma_k = 1) \times p_k}{\pi(\boldsymbol{\beta} \mid \boldsymbol{\gamma}_{-k}, \gamma_k = 1) \times p_k + \pi(\boldsymbol{\beta} \mid \boldsymbol{\gamma}_{-k}, \gamma_k = 0) \times (1 - p_k)},
\end{aligned}
$$

where $\tilde{\mathbf{W}}$ is the matrix of standardized inputs, $\tilde{\mathbf{y}}$ is the centered response vector, $\boldsymbol{\gamma}_{-k}$ denotes the vector composed of $\gamma_1, \gamma_2, \ldots, \gamma_K$ excluding γ_k, and $\pi(\boldsymbol{\beta} \mid \boldsymbol{\gamma}_{-k}, \gamma_k = \delta)$ is the posterior density of $\boldsymbol{\beta}$ evaluated at $\boldsymbol{\gamma}_{-k}$ and $\gamma_k = \delta$, where $\delta \in \{0, 1\}$.

In general, it is wise to consider the inclusion of regressors jointly due to potential correlations among them; that is, the marginal frequency of $\gamma_k = 1$ should be interpreted with caution. SSVS is more effective at identifying a good set of potential models rather than selecting a single best model.

Example: Simulation exercise to study SSVS performance

Let's use the simulation setting from the previous example to evaluate the performance of SSVS in uncovering the data-generating process. In particular, we use the *BoomSpikeSlab* package to implement this example.

The analysis is performed using 5,000 posterior draws and the default prior. However, the package allows the user to modify the default prior via the *SpikeSlabPrior* function.

The results show that the posterior inclusion probabilities for regressors 2 through 10 are 100%, and the model with the highest posterior probability (94%) includes all of these nine variables. However, the true data-generating process, which also includes regressor 1, receives a posterior model probability of 0%. This is because the population coefficient of this regressor is essentially zero. The plot comparing the posterior means with the true population parameters indicates good performance of SSVS. In general, Bayesian methods for variable selection perform well, and the choice of the most suitable method largely depends on the prior specification [301].

R code. Stochastic search variable selection

```
rm(list = ls()); set.seed(10101)
library(BoomSpikeSlab)
library(dplyr)
library(tibble)
# Parameters
n <- 500  # sample size
k <- 100  # number of predictors
s <- 10   # number of non-zero coefficients
# Generate design matrix
X <- matrix(rnorm(n * k), nrow = n, ncol = k)
# True beta: first s coefficients are non-zero, rest are zero
beta_true <- c(runif(s, -3, 3), rep(0, k - s))
# Generate response with some noise
sigma <- 1
y <- X %*% beta_true + rnorm(n, sd = sigma)
df <- data.frame(X,y)
### Using BoomSpikeSlab ###
#Scale regressors
W <- scale(X); yh <- y - mean(y)
prior <- SpikeSlabPrior(W, yh,
expected.model.size = ncol(W)/2, # expect 50 nonzero predictors
prior.df = .01, # weaker prior than the default
prior.information.weight = .01,
diagonal.shrinkage = 0) # shrink to zero
niter <- 5000
######Estimate model########
SSBoomNew <- lm.spike(yh ~ W - 1, niter = niter, prior = prior)
Models <- SSBoomNew$beta != 0
PIP <- colMeans(SSBoomNew$beta != 0)
# Convert the logical matrix to a data frame and then to a tibble
df <- as.data.frame(Models); df_tbl <- as_tibble(df)
# Count identical rows
row_counts <- df_tbl %>% count(across(everything()), name = "frequency") %>% arrange(desc(frequency))
sum(row_counts[1:100,101])
# Ensure your vector and matrix are logical
trueModel <- c(rep(1, 10), rep(0, 90)) == 1  # convert to logical if needed
# Assume your matrix is named 'mat'
matching_rows <- apply(row_counts[,-101], 1, function(row) all(row == trueModel))
# Get indices (row numbers) where the match is TRUE
row_counts[which(matching_rows), 101]
# Coefficients
SummarySS <- summary(coda::mcmc(SSBoomNew$beta))
# Extract posterior means of beta
beta_post_mean <- SummarySS$statistics[, 1]
# Compare true vs estimated
plot(beta_true, beta_post_mean, pch = 19, col = "steelblue",
xlab = "True beta", ylab = "Posterior mean of beta",
main = "SSVS Shrinkage")
abline(0, 1, col = "red", lty = 2)
```

The examples and exercises presented thus far have considered scenarios in which the number of inputs is smaller than the number of observations ($K < N$). In Exercise 4, we challenge the Bayesian LASSO and SSVS in a setting where the number of inputs exceeds the sample size ($K > N$). As you will observe in that experiment, both the Bayesian LASSO and SSVS perform well. However, the Bayesian LASSO requires more time to produce results compared to SSVS in this exercise. [351] propose a connection between the LASSO and spike-and-slab priors for variable selection in linear models, offering oracle properties and optimal posterior concentration even in high-dimensional settings where $K > N$.

In addition, there are other Bayesian methods for regularization, such as the spike-and-slab approach proposed by [205] and non-local priors introduced by [215], which can be implemented using the **R** packages *spikeslab* and *mombf*, respectively.

12.3 Bayesian additive regression trees

A Classification and Regression Tree (CART) is a method used to predict outcomes based on inputs without assuming a parametric model. It is referred to as a classification tree when the outcome variable is qualitative, and as a regression tree when the outcome variable is quantitative. The method works by recursively partitioning the dataset into smaller, more homogeneous subsets using decision rules based on the input variables. For regression tasks, CART selects splits that minimize prediction error, typically measured by the sum of squared residuals. For classification problems, it aims to create the purest possible groups, using criteria such as Gini impurity or entropy. The result is a tree-like structure in which each internal node represents a decision based on one variable, and each leaf node corresponds to a prediction. CART was popularized in the statistical community by [57].

Figure 12.2 displays a binary regression tree with two variables: one continuous (x_1) and one categorical ($x_2 \in A, B, C$). The tree has seven nodes in total, four of which are terminal nodes ($B = 4$), dividing the input space $\mathbf{x} = (x_1, x_2)$ into four non-overlapping regions. Each internal node indicates the splitting variable and rule, while each terminal node shows the value θ_b, representing the conditional mean of y given $\mathbf{x}$.

The first split is based on the rule $x_1 \leq 5$ (left) versus $x_1 > 5$ (right). The leftmost terminal node corresponds to $x_2 \in A$ with $\mu_1 = 2$. The second terminal node, with $\mu_2 = 3$, is defined by $x_1 \leq 3$ and $x_2 \in B, C$. The third node assigns $\mu_3 = 5$ for $3 < x_1 \leq 5$ and $x_2 \in B, C$. Finally, the rightmost terminal node assigns $\mu_4 = 8$ for all observations with $x_1 > 5$.

We can view a CART model as specifying the conditional distribution of y given the vector of features $\mathbf{x} = [x_1, x_2, \ldots, x_K]^\top$. There are two main

Notes: Decision tree with one continuous variable (x_1) and one categorical variable ($x_2 \in \{A, B, C\}$). The tree has four terminal nodes, each associated with a value denoted by μ_b.

FIGURE 12.2
Decision tree.

components: (i) the binary tree structure T, which consists of a sequence of binary decision rules of the form $x_k \in A$ versus $x_k \notin A$, where A is a subset of the range of x_k, and B terminal nodes that define a non-overlapping and exhaustive partition of the input space; and (ii) the parameter vector $\boldsymbol{\theta} = [\mu_1, \mu_2, \ldots, \mu_B]^\top$ and σ^2, where each μ_b corresponds to the parameter associated with terminal node b, and σ^2 is the variance. Consequently, the response $y \mid \mathbf{x} \sim p(y \mid \mu_b, \sigma^2)$ if $\mathbf{x}$ belongs to the region associated with terminal node b, where p denotes a parametric distribution indexed by μ_b and σ^2.

Assuming that y_{bi} is independently and identically distributed within each terminal node and independently across nodes, for $b = 1, 2, \ldots, B$ and $i = 1, 2, \ldots, n_b$, we have:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}, T, \boldsymbol{\theta}, \sigma^2) &= \prod_{b=1}^{B} p(\mathbf{y}_b \mid \mu_b, \sigma^2) \\ &= \prod_{b=1}^{B} \prod_{i=1}^{n_b} p(y_{bi} \mid \mu_b, \sigma^2), \end{aligned}$$

where $\mathbf{y}_b = [y_{b1}, y_{b2}, \ldots, y_{bn_b}]^\top$ is the set of observations in the terminal node b.

[91] introduced a Bayesian formulation of CART models, in which inference is carried out through a combination of prior specification on the binary tree structure T and the parameters $\boldsymbol{\theta}, \sigma^2 \mid T$, along with a stochastic search strategy based on a Metropolis-Hastings algorithm. The transition kernels used in the algorithm include operations such as growing, pruning, changing, and swapping tree branches, and candidate trees are evaluated based on their marginal likelihood. This approach enables exploration of a richer set of potential tree models and offers a more flexible and effective alternative to the greedy algorithms commonly used in classical CART.

While CART is a simple yet powerful tool, it is prone to overfitting. To mitigate this, ensemble methods such as boosting, bagging, and random forests are often used. *Boosting* combines multiple weak trees sequentially, each correcting the errors of its predecessor, to create a strong predictive model [136]. *Bagging* builds multiple models on bootstrapped datasets and averages their predictions to reduce variance [55], and *random forests* extend bagging by using decision trees and adding random input selection at each split to increase model diversity [56]. Although a single tree might overfit and generalize poorly, aggregating many randomized trees typically yields more accurate and stable predictions.

[92] introduced Bayesian Additive Regression Trees (BART). The starting point is the model

$$y_i = f(\mathbf{x}_i) + \mu_i,$$

where $\mu_i \sim N(0, \sigma^2)$.

Thus, the conditional expectation $\mathbb{E}[y_i \mid \mathbf{x}_i] = f(\mathbf{x}_i)$ is approximated as

$$\begin{aligned} f(\mathbf{x}_i) &\approx h(\mathbf{x}_i) \\ &= \sum_{j=1}^{J} g_j(\mathbf{x}_i \mid T_j, \boldsymbol{\theta}_j), \end{aligned}$$

that is, $f(\mathbf{x}_i)$ is approximated by the sum of J regression trees. Each tree is defined by a structure T_j and a corresponding set of terminal node parameters $\boldsymbol{\theta}_j$, where $g_j(\mathbf{x}_i \mid T_j, \boldsymbol{\theta}_j)$ denotes the function that assigns the value $\mu_{bj} \in \boldsymbol{\theta}_j$ to $\mathbf{x}_i$ according to the partition defined by T_j.

The main idea is to construct this sum-of-trees model by imposing a prior that regularizes the fit, ensuring that the individual contribution of each tree remains small. Thus, each tree g_j explains a small and distinct portion of f. This is achieved through Bayesian backfitting MCMC [182], where successive fits to the residuals are added. In this sense, BART can be viewed as a Bayesian version of boosting.

[92] use the following prior structure

$$\begin{aligned}\pi(\{(T_1,\boldsymbol{\theta}_1),(T_2,\boldsymbol{\theta}_2),\ldots,(T_J,\boldsymbol{\theta}_J),\sigma^2\}) &= \left[\prod_{j=1}^{J}\pi(T_j,\boldsymbol{\theta}_j)\right]\pi(\sigma)\\ &= \left[\prod_{j=1}^{J}\pi(\boldsymbol{\theta}_j \mid T_j)\pi(T_j)\right]\pi(\sigma)\\ &= \left[\prod_{j=1}^{J}\prod_{b=1}^{B}\pi(\mu_{bj} \mid T_j)\pi(T_j)\right]\pi(\sigma).\end{aligned}$$

The prior for the binary tree structure has three components: (i) the probability that a node at depth $d = 0, 1, \ldots$ is nonterminal, given by $\alpha(1+d)^{-\beta}$, where $\alpha \in (0,1)$ and $\beta \in [0,\infty)$; the default values are $\alpha = 0.95$ and $\beta = 2$; (ii) a uniform prior over the set of available regressors to define the distribution of splitting variable assignments at each interior node; and (iii) a uniform prior over the discrete set of available splitting values to define the splitting rule assignment, conditional on the chosen splitting variable.

The prior for the terminal node parameters, $\pi(\mu_{bj} \mid T_j)$, is specified as $N(0, 0.5/(k\sqrt{J}))$. The observed values of y are scaled and shifted to lie within the range $y_{\min} = -0.5$ to $y_{\max} = 0.5$, and the default value $k = 2$ ensures that

$$P_{\pi(\mu_{bj}|T_j)}\left(\mathbb{E}[y \mid \mathbf{x}] \in (-0.5, 0.5)\right) = 0.95.$$

Note that this prior shrinks the effect of individual trees toward zero, ensuring that each tree contributes only a small amount to the overall prediction. Moreover, although the dependent variable is transformed, there is no need to transform the input variables, since the tree-splitting rules are invariant to monotonic transformations of the regressors.

The prior for σ^2 is specified as $\pi(\sigma^2) \sim v\lambda/\chi^2_v$. [92] recommend setting $v = 3$, and choosing λ such that $P(\sigma < \hat{\sigma}) = q, q = 0.9$, where $\hat{\sigma}$ is an estimate of the residual standard deviation from a saturated linear model, i.e., a model including all available regressors when $K < N$, or the standard deviation of y when $K \geq N$.

Finally, regarding the number of trees J, the default value is 200. As J increases from 1, BART's predictive performance typically improves substantially until it reaches a plateau, after which performance may degrade very slowly. However, if the primary goal is variable selection, using a smaller J is preferable, as it facilitates identifying the most important regressors by enhancing the internal competition among variables within a limited number of trees.

To sum up, the default hyperparameters are $(\alpha, \beta, k, J, v, q) = (0.95, 2, 2, 200, 3, 0.9)$; however, cross-validation can be used to tune these hyperparameters if desired. BART's predictive performance appears to be relatively robust to the choice of hyperparameters, provided they are set to

sensible values, except in cases where $K > N$, in which cross-validated tuning tends to yield better performance, albeit at the cost of increased computational time.

Given this specification, we can use a Gibbs sampler that cycles through the J trees. At each iteration, we sample from the conditional posterior distribution

$$\pi(T_j, \boldsymbol{\theta}_j \mid R_j, \sigma) = \pi(T_j \mid R_j, \sigma) \times \pi(\boldsymbol{\theta}_j \mid T_j, R_j, \sigma),$$

where $R_j = y - \sum_{k \neq j} g_k(\mathbf{x} \mid T_k, \boldsymbol{\theta}_k)$ represents the residuals excluding the contribution of the j-th tree.

The posterior distribution $\pi(T_j \mid R_j, \sigma)$ is explored using a Metropolis-Hastings algorithm, where the candidate tree is generated by one of the following moves: (i) growing a terminal node with probability 0.25; (ii) pruning a pair of terminal nodes with probability 0.25; (iii) changing a nonterminal splitting rule with probability 0.4; or (iv) swapping a rule between a parent and child node with probability 0.1 [91].

The posterior distribution of $\boldsymbol{\theta}_j$ is the product of the posterior distributions $\pi(\mu_{jb} \mid T_j, R_j, \sigma)$, which are Gaussian. The posterior distribution of σ is inverse-gamma. The posterior draws of μ_{bj} are then used to update the residuals R_{j+1}, allowing the sampler to proceed to the next tree in the cycle. The number of iterations in the Gibbs sampler does not need to be very large; for instance, 200 burn-in iterations and 1,000 post-burn-in iterations typically work well.

As we obtain posterior draws from the sum-of-trees model, we can compute point estimates at each $\mathbf{x}_i$ using the posterior mean, $\mathbb{E}[y \mid \mathbf{x}_i] = f(\mathbf{x}_i)$. Additionally, pointwise measures of uncertainty can be derived from the quantiles of the posterior draws. We can also identify the most relevant predictors by tracking the relative frequency with which each regressor appears in the sum-of-trees model across iterations. Moreover, we can perform inference on functionals of f, such as the *partial dependence functions* [138], which quantify the marginal effects of the regressors.

Specifically, if we are interested in the effect of $\mathbf{x}_s$ on y, while marginalizing over the remaining variables $\mathbf{x}_c$, such that $\mathbf{x} = [\mathbf{x}_s^\top \ \mathbf{x}_c^\top]^\top$, the partial dependence function is defined as

$$f(\mathbf{x}_s) = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_s, \mathbf{x}_{ic}),$$

where $\mathbf{x}_{ic}$ denotes the i-th observed value of $\mathbf{x}_c$ in the dataset.

Note that the calculation of the partial dependence function assumes that a subset of the variables is held fixed while averaging over the remainder. This assumption may be questionable when strong dependence exists among input variables, as fixing some variables while varying others may result in unrealistic or implausible combinations. Therefore, caution is warranted when interpreting the results.

[255] extended BART models to high-dimensional settings for prediction and variable selection, while [191, 177] applied them to causal inference. The asymptotic properties of BART models have been studied by [352, 353, 346].

Example: Simulation exercise to study BART performance

We use the *BART* package [385] in the **R** software environment to perform estimation, prediction, inference, and marginal analysis using Bayesian Additive Regression Trees. In addition to modeling continuous outcomes, this package also supports dichotomous, categorical, and time-to-event outcomes.

We adopt the simulation setting proposed by [137], which is also used by [92]:

$$y_i = 10\sin(\pi x_{i1}x_{i2}) + 20(x_{i3} - 0.5)^2 + 10x_{i4} + 5x_{i5} + \mu_i,$$

where $\mu_i \sim N(0, 1)$, for $i = 1, 2, \ldots, 500$.

We set the hyperparameters to $(\alpha, \beta, k, J, v, q) = (0.95, 2, 2, 200, 3, 0.9)$, with a burn-in of 100 iterations, a thinning parameter of 1, and 1,000 post-burn-in MCMC iterations. We analyze the trace plot of the posterior draws of σ to assess convergence, compare the true values of y with the posterior mean and 95% predictive intervals for both the training and test sets (using 80% of the data for training and 20% for testing), and visualize the relative importance of the regressors across different values of $J = 10, 20, 50, 100, 200$.

R code. Bayesian additive regression trees

```
rm(list = ls()); set.seed(10101)
library(BART); library(tidyr)
library(ggplot2); library(dplyr)
N <- 500; K <- 10
# Simulate the dataset
MeanFunct <- function(x){
  f <- 10*sin(pi*x[,1]*x[,2]) + 20*(x[,3]-.5)^2+10*x[,4]+5*x[,5]
  return(f)
}
sig2 <- 1
e <- rnorm(N, 0, sig2^0.5)
X <- matrix(runif(N*K),N,K)
y <- MeanFunct(X[,1:5]) + e
# Train and test
c <- 0.8
Ntrain <- floor(c * N)
Ntest <- N - Ntrain
X.train <- X[1:Ntrain, ]
y.train <- y[1:Ntrain]
X.test <- X[(Ntrain+1):N, ]
y.test <- y[(Ntrain+1):N]
# Hyperparameters
alpha <- 0.95; beta <- 2; k <- 2
v <- 3; q <- 0.9; J <- 200
# MCMC parameters
MCMCiter <- 1000; burnin <- 100; thinning <- 1
# Estimate BART
BARTfit <- wbart(x.train = X.train, y.train = y.train, x.test = X.test, base = alpha,
power = beta, k = k, sigdf = v, sigquant = q, ntree = J,
ndpost = MCMCiter, nskip = burnin, keepevery = thinning)

# Trace plot sigma
keep <- seq(burnin + 1, MCMCiter + burnin, thinning)
df_sigma <- data.frame(iteration = 1:length(keep), sigma = BARTfit$sigma[keep])
ggplot(df_sigma, aes(x = iteration, y = sigma)) + geom_line(color = "steelblue") + labs(title = "Trace Plot of Sigma", x = "Iteration", y = expression(sigma)) + theme_minimal()

# Prediction plot training
train_preds <- data.frame( y_true = y.train, mean = apply(BARTfit$yhat.train, 2, mean),
lower = apply(BARTfit$yhat.train, 2, quantile, 0.025), upper = apply(BARTfit$yhat.train, 2, quantile, 0.975)) %>% arrange(y_true) %>% mutate(index = row_number())
```

R code. Bayesian additive regression trees

```
ggplot(train_preds, aes(x = index)) + geom_ribbon(aes(ymin = lower, ymax = upper, fill = "95% Interval"), alpha = 0.4, show.legend = TRUE) + geom_line(aes(y = mean, color = "Predicted Mean")) + geom_line(aes(y = y_true, color = "True y")) + scale_color_manual(name = "Line", values = c("Predicted Mean" = "blue", "True y" = "black")) + scale_fill_manual(name = "Interval", values = c("95% Interval" = "lightblue")) + labs(title = "Training Data: Ordered Predictions with 95% Intervals",
x = "Ordered Index", y = "y") + theme_minimal()
# Prediction plot test
test_preds <- data.frame( y_true = y.test, mean = apply(BARTfit$yhat.test, 2, mean), lower = apply(BARTfit$yhat.test, 2, quantile, 0.025), upper = apply(BARTfit$yhat.test, 2, quantile, 0.975)) %>% arrange(y_true) %>% mutate(index = row_number())

ggplot(test_preds, aes(x = index)) + geom_ribbon(aes(ymin = lower, ymax = upper, fill = "95% Interval"), alpha = 0.4, show.legend = TRUE) + geom_line(aes(y = mean, color = "Predicted Mean")) + geom_line(aes(y = y_true, color = "True y")) + scale_color_manual(name = "Line", values = c("Predicted Mean" = "blue", "True y" = "black")) + scale_fill_manual(name = "Interval", values = c("95% Interval" = "lightblue")) + labs(title = "Test Data: Ordered Predictions with 95% Intervals",
x = "Ordered Index", y = "y") + theme_minimal()
# Relevant regressors
Js <- c(10, 20, 50, 100, 200)
VarImportance <- matrix(0, length(Js), K)
l <- 1
for (j in Js){
  BARTfit <- wbart(x.train = X.train, y.train = y.train, x.test = X.test, base = alpha,
  power = beta, k = k, sigdf = v, sigquant = q, ntree = j,
  ndpost = MCMCiter, nskip = burnin, keepevery = thinning)
  VarImportance[l, ] <- BARTfit[["varcount.mean"]]/j
  l <- l + 1
}

rownames(VarImportance) <- c("10", "20", "50", "100", "200")
colnames(VarImportance) <- as.character(1:10)
importance_df <- as.data.frame(VarImportance) %>% mutate(trees = rownames(.)) %>%
pivot_longer(cols = -trees, names_to = "variable", values_to = "percent_used")
importance_df$variable <- as.numeric(importance_df$variable)
importance_df$trees <- factor(importance_df$trees, levels = c("10", "20", "50", "100", "200"))

ggplot(importance_df, aes(x = variable, y = percent_used, color = trees, linetype = trees)) + geom_line() + geom_point() + scale_color_manual(values = c("10" = "red", "20" = "green", "50" = "blue", "100" = "cyan", "200" = "magenta")) + scale_x_continuous(breaks = 1:10) + labs(x = "variable", y = "percent used", color = "#trees", linetype = "#trees") + theme_minimal()
```

Figure 12.3 displays the trace plot of σ, which appears to have reached a stationary distribution. However, the posterior draws are slightly lower than the true value (1).

FIGURE 12.3
Trace plot of the posterior draws of σ in the BART simulation.

Figures 12.4 and 12.5 compare the true values of y with the posterior mean and 95% predictive intervals. The BART model performs well in both sets, and the intervals in the test set are notably wider than those in the training set.

Figure 12.6 shows the relative frequency with which each variable is used in the trees, a measure of variable relevance. When the number of trees is small, the algorithm more clearly identifies the most relevant predictors (variables 1–5). As the number of trees increases, this discrimination gradually disappears.

12.4 Gaussian process

A Gaussian Process (GP) is an infinite collection of random variables, any finite subset of which follows a joint Gaussian distribution. A GP is fully specified by its mean function and covariance function, that is,

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')),$$

FIGURE 12.4
Training data: true y versus posterior mean and 95% predictive intervals.

FIGURE 12.5
Test data: true y versus posterior mean and 95% predictive intervals.

where $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$. It is common to assume $m(\mathbf{x}) = 0$ to simplify calculations, although this is not required.

FIGURE 12.6
Relative frequency of variable usage in the BART trees for different numbers of trees.

Perhaps the most commonly used covariance function in Gaussian Processes (GPs) is the *squared exponential* kernel (or *radial basis function*) [206], defined as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\|\mathbf{x} - \mathbf{x}'\|^2\right),$$

where σ_f^2 is the signal variance, which controls the vertical variation (amplitude) of the function, l is the length-scale parameter, which determines how quickly the function varies with features distance, and $\|\mathbf{x}-\mathbf{x}'\|^2$ is the squared Euclidean distance between the feature vectors $\mathbf{x}$ and $\mathbf{x}'$.

The squared exponential kernel implies that the function is infinitely differentiable, leading to very smooth function draws. While this smoothness may be desirable in some applications, it can be too restrictive in others. Alternative kernels like the Matérn class allow for more flexibility by controlling the degree of differentiability [336].

A GP can be interpreted as a prior distribution over a space of functions. The starting point in working with GPs is the specification of this prior before any data are observed. The following code illustrates five sample paths drawn from a GP with a squared exponential kernel, assuming a signal variance $\sigma_f^2 = 1$ and a length-scale $l = 0.2$, evaluated over a grid of input values $x \in [0, 1]$. A small *jitter term* is added to the covariance matrix to ensure numerical stability during simulation. Figure 12.7 displays the five realizations drawn from the Gaussian Process.

R code. Simulation of five realizations: Gaussian process

```r
rm(list = ls())
set.seed(10101)

library(ggplot2); library(dplyr)
library(tidyr); library(MASS)

# Simulation setup
n <- 100
x <- seq(0, 1, length.out = n)
sigma_f <- 1
l <- 0.2
sigma_n <- 1e-8

# Squared Exponential Kernel function
SE_kernel <- function(x1, x2, sigma_f, l) {
  outer(x1, x2, function(a, b) sigma_f^2 * exp(-0.5 * (a - b)^2 / l^2))
}

K <- SE_kernel(x, x, sigma_f, l) + diag(sigma_n, n)
samples <- mvrnorm(n = 5, mu = rep(0, n), Sigma = K)

# Transpose and rename columns to f1, f2, ..., f5
samples_t <- t(samples)
colnames(samples_t) <- paste0("f", 1:5)

# Convert to tidy data frame
df <- data.frame(x = x, samples_t) |>
pivot_longer(cols = -x, names_to = "draw", values_to = "value")

# Plot
ggplot(df, aes(x = x, y = value, color = draw)) + geom_line(linewidth = 1) +
labs( title = "Simulated Gaussian Process Draws", x = "x", y = "f(x)", color = "Function" ) + theme_minimal(base_size = 14) + theme(legend.position = "top")
```

Thus, for any finite set of feature points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, the corresponding function values follow a multivariate Gaussian distribution:

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} \sim N(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X})),$$

FIGURE 12.7
Simulation: Gaussian process.

where the (i, j)-th entry of the covariance matrix $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is given by $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

If we are interested in the properties of a function evaluated at a finite set of input points $\{(f_i, x_i)\}_{i=1}^N$, inference can be performed using only those points, effectively disregarding the uncountably infinite values the function may take elsewhere.

The following code illustrates how to perform inference for a GP given four observed points $\{(f_i, x_i)\}_{i=1}^4$, assuming that the true underlying process is

$$f_i = \sin(2\pi x_i).$$

The inference is based on the properties of the conditional Gaussian distribution (see below). Figure 12.8 shows that the posterior mean (solid blue line) interpolates the observed points (red dots). Moreover, the level of uncertainty (light blue shaded area) increases in regions that are farther from the observed inputs, where the posterior mean tends to deviate more from the true underlying function (dashed green line).

In situations where the input locations can be selected—such as in experimental designs, *active learning strategies* can be employed to choose the points that minimize predictive uncertainty. This is typically achieved by optimizing an *acquisition function* that quantifies the expected informativeness of candidate locations [374].

Consequently, GPs play a central role in *Bayesian optimization*, a stochastic method for finding the maximum of expensive or unknown objective functions. In this approach, a prior is placed over the objective function, which

is then updated using observed data to form a posterior distribution over possible functions. This posterior guides the selection of new input points by balancing exploration and exploitation through the acquisition function [59].

R code. Simulation: $f_i = \sin(2\pi x_i)$ ***and Gaussian process***

```r
rm(list = ls()); set.seed(10101)
library(ggplot2); library(MASS)
# Define the squared exponential kernel
SE_kernel <- function(x1, x2, sigma_f, l) {
  outer(x1, x2, function(a, b) sigma_f^2 * exp(-0.5 * (a - b)^2 / l^2))
}
# Define the input space and observed points
x_star <- seq(0, 1, length.out = 200)
x0 <- c(0.1, 0.2, 0.5, 0.9)
y0 <- sin(2 * pi * x0)
# Hyperparameters
sigma_f <- 1
l <- 0.2
sigma_n <- 1e-8  # Jitter term for stability
# Compute covariance matrices
K_x0x0 <- SE_kernel(x0, x0, sigma_f, l) + diag(sigma_n, length(x0))
K_xstarx0 <- SE_kernel(x_star, x0, sigma_f, l)
K_xstarxstar <- SE_kernel(x_star, x_star, sigma_f, l) + diag(sigma_n, length(x_star))
# Compute posterior mean and covariance
K_inv <- solve(K_x0x0)
posterior_mean <- K_xstarx0 %*% K_inv %*% y0
posterior_cov <- K_xstarxstar - K_xstarx0 %*% K_inv %*% t(K_xstarx0)
# Sample from the posterior
sample_draw <- sin(2 * pi * x_star)
# Compute 95% intervals
posterior_sd <- sqrt(diag(posterior_cov))
lower <- posterior_mean - 1.96 * posterior_sd
upper <- posterior_mean + 1.96 * posterior_sd
# Data frame for plotting
df <- data.frame(
x = x_star,
mean = posterior_mean,
lower = lower,
upper = upper,
sample = sample_draw
)
obs <- data.frame(x = x0, y = y0)
# Plot
ggplot(df, aes(x = x)) + geom_ribbon(aes(ymin = lower, ymax = upper), fill = "lightblue", alpha = 0.4) + geom_line(aes(y = mean), color = "blue", linewidth = 1.2) + geom_line(aes(y = sample), color = "darkgreen", linewidth = 1, linetype = "dashed") + geom_point(data = obs, aes(x = x, y = y), color = "red", size = 3) + labs( title = "Gaussian Process with Conditioning Points", x = "x", y = "f(x)", caption = "Blue: Posterior mean | Light blue: 95% interval | Dashed green: Population | Red: Observed points" ) + theme_minimal(base_size = 14)
```

FIGURE 12.8
Simulation: $f_i = \sin(2\pi x_i)$ and Gaussian process

In practice, we have an observed dataset $\{(y_i, \mathbf{x}_i)\}_{i=1}^N$ such that

$$y_i = f(\mathbf{x}_i) + \mu_i,$$

where $\mu_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$. This means that y_i is a noisy observation of $f(\mathbf{x}_i)$.

Thus, the marginal distribution of the observed outputs is

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}_N),$$

where $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is the covariance matrix generated by the GP kernel evaluated at the training inputs.

Note that this implies the log marginal likelihood is given by

$$\log p(\mathbf{y} \mid \mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma^2 \mathbf{I}_N)^{-1} \mathbf{y} - \frac{1}{2} \log \left|\mathbf{K} + \sigma^2 \mathbf{I}_N\right| - \frac{N}{2} \log 2\pi.$$

We can adopt an empirical Bayes approach to estimate the hyperparameters of the GP prior by maximizing the log marginal likelihood with respect to the kernel parameters (e.g., σ_f^2, l) and the noise variance σ^2.

To make predictions at a new set of features $\mathbf{X}_*$, we consider the joint distribution:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}_N & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right).$$

Using the conditional distribution of a multivariate Gaussian, the *posterior predictive distribution* [336] is:

$$\mathbf{f}_* \mid \mathbf{y} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)),$$

where

$$\bar{\mathbf{f}}_* = \mathbb{E}[\mathbf{f}_* \mid \mathbf{y}, \mathbf{X}, \mathbf{X}_*] = \mathbf{K}(\mathbf{X}_*, \mathbf{X})[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}_N]^{-1}\mathbf{y},$$
$$\text{cov}(\mathbf{f}_*) = \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) - \mathbf{K}(\mathbf{X}_*, \mathbf{X})[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}_N]^{-1}\mathbf{K}(\mathbf{X}, \mathbf{X}_*).$$

Therefore, GP regression provides a flexible and efficient nonparametric framework for predicting unobserved responses, with accuracy that improves as more data become available. GPs are widely used due to their favorable computational properties, including the availability of closed-form expressions, and posterior consistency under mild conditions [93, 392]. Moreover, predictive performance can be further enhanced by incorporating derivative information, as the derivative of a GP is itself a GP [383, 206].

However, a major limitation of GPs is the need to invert an $N \times N$ covariance matrix, which requires $O(N^3)$ computational operations, making them computationally expensive for large datasets. To address this, several scalable methods have been proposed that reduce the computational burden. For instance, [416, 144, 311] develop algorithms that reduce complexity to $O(N)$.

Example: Simulation exercise to study GP performance

We simulate the process

$$f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2}),$$

where x_{i1} and x_{i2} are independently drawn from a uniform distribution on the interval $[0, 1]$, for $i = 1, 2, \ldots, 100$.

We use the *DiceKriging* package in **R** to estimate and make predictions using a GP. This package applies maximum likelihood estimation to infer the length-scale parameters (l_k) and the signal variance (σ_f^2). Note that there are two separate length-scale parameters, one for each input variable. The following code illustrates how to carry out this example, and Figure 12.9 displays a 3D plot with the observed points and the posterior mean surface. The package also provides pointwise credible intervals for the predictions.

R code. Simulation:
$f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2})$ ***and Gaussian process***

```r
# Load required packages
library(DiceKriging)
library(rgl)

# Simulate training data
set.seed(10101)
n_train <- 100
x1 <- runif(n_train)
x2 <- runif(n_train)
X_train <- data.frame(x1 = x1, x2 = x2)

# True function without noise
f_train <- sin(2 * pi * X_train$x1) + cos(2 * pi * X_train$x2) + sin(X_train$x1 * X_train$x2)

# Fit Gaussian Process
fit_km <- km(design = X_train, response = f_train, covtype = "gauss", nugget = 1e-10)

# Prediction grid
grid_points <- 30
x1_seq <- seq(0, 1, length.out = grid_points)
x2_seq <- seq(0, 1, length.out = grid_points)
grid <- expand.grid(x1 = x1_seq, x2 = x2_seq)

# Predict GP surface
pred <- predict(fit_km, newdata = grid, type = "UK")
z_pred <- matrix(pred$mean, nrow = grid_points, ncol = grid_points)

# Plot
persp3d(x = x1_seq, y = x2_seq, z = z_pred,
col = "lightblue", alpha = 0.7,
xlab = "x1", ylab = "x2", zlab = "GP Mean")
points3d(x = X_train$x1, y = X_train$x2, z = f_train, col = "red", size = 8)

fit_km@covariance@range.val # length-scale
fit_km@covariance@sd2 # Signal variance
```

A limitation of the *DiceKriging* package is that it is designed for deterministic simulations and, consequently, does not estimate the noise variance. Therefore, in Exercise 7, we ask to simulate the process

$$f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2}) + \mu_i,$$

FIGURE 12.9
Simulation: $f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2})$ and Gaussian process

where $\mu_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$, and to use an empirical Bayes approach to estimate the hyperparameters. These estimated hyperparameters should then be used to perform GP prediction.

12.5 Tall data problems

In this section, we explore several methods developed to perform Bayesian inference when the sample size is large, particularly when there is a large number of observational units, commonly referred to as *tall datasets*.

Bayesian inference in such settings is computationally demanding because each iteration of an MCMC algorithm requires evaluating the likelihood function over all N observations. For large N, this renders standard MCMC methods prohibitively expensive. Recent efforts have focused on developing scalable Monte Carlo algorithms that significantly reduce the computational cost compared to standard approaches. One alternative is to use *Variational Bayes* (see Chapter 14); however, it can be challenging to implement and may exhibit limitations in uncertainty quantification, particularly for the joint posterior distribution. Another alternative is the *Integrated Nested Laplace Approximation* (INLA, see Chapter 14); however, its computational cost grows exponentially with the dimension of the parameter space.

In scenarios where observations are assumed to be independent, two main frameworks have been proposed to scale MCMC algorithms: *divide-and-conquer approaches* and *subsampling-based algorithms*. Divide-and-conquer methods partition the dataset into disjoint subsets, run MCMC independently on each batch to obtain subposteriors, and then combine them to approximate the full posterior distribution. Subsampling-based algorithms, on the other hand, aim to reduce the number of data points used to evaluate the likelihood at each iteration, often relying on *pseudo-marginal* MCMC methods [13]. The key idea of pseudo-marginal MCMC is to augment the model with latent variables such that the sample average of the likelihood, computed over draws from these latent variables, provides an unbiased estimator of the marginal likelihood. This approach is particularly valuable when the marginal likelihood is not available in closed form. Moreover, the same principles can be adapted to reduce the computational burden of evaluating the log-likelihood. For an excellent review of divide-and-conquer and subsampling-based approaches, see [24].

12.5.1 Divide-and-conquer methods

In *divide-and-conquer* methods, the main idea is to partition the dataset and distribute the subsets across multiple computing machines/cores. An independent MCMC algorithm is then executed on each subset to obtain a corresponding *subposterior* distribution. The central challenge lies in accurately and efficiently combining these subposteriors into a single approximation of the full posterior distribution. Several approaches have been proposed to address this issue. For instance, [198, 367, 338, 368] introduce the *Consensus Monte Carlo* algorithm; [411] develop a method based on the *Weierstrass sampler* for parallelizing MCMC; [286] propose using the *geometric median of posterior distributions*; and [425] suggest combining *rescaled subposteriors*.

In divide-and-conquer methods, the dataset is partitioned into B disjoint batches $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_B$, and the posterior is rewritten using the identity:

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}) \propto \prod_{b=1}^{B} \pi(\boldsymbol{\theta})^{1/B} p(\mathbf{y}_b \mid \boldsymbol{\theta}),$$

which implies that the full posterior is proportional to the product of appropriately rescaled subposteriors.

Consensus Monte Carlo (CMC) operates by running separate Monte Carlo algorithms on each subset in parallel, and then averaging the resulting posterior draws. Specifically, given samples $\boldsymbol{\theta}_b^{(s)}$, for $b = 1, 2, \ldots, B$ and $s = 1, 2, \ldots, S$, obtained independently from each batch $\mathbf{y}_b$, the s-th draw from the consensus posterior is computed as:

$$\boldsymbol{\theta}^{(s)} = \left(\sum_{b=1}^{B} \mathbf{w}_b \right)^{-1} \sum_{b=1}^{B} \mathbf{w}_b \boldsymbol{\theta}_b^{(s)},$$

where the optimal weight is the inverse covariance matrix of the subposterior, i.e., $\mathbf{w}_b = \mathrm{Var}^{-1}(\boldsymbol{\theta} \mid \mathbf{y}_b)$. In practice, one may use the marginal variances of each parameter to simplify the computation, which can still yield good performance.

When each subposterior $\pi_b(\boldsymbol{\theta} \mid \mathbf{y}_b)$ is Gaussian, the full posterior $\pi(\boldsymbol{\theta} \mid \mathbf{y})$ is also Gaussian, and can be recovered exactly by combining the subposteriors using simple rules based on their means and covariances [367, 368]. In the non-Gaussian case, standard asymptotic results in Bayesian inference (see Chapter 1) imply that the posterior distributions converge to a Gaussian distribution as the batch size increases. Alternative merging procedures that are more robust to non-Gaussianity have also been proposed [292, 287]; however, it remains difficult to quantify the approximation error in these approaches. Moreover, this procedure is limited to continuous parameter spaces and may exhibit small-sample bias; that is, when the dataset is divided into small batches, the subposterior distributions may be biased. In such cases, jackknife bias corrections are recommended to reduce the overall approximation error.

In particular, we perform CMC following Algorithm A33 [367]. Next, we compute the CMC posterior repeatedly, each time leaving out one of the B subsets. Let $\pi_{-b}(\boldsymbol{\theta} \mid \mathbf{y})$ denote the resulting posterior when subset b is excluded. The average of these leave-one-out posteriors is denoted by

$$\bar{\pi}_{-b}(\boldsymbol{\theta} \mid \mathbf{y}) = \frac{1}{B} \sum_{b=1}^{B} \pi_{-b}(\boldsymbol{\theta} \mid \mathbf{y}).$$

Then, the jackknife bias-corrected posterior is given by

$$\pi_{\mathrm{jk}}(\boldsymbol{\theta} \mid \mathbf{y}) = B \cdot \pi_{\mathrm{CMC}}(\boldsymbol{\theta} \mid \mathbf{y}) - (B-1) \cdot \bar{\pi}_{-b}(\boldsymbol{\theta} \mid \mathbf{y}),$$

where $\pi_{\mathrm{CMC}}(\boldsymbol{\theta} \mid \mathbf{y})$ is the original CMC posterior based on all B subsets.

Algorithm A33 Consensus Monte Carlo algorithm

1: Divide the dataset into B disjoint batches $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_B$
2: Run B separate MCMC algorithms to sample $\boldsymbol{\theta}_b^{(s)} \sim \pi(\boldsymbol{\theta} \mid \mathbf{y}_b)$, $b = 1, 2, \ldots, B$, and $s = 1, 2, \ldots, S$ using the prior distribution $\pi(\boldsymbol{\theta})^{1/B}$
3: Combine the posterior draws using $\boldsymbol{\theta}^{(s)} = \left(\sum_{b=1}^{B} \mathbf{w}_b\right)^{-1} \sum_{b=1}^{B} \mathbf{w}_b \boldsymbol{\theta}_b^{(s)}$ using $\mathbf{w}_b = \mathrm{Var}^{-1}(\boldsymbol{\theta} \mid \mathbf{y}_b)$

The main difficulty is how to effectively merge the subposterior distributions, especially when their supports are not well-aligned. This misalignment can lead to poor scalability with an increasing number of batches. Moreover, most theoretical guarantees for these methods are asymptotic in the size of each batch, which may limit their performance in practice [24].

12.5.2 Subsampling-based algorithms

An alternative to divide-and-conquer methods is to avoid evaluating the likelihood over all observations, which requires $O(N)$ operations. Instead, the likelihood is approximated using a smaller subset of observations, $n \ll N$, in order to reduce the computational burden of the algorithm. The starting point is the log-likelihood function for N independent observations:

$$\log p(\mathbf{y} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(y_i \mid \boldsymbol{\theta}).$$

The literature has focused on the log-likelihood because it is a sum over independent contributions, which is analogous to the problem of estimating a population total.

A class of subsampling methods relies on estimating the marginal likelihood via the *pseudo-marginal* approach. Examples include the confidence sampler [23], the Firefly Monte Carlo algorithm [262], whose relationship to subsampling MCMC is formally established in [24], and approaches using data-expanded and parameter-expanded control variates [316].

The intuition behind the pseudo-marginal method is straightforward: introduce a set of auxiliary random variables $\mathbf{z} \sim p(\mathbf{z})$, such that the marginal likelihood can be written as an expectation with respect to $\mathbf{z}$:

$$\mathbb{E}_{\mathbf{z}}[p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{z})] = \int_{\mathcal{Z}} p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} = p(\mathbf{y} \mid \boldsymbol{\theta}).$$

This implies that

$$\hat{p}(\mathbf{y} \mid \boldsymbol{\theta}) = \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{z}^{(s)}),$$

where $\mathbf{z}^{(s)} \sim p(\mathbf{z})$, is an unbiased estimator of the marginal likelihood.

As a result, the pseudo-marginal method enables exact simulation-based inference for $p(\mathbf{y} \mid \boldsymbol{\theta})$ in settings where the likelihood cannot be evaluated analytically [13], for instance, in non-linear random effects models (see also approximate methods such as *approximate Bayesian computation* and *Bayesian synthetic likelihood* in Chapter 14). [13] show that replacing the likelihood with an unbiased and positive estimator within the Metropolis–Hastings (MH) algorithm yields samples from the correct posterior distribution.

The pseudo-marginal likelihood approach can also be applied in settings where the sample size is so large that evaluating the full likelihood at each iteration of an MCMC algorithm becomes computationally prohibitive. In such cases, the likelihood can be approximated using a small subset of observations, $n \ll N$. The choice of the subset size n is particularly important, as it directly affects the variance of the likelihood estimator, which in turn is critical to ensuring an efficient Metropolis–Hastings (MH) algorithm.

In particular, a likelihood estimator with high variance may result in an

accepted draw that overestimates the likelihood. As a consequence, subsequent proposals are unlikely to be accepted, causing the algorithm to become stuck and leading to a very low acceptance rate. Therefore, the choice of n determines the computational efficiency of the algorithm: a small n increases the estimator's variance, which reduces the acceptance rate, whereas a large n increases the number of likelihood evaluations per iteration.

[317] recommend targeting a likelihood estimator variance between 1 and 3.3 to optimize computational efficiency, as supported by the findings of [310].

Let $\ell_i(y_i \mid \boldsymbol{\theta}) = \log p(y_i \mid \boldsymbol{\theta})$ denote the contribution of the i-th observation to the log-likelihood, and let $z_1, \ldots, z_N$ be latent binary variables such that $z_i = 1$ indicates that y_i is included in a subsample of size n, selected without replacement. Then, an unbiased estimator of the log-likelihood is given by

$$\hat{\ell}(\mathbf{y} \mid \boldsymbol{\theta}) = \frac{N}{n} \sum_{i=1}^{N} \ell_i(y_i \mid \boldsymbol{\theta}) z_i.$$

However, note that what we require is an unbiased estimator of the likelihood, not the log-likelihood. Consequently, a bias correction is needed:

$$\hat{p}(\mathbf{y} \mid \boldsymbol{\theta}) = \exp\left\{\hat{\ell}(\mathbf{y} \mid \boldsymbol{\theta}) - \frac{1}{2}\sigma^2_{\hat{\ell}}(\boldsymbol{\theta})\right\},$$

where $\sigma^2_{\hat{\ell}}(\boldsymbol{\theta})$ denotes the variance of $\hat{\ell}(\mathbf{y} \mid \boldsymbol{\theta})$ [70]. This correction is exact if $\sigma^2_{\hat{\ell}}(\boldsymbol{\theta})$ is known and $\hat{\ell}(\mathbf{y} \mid \boldsymbol{\theta})$ follows a normal distribution.

Given the importance of controlling the variance of the log-likelihood estimator in subsampling methods, and the limitations of simple random sampling in achieving low variability, [316] propose a highly efficient, unbiased estimator of the log-likelihood based on *control variates*, specifically through data-expanded and parameter-expanded control variates. The key idea is to construct a function $q_i(\boldsymbol{\theta})$ that is highly correlated with the log-likelihood contribution $\ell_i(y_i \mid \boldsymbol{\theta})$, thereby stabilizing the log-likelihood estimator. In particular, [316] introduce a *difference estimator* of the form:

$$\hat{\ell}_{\mathrm{DE}}(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{z}) = \sum_{i=1}^{N} q_i(\boldsymbol{\theta}) + \frac{N}{n} \sum_{i:z_i=1} \left(\ell_i(y_i \mid \boldsymbol{\theta}) - q_i(\boldsymbol{\theta})\right).$$

This estimator $\hat{\ell}_{\mathrm{DE}}(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{z})$ is unbiased for the full log-likelihood $\log p(\mathbf{y} \mid \boldsymbol{\theta})$.

[316] propose constructing $q_i(\boldsymbol{\theta})$ using a second-order Taylor expansion of the log-likelihood around a central value of $\boldsymbol{\theta}$, such as the mode. An alternative approach is to perform a second-order Taylor expansion around the nearest centroid of each observation, where the centroids are obtained from a pre-clustering of the data.

The first approach can perform poorly when the current draw $\boldsymbol{\theta}$ is far from the central expansion point, leading to inaccurate approximations. The second approach encounters difficulties in high-dimensional settings due to

the curse of dimensionality: many observations will be far from their assigned centroid. To address these issues, the authors propose an adaptive strategy: initialize the algorithm using data-expanded control variates, and switch to parameter-expanded control variates once the sampler reaches a region closer to the center of the parameter space.

It is important to note that this strategy targets an approximation to the posterior distribution, due to the small bias introduced by the difference estimator. However, this bias diminishes rapidly and has a negligible impact on the quality of the posterior inference.

Once a good estimator of the log-likelihood is obtained, meaning it has low variance, the likelihood can be recovered using the appropriate bias correction. This corrected likelihood estimator is then used within the acceptance probability of the Metropolis–Hastings algorithm (see Section 4.1.2), resulting in the so-called pseudo-marginal Metropolis–Hastings (PMMH) method. This strategy significantly reduces computational cost in tall data settings.

Another class of subsampling methods, which does not rely on the pseudo-marginal likelihood, consists of stochastic gradient MCMC algorithms. These methods begin based on ideas from stochastic gradient descent [341] and the Langevin diffusion-based stochastic differential equations.

The point of departure is the unnormalized posterior distribution:

$$\begin{aligned}
\pi(\boldsymbol{\theta} \mid \mathbf{y}) &\propto \pi(\boldsymbol{\theta}) \prod_{i=1}^{N} p(y_i \mid \boldsymbol{\theta}) \\
&= \exp\left\{ \sum_{i=1}^{N} \left[\frac{1}{N} \log \pi(\boldsymbol{\theta}) + \log p(y_i \mid \boldsymbol{\theta}) \right] \right\} \\
&= \exp\left\{ -\sum_{i=1}^{N} U_i(\boldsymbol{\theta}) \right\} \\
&= \exp\left\{ -U(\boldsymbol{\theta}) \right\},
\end{aligned}$$

where $\boldsymbol{\theta} \in \mathbb{R}^K$, $U_i(\boldsymbol{\theta}) = -\frac{1}{N} \log \pi(\boldsymbol{\theta}) - \log p(y_i \mid \boldsymbol{\theta})$, and $U(\boldsymbol{\theta}) = \sum_{i=1}^{N} U_i(\boldsymbol{\theta})$ is assumed to be continuous and differentiable almost everywhere.

The advantage of this formulation is that, under mild regularity conditions [345], the Langevin diffusion process

$$d\boldsymbol{\theta}(s) = -\frac{1}{2} \nabla U(\boldsymbol{\theta}(s))\, ds + d\mathbf{B}_s,$$

has $\pi(\boldsymbol{\theta} \mid \mathbf{y})$ as its stationary distribution. Here, $\nabla U(\boldsymbol{\theta}(s))$ is the drift term, and $\mathbf{B}_s$ is a K-dimensional Brownian motion.[3]

Using an Euler-Maruyama discretization of the Langevin diffusion gives a

[3] A Brownian motion is a continuous-time stochastic process that starts at zero, has independent increments with $B(s) - B(t) \sim N(0, s-t)$, and is continuous almost surely but nowhere differentiable.

proposal draw from the posterior:

$$\boldsymbol{\theta}^{(c)} = \boldsymbol{\theta}^{(s)} - \frac{\epsilon}{2}\nabla U(\boldsymbol{\theta}^{(s)}) + \boldsymbol{\psi},$$

where $\boldsymbol{\psi} \sim \mathcal{N}(\mathbf{0}, \epsilon\mathbf{I}_K)$ and $\epsilon > 0$ is a suitably chosen step size (learning rate). This proposal is used within a Metropolis–Hastings algorithm (see Section 4.1.2) to correct for the discretization error introduced by the Euler approximation. This method is known as the *Metropolis-adjusted Langevin algorithm* (MALA) [345].

A simpler variant, known as the *unadjusted Langevin algorithm* (ULA), omits the acceptance step. As a result, ULA produces a biased approximation of the posterior distribution. However, a major computational bottleneck in both MALA and ULA is the requirement to evaluate the full gradient $\nabla U(\boldsymbol{\theta}) = \sum_{i=1}^{N} \nabla U_i(\boldsymbol{\theta})$ at every iteration, which becomes computationally prohibitive when N is large.

To overcome this limitation, [414] proposed the *Stochastic Gradient Langevin Dynamics* (SGLD), which replaces the full gradient with an unbiased estimate computed using a mini-batch of data. Given a random sample of size $n \ll N$, the stochastic gradient estimate at iteration s is:

$$\hat{\nabla} U(\boldsymbol{\theta})^{(n)} = \frac{N}{n} \sum_{i \in \mathcal{S}_n} \nabla U_i(\boldsymbol{\theta}), \tag{12.1}$$

where $\mathcal{S}_n \subset \{1, 2, \ldots, N\}$ is a randomly selected subset of dimension n without replacement.

Therefore,

$$\boldsymbol{\theta}^{(s+1)} = \boldsymbol{\theta}^{(s)} - \frac{\epsilon_s}{2}\hat{\nabla} U(\boldsymbol{\theta}^{(s)})^{(n)} + \boldsymbol{\psi}_s,$$

such that $\sum_{s=1}^{\infty} \epsilon_s = \infty$ and $\sum_{s=1}^{\infty} \epsilon_s^2 < \infty$. These conditions guarantee almost sure convergence: the former condition allows the algorithm to continue exploring the parameter space (no premature convergence), and the latter ensures that the cumulative noise does not explode.

[395] formally show that, under verifiable assumptions, the SGLD algorithm is consistent. That is, given a test function $\phi(\boldsymbol{\theta}) : \mathbb{R}^K \to \mathbb{R}$,

$$\lim_{S \to \infty} \frac{\epsilon_1 \phi(\boldsymbol{\theta}_1) + \epsilon_2 \phi(\boldsymbol{\theta}_2) + \cdots + \epsilon_S \phi(\boldsymbol{\theta}_S)}{\sum_{s=1}^{S} \epsilon_s} = \int_{\mathbb{R}^K} \phi(\boldsymbol{\theta}) \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta}.$$

Moreover, the algorithm satisfies a central limit theorem: $\lim_{S \to \infty} \pi_S(\phi(\boldsymbol{\theta})) = \pi(\phi(\boldsymbol{\theta}))$, and its asymptotic bias-variance decomposition is characterized by a functional of ϵ_s, such that the optimal step size that minimizes the asymptotic mean squared error is proportional to $s^{-1/3}$. In the common practice of using a constant step size, it has been shown that the optimal choice to minimize the asymptotic mean squared error is of order $S^{-1/3}$ [407]. However, we recommend tuning this parameter based on the specific application, guided by the theoretical results presented here.

Importantly, this iterative process does not require the computation of acceptance probabilities, which significantly reduces the computational burden. Empirical evidence suggests that SGLD often outperforms the Metropolis–Hastings algorithm when applied to large datasets under a fixed computational budget [251].

Algorithm A34 summarizes the SGLD procedure [294].

Algorithm A34 Stochastic gradient Langevin dynamic algorithm

1: Set $\boldsymbol{\theta}^{(0)}$ and the step size schedule ϵ_s
2: **for** $s = 1, \ldots, S$ **do**
3: Draw $\mathcal{S}_n$ of size n from $i = \{1, 2, \ldots, N\}$ without replacement
4: Calculate $\hat{\nabla} U(\boldsymbol{\theta})^{(n)}$ using Equation 12.1
5: Draw $\boldsymbol{\psi}_s \sim N(\mathbf{0}, \epsilon_s \mathbf{I}_K)$
6: Update $\boldsymbol{\theta}^{(s+1)} \leftarrow \boldsymbol{\theta}^{(s)} - \frac{\epsilon_s}{2} \hat{\nabla} U(\boldsymbol{\theta}^{(s)})^{(n)} + \boldsymbol{\psi}_s$
7: **end for**

A critical component of the SGLD algorithm is the estimation of the stochastic gradient (Equation 12.1), particularly because high variability in this estimator can lead to algorithmic instability, a challenge also encountered in pseudo-marginal methods, as described previously. To mitigate this issue, the literature also employs *control variates* to reduce the variance of the estimator. The core idea is to construct a simple function $u_i(\boldsymbol{\theta})$ that is highly correlated with $\nabla U_i(\boldsymbol{\theta})$ and has a known expectation. This correlation allows the fluctuations in $u_i(\boldsymbol{\theta})$ to "cancel out" some of the noise in $\nabla U_i(\boldsymbol{\theta})$, thereby stabilizing the stochastic gradient estimates. Specifically,

$$\sum_{i=1}^{N} \nabla U_i(\boldsymbol{\theta}) = \sum_{i=1}^{N} u_i(\boldsymbol{\theta}) + \sum_{i=1}^{N} \left(\nabla U_i(\boldsymbol{\theta}) - u_i(\boldsymbol{\theta}) \right),$$

which leads to the following unbiased estimator:

$$\sum_{i=1}^{N} u_i(\boldsymbol{\theta}) + \frac{N}{n} \sum_{i \in \mathcal{S}_n} \left(\nabla U_i(\boldsymbol{\theta}) - u_i(\boldsymbol{\theta}) \right).$$

To construct effective control variates, one common strategy is to first approximate the posterior mode $\hat{\boldsymbol{\theta}}$ using stochastic gradient descent (SGD), which serves as the initialization point for Algorithm A34. SGD proceeds via a stochastic approximation of the gradient:

$$\boldsymbol{\theta}^{(s+1)} = \boldsymbol{\theta}^{(s)} - \epsilon_s \frac{1}{n} \sum_{i \in \mathcal{S}_n} \nabla U_i(\boldsymbol{\theta}^{(s)}).$$

This approximation introduces stochasticity into the updates but significantly reduces computational cost.

Two commonly used learning rate (or step size) schedules are $\epsilon_s = s^{-\kappa}$ and

$\epsilon_s = \epsilon_0/(1+s/\tau)^\kappa$, where ϵ_0 is the initial learning rate, τ is a stability constant that slows down early decay (larger values lead to more stable early behavior), and $\kappa \in (0.5, 1]$ controls the long-run decay rate. If κ is too large, the learning rate decays too quickly and the algorithm may stagnate. Conversely, if κ is too small, the algorithm may remain unstable or fail to converge.

An important distinction to note is that SGLD operates with gradient *sums*, while SGD typically uses *averages*. This distinction affects how step sizes and noise scaling should be interpreted in practice.

After convergence, we obtain a reliable estimate of the posterior mode $\hat{\boldsymbol{\theta}}$. Based on this, we define the control variate as $u_i(\boldsymbol{\theta}) = \nabla U_i(\hat{\boldsymbol{\theta}})$. The resulting control variate estimator of the gradient is:

$$\hat{\nabla}_{\text{cv}} U(\boldsymbol{\theta}) = \sum_{i=1}^{N} \nabla U_i(\hat{\boldsymbol{\theta}}) + \frac{N}{n} \sum_{i \in \mathcal{S}_n} \left(\nabla U_i(\boldsymbol{\theta}) - \nabla U_i(\hat{\boldsymbol{\theta}}) \right).$$

This expression is used in place of $\hat{\nabla} U(\boldsymbol{\theta})^{(n)}$ from Equation 12.1 within Algorithm A34.

It has been shown that the computational cost per effective sample size of SGLD algorithm with control variates is $O(1)$, rather than $O(N)$ [294]. However, it is also known that the approximate posterior distribution produced by SGLD, despite capturing the correct mode, tends to exhibit higher variance compared to the true target posterior.

[74] propose the stochastic gradient Hamiltonian Monte Carlo (SGHMC) algorithm, which performs better asymptotically than SGLD in terms of the Wasserstein distance to the target distribution, especially in high-dimensional settings (with large K). Similarly, [261] propose an alternative diffusion process whose stationary distribution is the desired posterior. See Table 1 in [294] for a comprehensive list of stochastic gradient MCMC (SGMCMC) methods that fit in the general diffusion process proposed by [261].

[294] also discuss convergence diagnostics for SGMCMC methods. They argue that standard diagnostics, such as those discussed in Section 4.4, are inadequate for assessing SGMCMC behavior, as these methods target asymptotically biased approximations of the posterior. Consequently, a growing body of research focuses on designing suitable diagnostics for SGMCMC, with approaches based on Stein's discrepancy being among the most prominent (see [294] for details).

A limitation of SGMCMC algorithms is that they are applicable only to a restricted class of models, specifically, those with fixed-dimensional parameter spaces in $\mathcal{R}^K$ and differentiable log-posterior densities. To broaden their applicability, [384] propose an extended stochastic gradient MCMC framework capable of addressing more general large-scale Bayesian inference problems, including those involving dimension-jumping and missing data.

Example: Simulation exercise to study the performance of CMC and SGLD

In this example, we follow the logistic regression simulation setup introduced by [294]:

$$P(y_i = 1 \mid \boldsymbol{\beta}, \mathbf{x}_i) = \frac{\exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}}{1 + \exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}},$$

with log-likelihood function given by:

$$\begin{aligned}\log p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x}) &= \sum_{i=1}^{N} y_i \left(\mathbf{x}_i^\top \boldsymbol{\beta} - \log\left(1 + \exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}\right)\right) + (1 - y_i)\left(-\log\left(1 + \exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}\right)\right) \\ &= \sum_{i=1}^{N} y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \log\left(1 + \exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}\right).\end{aligned}$$

Note that this implies that the gradient vector is

$$\nabla \log p(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{x}) = \sum_{i=1}^{N} \left(y_i - \frac{\exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}}{1 + \exp\left\{\mathbf{x}_i^\top \boldsymbol{\beta}\right\}}\right) \mathbf{x}_i.$$

We assume a prior distribution for $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, 10\mathbf{I}_K)$, leading to the log-prior:

$$\log \pi(\boldsymbol{\beta}) = -\frac{K}{2}\log(2\pi) - \frac{1}{2}\log\left(|10\mathbf{I}_K|\right) - \frac{1}{2}\boldsymbol{\beta}^\top (10^{-1}\mathbf{I}_K)\boldsymbol{\beta}.$$

This means that the gradient vector of the log-prior is

$$\nabla \log \pi(\boldsymbol{\beta}) = -\frac{1}{10}\boldsymbol{\beta}.$$

Also note that

$$\begin{aligned}\pi(\boldsymbol{\beta})^{1/B} &\propto \left\{\exp\left(-\frac{1}{2 \cdot 10}\boldsymbol{\beta}^\top \boldsymbol{\beta}\right)\right\}^{1/B} \\ &= \exp\left(-\frac{1}{2 \cdot 10 \cdot B}\boldsymbol{\beta}^\top \boldsymbol{\beta}\right).\end{aligned}$$

This implies that, when implementing CMC, the prior variance must be scaled by the number of batches B. In other words, each subposterior should use a prior with variance $10 \cdot B$ in this example, so that the product of the B subposteriors reconstructs the correct full posterior.

We set $K = 10$, $\boldsymbol{\beta} = 0.5 \cdot \mathbf{1}_K$, and $N = 10^5$. The covariates $\mathbf{x}_i \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, where the covariance matrix $\boldsymbol{\Sigma}^{(i,j)} = U[-\rho, \rho]^{|i-j|}$ with $\rho = 0.4$, and $\mathbf{1}_K$ denotes a K-dimensional vector of ones.

We run 2,000 MCMC iterations initialized at zero, and 500 as burn-in. We scale the regressors beforehand, as this is generally recommended to improve numerical stability and convergence. The following code simulates the model, and sets the hyperparameters of the algorithms.

R code. Simulation: Logit model

```r
rm(list = ls()); set.seed(10101)
library(mvtnorm); library(MCMCpack)
library(ggplot2); library(dplyr)
library(parallel); library(GGally)

#--- Generate correlated covariates
genCovMat <- function(K, rho = 0.4) {
  Sigma0 <- matrix(1, K, K)
  for (i in 2:K) {
    for (j in 1:(i - 1)) {
      Sigma0[i, j] <- runif(1, -rho, rho)^(i - j)
    }
  }
  Sigma0 <- Sigma0 * t(Sigma0)
  diag(Sigma0) <- 1
  return(Sigma0)
}

#--- Simulate logistic regression data
simulate_logit_data <- function(K, N, beta_true) {
  Sigma0 <- genCovMat(K)
  X <- rmvnorm(N, mean = rep(0, K), sigma = Sigma0)
  linpred <- X %*% beta_true
  p <- 1 / (1 + exp(-linpred))
  y <- rbinom(N, 1, p)
  list(y = y, X = X)
}

#--- Parameters
K <- 10
N <- 100000
beta_true <- rep(0.5, K)
B <- 5
batch_prop <- 0.01
Prior_prec <- 0.1
n_iter <- 2000
burnin <- 500
stepsize <- 1e-4
k_target1 <- 4  # beta5
k_target2 <- 5  # beta5
ks <- k_target1:k_target2
#--- Simulate data
sim_data <- simulate_logit_data(K, N, beta_true)
y <- sim_data$y
X <- scale(sim_data$X)

#--- Run MCMCpack logit
df <- as.data.frame(X)
colnames(df) <- paste0("X", 1:K)
df$y <- y
formula <- as.formula(paste("y ~", paste(colnames(df)[1:K],
    collapse = " + "), "-1"))
posterior_mh <- MCMClogit(formula, data = df, b0 = 0, B0 =
    Prior_prec,
burnin = burnin, mcmc = n_iter)
full_posterior <- as.matrix(posterior_mh)[, 1:K]
```

The following code implements Algorithm A33, running five parallel MCMC chains and combining the resulting subposteriors using three different weighting schemes: equal weights, weights based on marginal variances, and weights based on the full covariance matrices.

R code. Consensus Monte Carlo (CMC): Logit regression

```r
#--- Split data
batch_ids <- split(1:N, sort(rep(1:B, length.out = N)))
#--- Function to run MCMC on a subset
mcmc_batch <- function(batch_index, X, y, n_iter, burnin) {
  ids <- batch_ids[[batch_index]]
  X_b <- X[ids, ]
  y_b <- y[ids]
  mcmc_out <- MCMClogit(y_b ~ X_b - 1, burnin = burnin, mcmc = n_iter, verbose = 0, b0 = 0, B0 = Prior_prec * (1/B))
  return(mcmc_out)
}
#--- Run in parallel
cl <- makeCluster(B)
clusterExport(cl, c("X", "y", "batch_ids", "n_iter", "burnin", "mcmc_batch", "Prior_prec", "B"))
clusterEvalQ(cl, library(MCMCpack))
chains <- parLapply(cl, 1:B, function(b) mcmc_batch(b, X, y, n_iter, burnin))
stopCluster(cl)
# Stack MCMC results
posteriors <- lapply(chains, function(x) x[, 1:K])  # remove intercept if added
# CMC posteriors
equal_cmc <- Reduce("+", posteriors) / B
invvar_cmc <- {
  vars <- lapply(posteriors, function(x) apply(x, 2, var))
  weights <- lapply(vars, function(v) 1 / v)
  weights_sum <- Reduce("+", weights)

  weighted_post <- Reduce("+", Map(function(x, w) sweep(x, 2, w, "*"), posteriors, weights))
  sweep(weighted_post, 2, weights_sum, "/")
}
invmat_cmc <- {
  covs <- lapply(posteriors, cov)                        # Get posterior covariances
  invs <- lapply(covs, solve)                            # Invert each covariance
  weight_sum <- Reduce("+", invs)                        # Total weight matrix

  consensus <- matrix(NA, nrow = n_iter, ncol = K)
  for (i in 1:n_iter) {
    draws <- lapply(posteriors, function(p) matrix(p[i, ], ncol = 1))  # Ensure column matrix
    weighted_sum <- Reduce("+", Map(function(w, d) w %*% d, invs, draws))  # Weighted matrix product
    consensus[i, ] <- as.vector(solve(weight_sum, weighted_sum))  # Solve system and convert to vector
  }
  consensus
}
```

R code. Consensus Monte Carlo (CMC): Logit regression

```
# Combine all for plotting
build_df <- function(mat, method) {
  df <- as.data.frame(mat)
  colnames(df) <- paste0("x", ks)
  df$method <- method
  return(df)
}

df_full <- build_df(full_posterior[,ks], "overall")
df_equal <- build_df(equal_cmc[,ks], "equal")
df_scalar <- build_df(invvar_cmc[,ks], "scalar")
df_matrix <- build_df(invmat_cmc[,ks], "matrix")

df_plot <- rbind(df_full, df_matrix, df_scalar, df_equal)

# Plot
ggpairs(
df_plot,
aes(color = method, fill = method, alpha = 0.4),
upper = list(continuous = GGally::wrap("density", alpha =
    0.4)),
lower = list(continuous = GGally::wrap("density", alpha =
    0.4)),
diag = list(continuous = GGally::wrap("densityDiag", alpha =
     0.4))
)
```

By running the code, you can verify that the computational time of the CMC algorithm is lower than that of the Metropolis–Hastings algorithm. Figure 12.10 shows the posterior distributions of β_4 and β_5. We observe that all three weighting schemes perform reasonably well, yielding posterior modes similar to those obtained from the full-data MCMC algorithm. However, the consensus Monte Carlo (CMC) methods produce more dispersed draws, particularly when using equal weights. In contrast, the weighting schemes based on marginal variances and the full covariance matrices yield comparable and more concentrated posterior distributions.

To implement the SGLD algorithm (Algorithm A34), we set $n = 0.01 \cdot N$, and the step size to 1×10^{-4}. The following code illustrates how to implement the SGLD algorithm.[4]

[4]There is an **R** package called *sgmcmc*, developed by [21], which provides implementations of various stochastic gradient MCMC methods, including SGLD and SGHMC. However, this package depends on version 1 of the *tensorflow* package, while the current version

FIGURE 12.10
Simulation: Posterior distributions from consensus Monte Carlo in logit simulation exercise

In Exercise 8, you are asked to implement the control variate version of SGLD. Begin by running 1,500 SGD iterations to locate the posterior mode. This mode should then be used as the initial value for a subsequent run of 1,000 SGLD iterations.

By running the code, you can verify that the computational time of the SGLD algorithm is lower than that of the Metropolis–Hastings algorithm. Figure 12.11 shows the posterior distributions of the fifth location parameter obtained from SGLD and Metropolis–Hastings. We observe that both modes are centered around the true population value; however, the SGLD distribution exhibits greater dispersion compared to the Metropolis–Hastings distribution.

is 2, and *sgmcmc* has not been updated on CRAN. We attempted to install the package from its GitHub repository using the `devtools::install_github("STOR-i/sgmcmc")` command, but encountered compatibility issues due to conflicting TensorFlow versions.

R code. Stochastic Gradient Langevin Dynamic (SGLD): Logit regression

```
SGLD_step <- function(beta, y, X, stepsize, batch_size, prior_var = 10) {
  N <- nrow(X);   K <- length(beta)
  ids <- sample(1:N, size = batch_size, replace = FALSE)
  grad <- rep(0, K)
  for (i in ids) {
    xi <- X[i, ]
    eta <- sum(xi * beta)
    pi <- 1 / (1 + exp(-eta))
    grad_i <- -(y[i] - pi) * xi
    grad <- grad + grad_i
  }
  grad <- grad / batch_size * N
  grad <- grad + beta / prior_var  # gradient of log-prior
  noise <- rnorm(K, 0, sqrt(stepsize))
  beta_new <- beta - 0.5 * stepsize * grad + noise
  return(beta_new)
}
#--- SGLD algorithm
run_SGLD <- function(y, X, stepsize, batch_prop, n_iter, burnin, beta_init = NULL) {
  N <- nrow(X)
  K <- ncol(X)
  batch_size <- round(batch_prop * N)

  beta_mat <- matrix(0, n_iter + burnin, K)
  beta_mat[1, ] <- if (is.null(beta_init)) rep(0, K) else beta_init

  for (s in 2:(n_iter + burnin)) {
    beta_mat[s, ] <- SGLD_step(beta_mat[s - 1, ], y, X, stepsize, batch_size)
  }
  beta_mat[(burnin + 1):(n_iter + burnin), ]
}
#--- Run SGLD
posterior_sgld <- run_SGLD(y = y, X = X, stepsize, batch_prop, n_iter, burnin)
#--- Compare densities for beta5
df_plot <- data.frame(
value = c(posterior_sgld[, k_target2], posterior_mh[, k_target2]),
method = rep(c("SGLD", "MCMC"), each = n_iter)
)
ggplot(df_plot, aes(x = value, fill = method, color = method)) + geom_density(alpha = 0.4) + geom_vline(xintercept = beta_true[k_target2], linetype = "dashed", color = "black") + labs(title = expression(paste("Posterior density of ", beta[5])), x = expression(beta[5]), y = "Density") + theme_minimal()
```

FIGURE 12.11
Simulation: Posterior distributions from stochastic gradient Langevin Dynamic logit simulation exercise

12.6 Summary

In this chapter, we introduced several Bayesian machine learning methods designed to address the challenges posed by *wide* and *tall* data. However, the field of Bayesian machine learning is rapidly evolving, and the material presented here should be viewed as an introductory overview. Many important topics were not covered but are highly relevant, such as Bayesian neural networks [291] and neural posterior estimation [303, 259, 172]. Other key approaches, such as Variational Bayes, particularly in its stochastic implementations, which are rooted in machine learning and offer scalable solutions for tall data, are introduced in Chapter 14 [408].

12.7 Exercises

1. **Simulation exercise: the Bayesian LASSO continues**

   Program the Gibbs sampler for the Bayesian LASSO from scratch,

assuming a hierarchical structure for the global shrinkage parameter, where both the shape and rate parameters are set to 1. Perform inference using this sampler in the Bayesian LASSO simulation exercise and compare the results with those obtained using the *monomvn* package.

2. [212] employ SSVS to identify the main drivers of civil conflict in the post-Cold War era, considering a set of 35 potential determinants across 175 countries worldwide. We use a subset of their dataset provided in *Conflict.csv*, where the dependent variable is *conflictcw*, a binary indicator of civil conflict. Perform SSVS using the *BoomSpikeSlab* package, specifically the *lm.spike* function, to identify the best subset of models.

3. [402] proposes an SSVS approach for binary response models. Use the dataset *Conflict.csv*, where the dependent variable is *conflictcw*, to perform SSVS using the *BoomSpikeSlab* package, specifically the *logit.spike* function, in order to identify the best subset of models. Compare the results with those obtained in Exercise 2.

4. **Example: Simulation exercise $K > N$**

   Use the simulation setting from the Bayesian LASSO and SSVS examples, but now assume there are 600 inputs. This setup implies that the number of inputs exceeds the sample size. In such a scenario, there is no unique solution to the least squares estimator because the determinant of $\mathbf{W}^\top\mathbf{W}$ is zero. This means the matrix is not invertible, and consequently, standard inference procedures based on the least squares estimator cannot be applied. On the other hand, Bayesian inference in this setup is well-defined because the prior helps regularize the problem, which is a key motivation for these methods.

5. **Simulation exercise: the BART model continues**

   Compute Friedman's partial dependence functions [138] for all variables in the BART model simulation example, and plot the posterior mean along with the 95% credible intervals.

6. [92] presents BART probit for classification. This method can be implemented using the *BART* package through the function *pbart*. Use the file *Conflict.csv*, where the dependent variable is *conflictcw*, to perform BART probit, implementing *k-fold* cross-validation to select the threshold that maximizes the sum of the true positive and true negative rates. Additionally, identify the most important predictors by evaluating different numbers of trees.

7. **Simulation exercise: The Gaussian Process simulation continues**

Simulate the process

$$f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2}) + \mu_i,$$

where $\mu_i \overset{\text{i.i.d.}}{\sim} N(0, 0.1^2)$, $x_{ik} \sim U(0,1)$ for $k = 1, 2$, and the sample size is 500.

Define a grid of 20 evenly spaced values between 0 and 1 for each covariate x_{ik}, and use this grid to perform prediction.

Estimate the hyperparameters of the Gaussian Process by maximizing the log marginal likelihood. Then, use the *km* function from the *DiceKriging* package to fit the Gaussian Process, fixing the noise variance at the value that maximizes the log marginal likelihood.

Finally, use the fitted model to predict the outputs on the grid points, and produce a 3D plot showing the predicted surface along with the training data points.

8. **Simulation exercise: Stochastic gradient MCMC continues**

   Program from scratch the stochastic gradient Langevin dynamic algorithm for the logit simulation exercise implementing the control variate version performing 1,500 stochastic gradient descent iterations to locate the posterior mode, which should be then used as the initial value for 1,000 subsequent MCMC iterations using a step size set to 1×10^{-4}.

9. Perform the simulation according to the model $y_i = 1 - 2x_{i1} + 0.5x_{i2} + \mu_i$, where $\mu_i \sim N(0, 1)$, the sample size is 100,000, and the covariates $\mathbf{x}_i \sim N(\mathbf{0}, \mathbf{I}_2)$. Use 5,000 MCMC iterations and a batch size of 1,000 to implement the SGLD algorithm. Set a learning rate schedule that yields sensible results.

   Assume independent priors $\pi(\boldsymbol{\beta}, \sigma^2) = \pi(\boldsymbol{\beta}) \times \pi(\sigma^2)$, with $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{I}_3)$ and $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$, where $\alpha_0 = \delta_0 = 0.01$.

13

Causal inference

In this chapter, we present some Bayesian methods to perform inference on *causal* effects. The point of departure is *identification conditions* which are the set of assumptions that allow us to express the *estimand* —the causal or statistical quantity of interest— as a unique function of the observable data distribution. These identification conditions are conceptually distinct from the econometric or statistical framework used to perform inference once those conditions are satisfied. In other words, the assumptions necessary for identification do not constrain whether we apply a Frequentist or a Bayesian approach for estimation and inference.

Identification addresses the question: *Can the causal effect be expressed as a function of the observable data distribution under certain assumptions?* Once the causal effect is identified in terms of the observed data distribution, we can proceed with statistical inference using either Frequentist methods or Bayesian methods. These inferential frameworks differ in how they estimate and quantify uncertainty but operate on the same identified causal effect. Thus, the identification assumptions are logically prior and independent of whether we adopt a Frequentist or a Bayesian inferential paradigm.

Thus, we present the underlying identification assumptions employed in popular strategies such as randomized controlled trials, conditional independence, and instrumental variables, among others, as well as the Bayesian inferential framework used to estimate causal effects in these settings.

We should be clear that this is only an introduction to causal inference, and we do not aim to provide an in-depth discussion. There are excellent texts on causal inference, such as [17, 16, 204, 190, 100, 79]. We recommend that readers consult these references for a deeper understanding of some of the concepts and tools introduced in this chapter.

13.1 Identification setting

The aim is to identify the *causal* or *treatment effect* of a particular regressor or treatment on an outcome. For example, this could involve estimating the causal effect of a labor training program on individuals' earnings or unemployment, or the price elasticity of demand. The starting point is the *potential*

outcomes framework [354], which defines a *counterfactual scenario* describing what would happen to the outcome variable if the treatment status or level were different (commonly referred to as the *Rubin causal model* [196]).

In this context, *treatment status* refers to a binary treatment case with two potential outcomes; for instance, what would my earnings be if I had participated in the training program? Conversely, *treatment level* applies to cases where there is a potential outcome for each level of the treatment; for example, what would demand be if the price had increased by 10%?

Following the potential outcomes notation in the binary treatment case, let $D_i = 1$ and $D_i = 0$ indicate the treatment status, corresponding to *treated* and *control* units, respectively. The potential outcomes $Y_i(1)$ and $Y_i(0)$ represent the outcome for unit $i = 1, 2, \ldots, N$ under treatment and control, respectively. For instance, $Y_i(1)$ denotes the employment status an individual would have if she/he would have participated in the training program, regardless of their actual participation, whereas $Y_i(0)$ denotes the employment status if the individual would not have attended the program. The individual-level treatment effect is then defined as

$$\tau_i = Y_i(1) - Y_i(0).$$

The potential outcomes framework can also be extended to situations where the treatment is continuous [201]. In this case, let $Y_i(x_s)$ denote the outcome for unit i under the counterfactual scenario where the treatment variable X_s takes the value x_s. The "treatment effect" of changing X_s from x_s to x'_s, for example, the effect of increasing the price by 10% on demand, is given by

$$\beta_{si} = Y_i(x'_s) - Y_i(x_s).$$

When the change is infinitesimal, the causal effect can be expressed as a marginal effect:

$$\beta_{si} = \frac{\partial Y_i(x_s)}{\partial x_s}.$$

This setting is more complex than the binary treatment case because there is a potential outcome for each possible value of x_s [166]. Therefore, we begin by considering the binary treatment case. However, *the fundamental problem of causal inference* [196] remains: it is impossible to observe the same unit under different *treatment statuses* simultaneously. Consequently, we must learn about causal effects by comparing the outcomes of treated and untreated units.

Bayesian inference for causal effects [355] is more direct than procedures based on Fisher's p-value approach, which relies on the logic of stochastic proof by contradiction, or Neyman's randomization-based inference, which is grounded in the idea of repeated sampling and the construction of confidence intervals for treatment effects [359].

Bayesian inference for causal effects treats the potential outcomes as random variables and involves computing the posterior predictive distribution to evaluate treatments not received, conditional on the observed responses to treatments actually received. This approach yields the posterior distribution of the causal estimands as a function of both the observed outcomes and the unobserved potential outcomes, which are treated as missing data and handled through data augmentation methods [393].

In addition, it is important to note that the likelihood function does not contain information about the correlation between potential outcomes due to *the fundamental problem of causal inference*. Consequently, most of the classical literature on treatment effects has focused on average treatment effects, which require only the marginal distributions of the potential outcomes $Y(1)$ and $Y(0)$ [187]. Although, it is also possible to estimate *quantile treatment effects* [2, 78].

By contrast, a Bayesian inferential approach allows us to estimate *distributional treatment effects* with uncertainty quantification, as formally defined by [1, 187]. These effects characterize the entire distribution of potential outcomes under treatment and control, rather than focusing solely on the average treatment effect, and thus capture heterogeneity. This is particularly relevant for policy, as it enables estimating the proportion of the population that benefits from a social program or the probability that a treated individual experiences a positive effect [328].

Furthermore, under a Bayesian framework, the impact of adding or relaxing identification assumptions can be assessed by examining how the distributional treatment effects change [203].

13.2 Randomized controlled trial (RCT)

Two of the most relevant estimands in the causal inference literature are the average treatment effect (ATE),

$$\text{ATE} = \mathbb{E}[Y_i(1) - Y_i(0)],$$

and the average treatment effect on the treated (ATT),

$$\text{ATT} = \mathbb{E}[Y_i(1) - Y_i(0) \mid D_i = 1].$$

However, these estimands are not directly identifiable without additional assumptions because we never observe the same unit under both treatment states simultaneously. Therefore, identification restrictions are required to express these causal quantities in terms of the observed data.

Note that the observed mean difference between treated and untreated

units can be decomposed as:

$$\mathbb{E}[Y_i \mid D_i = 1] - \mathbb{E}[Y_i \mid D_i = 0] = \underbrace{\mathbb{E}[Y_i(1) - Y_i(0) \mid D_i = 1]}_{\text{ATT}} + \underbrace{\Big(\mathbb{E}[Y_i(0) \mid D_i = 1] - \mathbb{E}[Y_i(0) \mid D_i = 0]\Big)}_{\textit{Selection Bias}},$$

that is, the observed mean difference equals the ATT plus the *selection bias*, which captures the difference between the expected potential outcome of treated individuals had they not been treated (unobserved) and the expected outcome of untreated individuals (observed).

The sign of the selection bias depends on the nature of self-selection into treatment. Two illustrative examples are:

TABLE 13.1
Illustrative examples of selection bias

Scenario	$\mathbb{E}[Y(0) \mid D = 1]$	$\mathbb{E}[Y(0) \mid D = 0]$	Selection Bias
Training Program	80	70	+10
Remedial Education	60	70	−10

In the first scenario, motivated individuals self-select into a training program and would have higher earnings even without participation, producing a positive selection bias and causing the observed mean difference to overestimate the ATT. In the second scenario, disadvantaged individuals are more likely to enroll in a remedial education program and would have lower outcomes without the program, resulting in a negative selection bias and causing the observed mean difference to underestimate the ATT.

Note that if we assume that *assignment to treatment is random*,

$$\{Y_i(1), Y_i(0)\} \perp D_i,$$

then the selection bias becomes zero,

$$\mathbb{E}[Y_i(0) \mid D_i = 1] - \mathbb{E}[Y_i(0) \mid D_i = 0] = 0,$$

which means there is no selection bias under random assignment.

Moreover, under random assignment, ATT equals ATE:

$$\tau = \underbrace{\mathbb{E}[Y_i(1) \mid D_i = 1] - \mathbb{E}[Y_i(0) \mid D_i = 1]}_{\text{ATT}} = \underbrace{\mathbb{E}[Y_i(1) - Y_i(0)]}_{\text{ATE}}.$$

Consequently,

$$\tau = \mathbb{E}[Y_i \mid D_i = 1] - \mathbb{E}[Y_i \mid D_i = 0], \tag{13.1}$$

that is, the average causal effect is function of the data under random assignment to treatment.

Randomized controlled trials (RCTs) are characterized by *random assignment to treatment*. The identification of causal effects in RCTs additionally relies on two key assumptions: *overlap*, which requires that every unit has a positive probability of being assigned to both treatment and control groups ($0 < P(D_i = 1) < 1$), and the *Stable Unit Treatment Value Assumption* (SUTVA). The latter consists of two components: (i) *no interference*, meaning one unit's potential outcome is unaffected by another unit's treatment, and (ii) *consistency*, meaning the observed outcome equals the potential outcome corresponding to the received treatment.

We can represent the causal mechanism in an RCT using causal diagrams [306, 307]. These diagrams provide a powerful framework for visualizing and analyzing the underlying structures that enable causal identification. It is important to note that causal diagrams do not impose specific functional forms, such as those assumed in linear regression models, and are closely related to structural equation models (SEMs). SEMs offer an alternative perspective on causal inference in econometrics, which is closely connected to the potential outcomes approach [176, 270, 201]. See below.

In particular, we can depict the causal mechanism using a *Directed Acyclic Graph (DAG)*, which is a graphical representation of a set of variables and their assumed causal relationships. A DAG consists of *nodes*, representing variables, and *directed edges (arrows)*, representing direct causal effects from one variable to another. The graph is *acyclic*, meaning it contains no directed cycles: starting from any node and following the arrows, it is impossible to return to the same node. A defining property of causal DAGs is that, conditional on its direct causes, any variable on the DAG is independent of any other variable for which it is not a cause (*causal Markov assumption*). See [190] for a nice introduction. Figure 13.1 illustrates the causal structure underlying RCTs.

FIGURE 13.1
Directed Acyclic Graph (DAG) implied by a Randomized Controlled Trial (RCT).

The counterfactual representation of the DAG, known as the *Single-World Intervention Graph* (SWIG), is shown in Figure 13.2. In this figure, the treatment variable D is split to reflect the intervention: we fix D at a specific value d (the intervention), and replace the outcome Y with its counterfactual representation $Y(d)$, which denotes the value of Y if D were set to d. The natural value of D is also shown but becomes irrelevant for the outcome once the intervention is imposed.

Note that we can express the observed outcome in terms of the potential

FIGURE 13.2
Single-World Intervention Graph (SWIG) for intervention $do(D = d)$: Y is replaced by the counterfactual $Y(d)$, and D is split into the fixed value $D = d$ and its natural value D. The dashed arrow indicates the fixed causal assignment.

outcomes as:

$$Y_i = [Y_i(1) - Y_i(0)]D_i + Y_i(0),$$

which shows that the observed outcome equals the control potential outcome plus the treatment effect if treated.

Under the assumption of a constant treatment effect and a linear *Conditional Expectation Function* (CEF) $\mathbb{E}[Y_i \mid D_i]$, this relationship can be represented as a linear regression model [17]:

$$Y_i = \underbrace{\beta_0}_{\mathbb{E}[Y_i(0)]} + \underbrace{\tau}_{\text{constant treatment effect}} D_i + \underbrace{\mu_i}_{Y_i(0)-\mathbb{E}[Y_i(0)]},$$

where τ represents the common treatment effect across all units.

Under random assignment, we have

$$\mathbb{E}[\mu_i \mid D_i] = \mathbb{E}[Y_i(0) \mid D_i] - \mathbb{E}[Y_i(0)] = \mathbb{E}[Y_i(0)] - \mathbb{E}[Y_i(0)] = 0, \tag{13.2}$$

that is, the error term is *mean-independent of treatment status*. This implies that a regression of Y_i on D_i consistently estimates the causal effect under random assignment.

Example: Simple example

Bayesian inference for causal effects in a completely randomized experiment without covariates can be illustrated using the normal model in [358]. Assume that $Y_i(d) \sim N(\mu_d, \sigma_d^2)$ for $d \in \{1, 0\}$. Then, the likelihood function given a random sample and independent treatment assignment is:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{d}; \mu_1, \mu_0, \sigma_1^2, \sigma_0^2) &= \prod_{i:d_i=1} \phi(y_i \mid \mu_1, \sigma_1^2) \prod_{i:d_i=0} \phi(y_i \mid \mu_0, \sigma_0^2) \\ &= \prod_i \phi(y_i \mid \mu_1, \sigma_1^2)^{d_i} \, \phi(y_i \mid \mu_0, \sigma_0^2)^{1-d_i}, \end{aligned}$$

where $\mathbf{y}$ and $\mathbf{d}$ collect the observations of the outcomes and treatments, and $\phi(\cdot)$ denotes the normal density function.

Assume conjugate priors:

$$\mu_d \mid \sigma_d^2 \sim N\left(\mu_{0d}, \frac{\sigma_d^2}{\beta_{0d}}\right), \qquad \sigma_d^2 \sim IG\left(\frac{\alpha_{0d}}{2}, \frac{\delta_{0d}}{2}\right).$$

From the normal/inverse-gamma model in Chapter 3, the posterior conditional distributions are:

$$\mu_d \mid \sigma_d^2, \{y_i, d_i\}_{i:d_i=d} \sim N\left(\mu_{nd}, \frac{\sigma_d^2}{\beta_{nd}}\right),$$

where

$$\mu_{nd} = \frac{\beta_{0d}\mu_{0d} + N_d\bar{y}_d}{\beta_{0d} + N_d}, \qquad \beta_{nd} = \beta_{0d} + N_d,$$

and $\bar{y}_d$, N_d denote the sample mean and sample size of group d. For the variance:

$$\sigma_d^2 \mid \{y_i, d_i\}_{i:d_i=d} \sim IG\left(\frac{\alpha_{nd}}{2}, \frac{\delta_{nd}}{2}\right),$$

where

$$\alpha_{nd} = \alpha_{0d} + N_d, \qquad \delta_{nd} = \sum_{i:d_i=d} (y_i - \bar{y}_d)^2 + \delta_{0d} + \frac{\beta_{0d}N_d}{\beta_{0d} + N_d}(\bar{y}_d - \mu_{0d})^2.$$

In addition, the predictive distribution is given by

$$Y(d) \mid \mathbf{y} \sim t\left(\mu_{nd}, \frac{(\beta_{nd}+1)\delta_{nd}}{\beta_{nd}\alpha_{nd}}, \alpha_{nd}\right).$$

Therefore, given the definition of the average treatment effect,

$$\begin{aligned}
\text{ATE} &= \mathbb{E}[Y(1) - Y(0)] \\
&= \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] \\
&= \int_{\mathcal{R}} y(1) f_{Y(1)}(y(1))dy(1) - \int_{\mathcal{R}} y(0) f_{Y(0)}(y(0))dy(0) \\
&= \mu_1 - \mu_0.
\end{aligned}$$

Note that this expectation is taken with respect to the population distribution of the potential outcomes, not with respect to the posterior distribution.[1] Thus, the posterior distribution of the ATE is given by

$$\pi(\text{ATE} \mid \mathbf{y}) = \pi(\mu_1 - \mu_0 \mid \mathbf{y}).$$

We can obtain this posterior distribution by simulation, because the difference of two independent Student's t-distributed random variables does not

[1] Other functions of the potential outcomes, such as $P(Y(1) \geq Y(0))$, require the joint distribution of the potential outcomes rather than just the marginal distributions, as is the case for the ATE.

follow a Student's t distribution. However, as the degrees of freedom increase, each posterior distribution of μ_d converges to a normal distribution, and the difference of two normals is also normal. Consequently, an approximate point estimate of the ATE is $\mu_{n1} - \mu_{n0}$, which asymptotically equals $\bar{y}_1 - \bar{y}_0$ under non-informative priors. This is the classical *estimator* of the ATE.

Finally, note that we can also obtain the posterior distribution of the ATE using the simple linear regression framework by assuming non-informative prior distributions, in which case the posterior mean of the slope parameter coincides with the maximum likelihood estimator (see Exercise 1).

Let's assume that $\mu_1 = 15$, $\mu_0 = 10$, $\sigma_1^2 = 4$, $\sigma_0^2 = 2$, and $N_1 = N_0 = 100$, and compute the posterior distribution of the ATE. The following code illustrates this procedure, while Figures 13.3 and 13.4 display the predictive densities of the potential outcomes and the posterior distribution of the ATE. The posterior mean is 4.60, and the 95% credible interval is (4.10, 5.12). Note that the population value is $\mu_1 - \mu_0 = 5$.

R code. Simple example: Average treatment effects

```r
set.seed(10101)
# Parameters
mu1 <- 15; mu0 <- 10
sigma1 <- sqrt(4); sigma0 <- sqrt(2)
N1 <- 100; N0 <- 100
# Simulate data
y1 <- rnorm(N1, mu1, sigma1); y0 <- rnorm(N0, mu0, sigma0)
# Prior hyperparameters
alpha0 <- 0.01; delta0 <- 0.01
simulate_t <- function(y) {
  N <- length(y); ybar <- mean(y)
  sse <- sum((y - ybar)^2)
  alpha_n <- alpha0 + N; delta_n <- sse + delta0
  scale2 <- ((N + 1) * delta_n) / (N * alpha_n)
  df <- alpha_n
  loc <- ybar; scale <- sqrt(scale2)
  rt(1, df = df) * scale + loc
}
# Posterior predictive draws
ppd1 <- replicate(1000, simulate_t(y1))
ppd0 <- replicate(1000, simulate_t(y0))
# Plot
hist(ppd1, col = rgb(1, 0, 0, 0.4), freq = FALSE, main = "Posterior Predictive",
xlab = "Y", xlim = c(5, 25))
hist(ppd0, col = rgb(0, 0, 1, 0.4), freq = FALSE, add = TRUE)
legend("topright", legend = c("Treatment", "Control"),
fill = c(rgb(1, 0, 0, 0.4), rgb(0, 0, 1, 0.4)))
# Posterior distribution of ATE
posterior_mu <- function(y) {
  N <- length(y); ybar <- mean(y)
  sse <- sum((y - ybar)^2)
  alpha_n <- alpha0 + N
  delta_n <- sse + delta0
  # Posterior variance for mean
  var_mu <- delta_n / (alpha_n * N)
  df <- alpha_n
  loc <- ybar; scale <- sqrt(var_mu)
  rt(1, df = df) * scale + loc
}
```

R code. Simple example: Average treatment effects

```
# Posterior draws for parameters
n_draws <- 10000
mu1_draws <- replicate(n_draws, posterior_mu(y1))
mu0_draws <- replicate(n_draws, posterior_mu(y0))
# Parameter uncertainty: difference of means
ate_draws <- mu1_draws - mu0_draws
summary(coda::mcmc(ate_draws))
# Summaries
ate_mean <- mean(ate_draws)
ci <- quantile(ate_draws, c(0.025, 0.975))
cat("Posterior mean of ATE:", ate_mean, "\n")
cat("95% Credible Interval:", ci, "\n")

# Plot posterior distribution of ATE
hist(ate_draws, breaks = 50, freq = FALSE,
main = "Posterior Distribution of ATE",
xlab = "ATE (Y(1) - Y(0))", col = "lightblue", border = "
    white")
abline(v = ate_mean, col = "red", lwd = 2)
abline(v = ci, col = "darkgreen", lty = 2, lwd = 2)

legend("topright", legend = c("Posterior Mean", "95%
    Credible Interval"), col = c("red", "darkgreen"), lwd =
    2, lty = c(1, 2), bty = "n", cex = 0.8)  # Smaller
    legend using cex
```

An RCT is considered the gold standard for identifying causal effects because it provides the strongest basis for satisfying the key identification assumption in causal inference: *independence between treatment assignment and potential outcomes*. However, RCTs may face challenges such as *non-compliance*, which occurs when individuals do not adhere to their assigned treatment in an experimental study. This is important because, in many real-world settings, some individuals do not take the treatment they were assigned, leading to deviations from the ideal randomized controlled trial design. In addition, RCTs are sometimes infeasible due to ethical, logistical, or financial constraints. Moreover, they can be too narrowly focused or too localized to provide general conclusions about what works [105].

It is also important to note that RCTs primarily identify the mean of the treatment effect distribution but do not capture other features, such as the median or higher-order moments. These additional aspects of the distribution of treatment effects can be highly relevant for policymakers and stakeholders. See [105] for a detailed discussion of other potential shortcomings of RCTs.

FIGURE 13.3
Predictive distributions: Potential outcomes.

FIGURE 13.4
Posterior distribution: Average treatment effects.

13.3 Conditional independence assumption (CIA)

In practice, researchers often work with *observational data*, where the treatment status is not randomly assigned because units actively choose the treatment they receive. In an RCT, the assignment mechanism is determined by chance, whereas in observational studies it is driven by choice. In these situations, we can identify the causal effect if the *conditional independence assumption* (CIA) holds. This assumption states that the potential outcomes are independent of the treatment status, conditional on a set of observed *pre-treatment variables*:

$$\{Y_i(1), Y_i(0)\} \perp D_i \mid \mathbf{X}_i.$$

This means that, conditional on the *pre-treatment variables* $\mathbf{X}_i$, treatment assignment is as good as random. This property is known as *unconfoundedness given* $\mathbf{X}_i$, or equivalently, the *no unmeasured confounders* assumption. When this condition is combined with the requirement that, for all possible values of $\mathbf{X}_i$, there is a positive probability of receiving each treatment level ($0 < P(D_i = 1 \mid \mathbf{X}_i) < 1$), the joint condition is referred to as *strong ignorability*. Identification of average causal effects in this setting also requires the SUTVA.

Note that under the CIA,

$$\begin{aligned}
\text{ATE} &= \mathbb{E}[Y_i(1)] - \mathbb{E}[Y_i(0)] \\
&= \mathbb{E}_{\mathbf{X}}\left\{\mathbb{E}[Y_i(1) \mid \mathbf{X}_i] - \mathbb{E}[Y_i(0) \mid \mathbf{X}_i]\right\} \\
&= \mathbb{E}_{\mathbf{X}}\left\{\mathbb{E}[Y_i(1) \mid \mathbf{X}_i, D_i = 1] - \mathbb{E}[Y_i(0) \mid \mathbf{X}_i, D_i = 0]\right\} \\
&= \mathbb{E}_{\mathbf{X}}\left\{\mathbb{E}[Y_i \mid \mathbf{X}_i, D_i = 1] - \mathbb{E}[Y_i \mid \mathbf{X}_i, D_i = 0]\right\} \\
&= \mathbb{E}_{\mathbf{X}}\left\{\tau(\mathbf{X}_i)\right\},
\end{aligned}$$

where the second equality follows from the law of iterated expectations, the third uses CIA, and

$$\tau(\mathbf{X}_i) = \mathbb{E}[Y_i \mid \mathbf{X}_i, D_i = 1] - \mathbb{E}[Y_i \mid \mathbf{X}_i, D_i = 0]$$

is the *conditional average treatment effect* (CATE) at covariate value $\mathbf{X}_i$, which captures treatment effect heterogeneity across different values of $\mathbf{X}$. Therefore, the ATE is the expectation of the CATE over the distribution of $\mathbf{X}$.

Figure 13.5 illustrates the causal structure underlying the CIA. We follow the convention that time flows from left to right. Here, $\mathbf{X}$ -a vector of pre-treatment variables or *confounders*- influences both the treatment (D) and the outcome (Y). Under the CIA, we can identify the causal effect of D on Y by adjusting for $\mathbf{X}$ because this causal structure satisfies the *back-door criterion*. This criterion states that a set of variables $\mathbf{X}$ satisfies the condition for identifying the effect of D on Y if no variable in $\mathbf{X}$ is a descendant of D and $\mathbf{X}$ blocks every back-door path from D to Y. A *back-door path* is any path

FIGURE 13.5
Directed Acyclic Graph (DAG) implied by the Conditional Independence Assumption (CIA).

FIGURE 13.6
Single-World Intervention Graph (SWIG) for $do(D = d)$: The treatment D is split into the fixed intervention $D = d$ and its natural value. The outcome is replaced by $Y(d)$. $\mathbf{X}$ still influences $Y(d)$, so adjustment for $\mathbf{X}$ is needed for identification.

from D to Y that begins with an arrow pointing into D. In addition, Figure 13.6 displays the SWIG.

The simple regression framework of the previous section can be extended to a multiple linear regression model by including the *pre-treatment variables*; therefore, the CEF is linear in the treatment and pre-treatment variables:

$$Y_i = \beta_0 + \tau D_i + \mathbf{X}_i^\top \boldsymbol{\beta} + \mu_i.$$

Note that, by assumption in RCTs, D_i and $\mathbf{X}_i$ are independent. Thus, the identification of τ is not affected under random assignment; however, including $\mathbf{X}_i$ helps explain part of the variability in Y_i, thereby improving the precision of the estimates.

On the other hand, if the CIA is satisfied, the error term in the linear regression is defined as

$$\mu_i = Y_i(0) - \mathbb{E}[Y_i(0) \mid \mathbf{X}_i].$$

Thus,

$$\mathbb{E}[\mu_i \mid D_i, \mathbf{X}_i] = \mathbb{E}[Y_i(0) - \mathbb{E}[Y_i(0) \mid \mathbf{X}_i] \mid D_i, \mathbf{X}_i] = \mathbb{E}[Y_i(0) \mid D_i, \mathbf{X}_i] - \mathbb{E}[Y_i(0) \mid \mathbf{X}_i].$$

By the CIA, $Y_i(0) \perp D_i \mid \mathbf{X}_i$, so:

$$\mathbb{E}[Y_i(0) \mid D_i, \mathbf{X}_i] = \mathbb{E}[Y_i(0) \mid \mathbf{X}_i].$$

Therefore:

$$\mathbb{E}[\mu_i \mid D_i, \mathbf{X}_i] = 0. \tag{13.3}$$

This condition is known as *conditional mean independence*: after controlling for $\mathbf{X}_i$, treatment assignment is independent of unobserved determinants of the outcome. It justifies unbiased estimation of τ by regression adjustment in observational studies.

Example: Treatment effect of 401(k) eligibility on net financial assets

We study the average treatment effect of eligibility (*e401*) for participation in the 401(k) retirement savings plan in the United States on net financial assets (*net_tfa*) using the dataset *401k.csv*. The rationale for the exogeneity of 401(k) eligibility is that it becomes exogenous after conditioning on observable characteristics related to job choice, which may correlate with whether a firm offers this retirement plan [79].

Accordingly, we control for the following covariates: age (*age*), income (*inc*), family size (*fsize*), years of education (*educ*), a marital status indicator (*marr*), a two-earner status indicator (*twoearn*), a defined benefit pension status indicator (*db*), an IRA participation indicator (*pira*), and a home ownership indicator (*hown*). Under this specification, the key assumption is that eligibility is conditionally independent of net financial assets given these covariates [79].

We can use the framework in Section 6.1 to estimate the average treatment effect of 401k eligibility on net financial assets. The code 13.3 shows how to get the posterior distribution of this effect. Figure 13.7 displays the posterior distribution of the treatment effect, the 95% credible interval is (USD 3,473, USD 8,371), and the posterior mean is USD 5,903.

R code. Treatment effect: 401(k) eligibility on net financial assets

```r
# 401k: Treatment effects
rm(list = ls())
set.seed(10101)

library(MCMCpack)
library(coda)
library(ggplot2)

mydata <- read.csv("https://raw.githubusercontent.com/BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/401k.csv", sep = ",", header = TRUE, quote = "")
attach(mydata )
y <- net_tfa
# net_tfa: net financial assets
# Regressors quantity including intercept
X <- cbind(e401, age, inc, fsize, educ, marr, twoearn, db, pira, hown, 1)
# e401: 401k eligibility
# age, income, family size, years of education, marital status indicator, two-earner status indicator, defined benefit pension status indicator, IRA participation indicator, and home ownership indicator.

# Posterior distributions using packages: MCMCpack sets the model in terms of the precision matrix. We use default parameters
posterior  <- MCMCpack::MCMCregress(y~X-1)
summary(coda::mcmc(posterior))

# Extract posterior samples for e401 (first coefficient)
beta_e401 <- posterior[, 1]

# Convert to data frame for ggplot
df_posterior <- data.frame(beta_e401)

# Plot with ggplot
ggplot(df_posterior, aes(x = beta_e401)) +
geom_density(fill = "steelblue", alpha = 0.6) +
geom_vline(xintercept = mean(beta_e401), color = "red", linetype = "dashed") +
labs(
title = "Posterior Distribution of Treatment Effect (e401)",
x = expression(beta[e401]),
y = "Density"
) +
theme_minimal(base_size = 14)
```

FIGURE 13.7
Posterior distribution: Treatment effect 401k eligibility on net financial assets.

Observe that the identification strategy relies on conditioning on $\mathbf{X}$. However, adding more controls is not always safe because some variables, known as *bad controls*, can introduce bias if included in the adjustment set [17]. One important type of bad control is a *collider*, a variable that is caused by (or is a common effect of) two or more other variables in a DAG. Colliders play a critical role because conditioning on them can create spurious associations between their causes, leading to what is known as *collider bias* (or selection bias).

Figure 13.8 illustrates this situation. Here, C is a common effect of D and $\mathbf{X}$. Conditioning on C opens an additional path between D and Y, creating a spurious association that violates the back-door criterion. In particular, the variable C is a collider on the path $D \to C \leftarrow \mathbf{X}$. By default, a collider *blocks* the flow of association along the path on which it lies, so this path is closed when C is not conditioned on. However, conditioning on C (or on any descendant of C) opens this path, creating a spurious association between D and $\mathbf{X}$. Since $\mathbf{X}$ also affects Y, this induces bias in estimating the causal effect of D on Y.

Example: Birth-weight "paradox", collider bias

The collider bias illustrated in DAG 13.9, based on [79], reflects the well-known Birth-weight "paradox". This phenomenon arises when conditioning on an intermediate variable (birth weight, B) induces a spurious association between the binary indicator of smoking (S) and infant mortality (Y). In this setting, U represents unobserved factors that affect both birth weight and

FIGURE 13.8
Directed Acyclic Graph (DAG): The additional collider C caused by both D and $\mathbf{X}$ would open a path between D and $\mathbf{X}$, inducing bias.

infant mortality. Conditioning on B creates collider bias because it opens a back-door path through the unobserved factors U (dashed).

FIGURE 13.9
DAG illustrating the birth-weight paradox: S (smoking), B (birth weight), Y (infant mortality), U (other unobserved health factors).

The code in 13.3 illustrates the bias by performing 100 replications, and Figure 13.10 shows the distribution of posterior means across these replications. In this setting, the total effect of smoking on infant mortality is 2, consisting of a direct effect of 1 and an indirect effect through birth weight, also equal to 1.

R code. Collider bias: Birth weight "paradox"

```r
rm(list = ls()); set.seed(10101)
library(MCMCpack); library(dplyr); library(ggplot2)
# Parameters
n <- 1000; true_effect <- 2; replications <- 100
# Store results
results <- data.frame(rep = 1:replications, correct = numeric(replications), biased = numeric(replications))

for (r in 1:replications) {
  # Simulate data under DAG: S -> B -> Y, S -> Y, U -> B, U -> Y
  U <- rnorm(n, 0, 1); S <- rbinom(n, prob = 0.7, size = 1)
  B <- S + U + rnorm(n)
  Y <- S + B + 1.5*U + rnorm(n)
  # Correct model: does NOT condition on collider B
  model_correct <- MCMCregress(Y ~ S, burnin = 1000, mcmc = 3000, verbose = FALSE)
  # Biased model: conditions on collider B
  model_biased <- MCMCregress(Y ~ S + B, burnin = 1000, mcmc = 3000, verbose = FALSE)
  # Posterior means for S
  results$correct[r] <- mean(as.matrix(model_correct)[, "S"])
  results$biased[r] <- mean(as.matrix(model_biased)[, "S"])
}
# Compute bias
results <- results %>% mutate(
bias_correct = correct - true_effect,
bias_biased = biased - true_effect
)
# Average bias and SD
avg_bias <- results %>%
summarise(
mean_correct = mean(bias_correct),
mean_biased = mean(bias_biased),
sd_correct = sd(bias_correct),
sd_biased = sd(bias_biased)
)
print(avg_bias)
# Visualization: distribution of posterior means across 100 simulations
df_long <- results %>%
select(rep, correct, biased) %>%
tidyr::pivot_longer(cols = c(correct, biased), names_to = "model", values_to = "estimate")

ggplot(df_long, aes(x = estimate, fill = model)) + geom_density(alpha = 0.5) + geom_vline(xintercept = true_effect, color = "black", linetype = "dashed", linewidth = 1) + labs(title = "Posterior Means Across 100 Simulations", x = expression(paste("Posterior Mean of ", beta[S])), y = "Density") + theme_minimal(base_size = 14)
```

FIGURE 13.10
Posterior distribution: Posterior means of the collider bias example.

Collider (selection) bias is one possible source of bias in the identification of causal effects. The well-known Heckman's sample selection problem in econometrics [186] can be interpreted as a form of collider bias because restricting the analysis to selected observations (e.g., those with positive wages) conditions on a variable that is a common effect of observed and unobserved factors, thereby opening a non-causal path and creating bias. In other words, the sample is no longer representative of the population, which undermines the identification of the causal effect.

Other well-known sources of bias in econometrics include the omission of common causes affecting both the treatment and the outcome, that is, *omission of correlated relevant regressors*, measurement error in regressors leading to *attenuation bias* (where the estimated causal effect is biased toward zero), and *reverse or simultaneous causality*, which often arises in systems of equations (see [420] for details). We will discuss these additional sources of bias later.

13.4 Instrumental variables (IV)

In many real-world situations, we face biases such as those described in the previous paragraph, arising from measurement errors, omission of relevant variables, and similar issues. Even in these cases, it is still possible to identify

the causal effect. One common strategy is to use a set of *instrumental variables* ($\mathbf{Z}$) that satisfy two key conditions:

1. *Relevance*, meaning the instruments are correlated with the treatment:
$$\mathbf{Z}_i \not\perp D_i.$$
2. *Exogeneity*, which in the linear model requires:
$$\mathbb{E}[\mu_i \mid \mathbf{Z}_i] = 0, \tag{13.4}$$
and, in terms of potential outcomes, entails:
   - *Exclusion restriction*: instruments affect the outcome only through the treatment:
   $$Y_i(D_i = d, \mathbf{Z}_i = \mathbf{z}) = Y_i(D_i = d, \mathbf{Z}_i = \mathbf{z}') = Y_i(D_i = d), \quad d \in \{0, 1\},$$
   - *Marginal exchangeability*: instruments are independent of the potential outcomes:
   $$\mathbf{Z}_i \perp \{Y_i(1), Y_i(0)\}.$$

Relevance is testable, typically by checking whether instruments significantly predict the endogenous variable [387, 99, 230, 391]. However, *weak instruments*, those that exhibit only a weak association with the treatment, can lead to serious consequences, including a high level of uncertainty and the potential amplification of even small biases when estimating the causal effect.

Exogeneity is fundamentally untestable. However, when the number of instruments exceeds the number of endogenous variables, the Sargan test (or its robust version, the Hansen J-test) can be used to assess the validity of overidentifying restrictions. However, it should not be misinterpreted as a direct test of the exclusion restriction. Instead, the test evaluates whether the overidentifying restrictions implied by the IV model hold. Specifically, the null hypothesis states that all instruments are jointly uncorrelated with the structural error term: $\mathbb{E}[\mathbf{Z}^\top \mu] = \mathbf{0}$ [363, 180]. If the test rejects, it indicates that at least one instrument is invalid, but it does not reveal whether the violation results from a failure of the exclusion restriction, correlation with unobserved confounders, or model misspecification. Conversely, if the test fails to reject, it suggests that the sample moment conditions do not provide evidence against instrument validity under the maintained model, but this does not prove that the exclusion restrictions hold. Therefore, while the Sargan test is informative about overall instrument validity, it is not a separate or definitive test of the exclusion restriction [420, 185].

Figure 13.11 illustrates a situation where pre-treatment variables that influence both the treatment and the outcome are partially observed or measured with error. In such cases, an instrument can help identify the causal effect of the treatment on the outcome.

FIGURE 13.11
Directed Acyclic Graph (DAG) showing an instrumental variable Z used to identify the causal effect of D on Y when some confounders $\mathbf{U}$ (dashed) are unobserved or measured with error.

However, the two conditions of the instruments only imply partial identification of the treatment effect [266, 267, 268]. Thus, it is necessary to add a third condition to get point identification of a particular treatment effect using IV. This condition is *monotonicity* which asserts that the instrument moves all units' treatment decisions in the same direction, that is,

$$D_i(1) \geq D_i(0), i = 1, 2, \ldots, N,$$

with equality for at least one unit i, and assuming a binary instrument $Z_i = \{0, 1\}$.

In the case where Z_i denotes the assignment to treatment and D_i the actual treatment status, there are four potential groups: *always-takers* ($D = 1 \mid Z = 1$ and $D = 1 \mid Z = 0$), *never-takers* ($D = 0 \mid Z = 1$ and $D = 0 \mid Z = 0$), *compliers* ($D = 1 \mid Z = 1$ and $D = 0 \mid Z = 0$), and *defiers* ($D = 0 \mid Z = 1$ and $D = 1 \mid Z = 0$). Note that these groups are not identified; for instance, we do not know whether an individual with $Z_i = 1$ and $D_i = 1$ is a complier or an always taker. The monotonicity assumption rules out defiers, meaning that the instrument (assignment to treatment) either does not change treatment status or only increases it.

Therefore, under relevance, exogeneity, monotonicity, and SUTVA, IV methods allow identification of the causal effect for the subgroup of *compliers*, that is, individuals who take the treatment when encouraged by the instrument and do not take it otherwise [202, 14]. This estimand is known as the *Local Average Treatment Effect* (LATE) because it applies to a specific subpopulation rather than the entire population [190]:

$$\tau_{LATE} = \mathbb{E}[Y_i(1) - Y_i(0) \mid D_i(1) = 1, D_i(0) = 0].$$

We define $Y_i(z_i, D_i(z))$ to be the outcome for unit i if exposed to treatment $D_i(z)$ after being assigned to treatment z. Therefore, the *Intention-to-Treat*

(ITT) causal effect of Z_i on Y_i is:

$$\begin{aligned}\mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0))] = &\underbrace{\mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0)) \mid D_i(1) = 1, D_i(0) = 1] \times P(D_i(1) = 1, D_i(0) = 1)}_{always-takers} \\ &+ \underbrace{\mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0)) \mid D_i(1) = 0, D_i(0) = 0] \times P(D_i(1) = 0, D_i(0) = 0)}_{never-takers} \\ &+ \underbrace{\mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0)) \mid D_i(1) = 1, D_i(0) = 0] \times P(D_i(1) = 1, D_i(0) = 0)}_{compliers} \\ &+ \underbrace{\mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0)) \mid D_i(1) = 0, D_i(0) = 1] \times P(D_i(1) = 0, D_i(0) = 1)}_{defiers},\end{aligned}$$

that is, ITT is a mixture over the four potential groups.

We know that the intention-to-treat (ITT) effect of Z_i on Y_i is zero for always-takers and never-takers due to the exclusion restriction, i.e., Z_i affects Y_i only through D_i, and D_i does not vary with Z_i for always-takers and never-takers. Moreover, the monotonicity assumption rules out the existence of defiers. Hence, only compliers contribute to the effect of Z_i on Y_i:

$$\mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0))] = \mathbb{E}[Y_i(1, D_i(1)) - Y_i(0, D_i(0)) \mid D_i(1) = 1, D_i(0) = 0] \cdot P(D_i(1) = 1, D_i(0) = 0).$$

For compliers, $D_i = Z_i$, so we can simplify $Y_i(1, D_i(1)) = Y_i(1, 1) = Y_i(1)$ and $Y_i(0, D_i(0)) = Y_i(0, 0) = Y_i(0)$. Thus,

$$\mathbb{E}[Y_i(1) - Y_i(0)] = \mathbb{E}[Y_i(1) - Y_i(0) \mid D_i(1) = 1, D_i(0) = 0] \cdot P(D_i(1) = 1, D_i(0) = 0),$$

which implies that

$$\tau_{LATE} = \mathbb{E}[Y_i(1) - Y_i(0) \mid D_i(1) = 1, D_i(0) = 0] = \frac{\mathbb{E}[Y_i(1) - Y_i(0)]}{P(D_i(1) = 1, D_i(0) = 0)}.$$

Under the assumption that the instrument Z_i is independent of the potential outcomes (i.e., random assignment), and using the fact that the expectation of a binary variable equals its probability, the LATE can be expressed as:

$$\tau_{LATE} = \frac{\mathbb{E}[Y_i \mid Z_i = 1] - \mathbb{E}[Y_i \mid Z_i = 0]}{\mathbb{E}[D_i \mid Z_i = 1] - \mathbb{E}[D_i \mid Z_i = 0]}.$$

This is the standard instrumental variables (IV) estimand, often referred to as the *Wald estimator*. It represents the ratio of the intention-to-treat effect on the outcome to the intention-to-treat effect on the treatment (i.e., the proportion of compliers). The instrument induces random variation in treatment: the numerator captures how outcomes respond to the instrument, and the denominator captures how treatment responds to the instrument. Dividing the two isolates the causal effect for compliers.

Note that the LATE is not the global average causal effect, and has been subject to several criticisms. First, the proportion of compliers in the population may be small, raising concerns about the policy relevance of this estimand. Observe that while we cannot identify all four principal strata (compliers, never-takers, always-takers, and defiers), the proportion of compliers is identifiable. Second, the monotonicity assumption is not always plausible, particularly in observational studies where instruments may induce heterogeneous behavioral responses. Third, partitioning the population into four strata can become ill-defined in settings where instruments arise from different mechanisms or individual-specific criteria [105, 190]. For arguments in defense of LATE against these critiques, see [15].

Example: Effects of vitamin A supplements on children's survival

This example is taken from [203]. The authors perform Bayesian inference for causal effects in a completely randomized experiment with no covariates and partial compliance. The experiment was conducted in Indonesia, where children were randomly assigned to receive vitamin A supplements. No individual assigned to the control group ($Z_i = 0$) actually took the vitamin, meaning that $D_i(0) = 1$ was never observed. This rules out always-takers and defiers. However, some individuals assigned to the treatment group ($Z_i = 1$) did not take the vitamin, so $D_i(1) = 0$ occurred for some units. Consequently, $D_i(1) \geq D_i(0)$ holds for all individuals (there are not defiers), satisfying the monotonicity assumption. Under this setting, the population consists only of compliers and never-takers.

TABLE 13.2
Effects of vitamin A supplements on children's survival [203]

Type	**Assignment** $Z_{\text{obs},i}$	**Treatment** $D_{\text{obs},i}$	**Survival** $Y_{\text{obs},i}$	**Units** 23,682
Complier or never-taker	0	0	0	74
Complier or never-taker	0	0	1	11,514
Never-taker	1	0	0	34
Never-taker	1	0	1	2,385
Complier	1	1	0	12
Complier	1	1	1	9,663

Table 13.2 summarizes the data. Using this table, we can illustrate key concepts in the identification of causal effects with noncompliance. Let C_i denote the compliance type of individual i, which in this case can be either a complier (c) or a never-taker (n). The probability of being a complier, $\omega :=$

$P(C_i = c)$, is given by

$$\begin{aligned} P(C_i = c) = & \underbrace{P(D_i = 1 \mid Z_i = 1)}_{P(\text{compliers})+P(\text{always-takers})} - \underbrace{P(D_i = 1 \mid Z_i = 0)}_{P(\text{always-takers})=0} \\ = & \frac{P(D_i = 1, Z_i = 1)}{P(Z_i = 1)}. \end{aligned}$$

Thus, the ML estimate is

$$\hat{\omega} = \frac{12 + 9{,}663}{12 + 9{,}663 + 34 + 2{,}385} = 0.8.$$

Note that $P(C_i = n) = 1 - P(C_i = c)$ in this example; thus, the ML estimate is $1 - \hat{\omega} = 0.2$.

The probability of survival for compliers assigned to treatment, $\eta_{c1} := P(Y_i = 1 \mid C_i = c, Z_i = 1)$, is

$$P(Y_i = 1 \mid C_i = c, Z_i = 1) = \frac{P(Y_i = 1, C_i = c \mid Z_i = 1) \times P(Z_i = 1)}{P(C_i = c \mid Z_i = 1) \times P(Z_i = 1)}.$$

Thus, the ML estimate is $\hat{\eta}_{c1} = \frac{9{,}663}{9{,}663+12} = 0.999$.

Similarly, the probability of survival for never-takers assigned to treatment, $\eta_{n1} := P(Y_i = 1 \mid C_i = n, Z_i = 1)$, is

$$P(Y_i = 1 \mid C_i = n, Z_i = 1) = \frac{P(Y_i = 1, C_i = n \mid Z_i = 1) \times P(Z_i = 1)}{P(C_i = n \mid Z_i = 1) \times P(Z_i = 1)}.$$

Consequently, the ML estimate is $\hat{\eta}_{n1} = \frac{2{,}385}{2{,}385+34} = 0.986$.

Note that $\eta_{c0} = P(Y_i = 1 \mid C_i = c, Z_i = 0)$ and $\eta_{n0} = P(Y_i = 1 \mid C_i = n, Z_i = 0)$ cannot be *point-identified*. However, we can obtain *set identification* because

$$\begin{aligned} \eta_0 := P(Y_i = 1 \mid Z_i = 0) &= P(Y_i = 1 \mid Z_i = 0, C_i = c) \times P(C_i = c) \\ &\quad + P(Y_i = 1 \mid Z_i = 0, C_i = n) \times P(C_i = n) \\ &= \frac{P(Y_i = 1, Z_i = 0)}{P(Z_i = 0)}, \end{aligned}$$

where the ML estimate is $\hat{\eta}_0 = \frac{11{,}514}{11{,}514+74} = 0.9936$.

The first equality holds because the probability of survival given no treatment assignment is expressed as a mixture by the law of total probability, and the second equality follows from Bayes' rule.

This implies that

$$0.9936 = \hat{\omega}\hat{\eta}_{c0} + (1 - \hat{\omega})\hat{\eta}_{n0},$$

which leads to

$$\hat{\eta}_{n0} = 4.968 - 4\hat{\eta}_{c0}.$$

Given that probabilities must lie in the unit interval, such that $0 \leq 4.968 - 4\hat{\eta}_{c0} \leq 1$, we obtain: (i) $\hat{\eta}_{c0} \leq 1 \leq 1.242$ (automatically satisfied), and (ii) $4\hat{\eta}_{c0} \geq 4.968 - 1 = 3.968$, which implies $\hat{\eta}_{c0} \geq 0.992$.

Consequently, the ML estimates $\hat{\eta}_{c0} \in [0.992, 1]$ and $\hat{\eta}_{n0} \in [0.968, 1]$.

These results imply that the *complier average causal effect* (CACE) is

$$\begin{aligned}\tau_{CACE} &= \mathbb{E}[Y_i(1) - Y_i(0) \mid D_i(1) = 1, D_i(0) = 0] \\ &= P[Y_i = 1 \mid C_i = c, Z_i = 1] - P[Y_i = 1 \mid C_i = c, Z_i = 0] \\ &= \eta_{c1} - \eta_{c0},\end{aligned}$$

where the first line is the formal definition, the second equality uses random assignment to express the estimand in terms of observed probabilities, and the third relies on the model parameterization.

The CACE is only set-identified, yielding the interval

$$\hat{\tau}_{CACE} \in [-0.001, 0.007],$$

meaning that the survival rate among compliers can vary between 1 and 7 per 1,000 children due to vitamin A supplementation.

Point identification can be achieved by imposing the *weak exclusion restriction* [203], which states that treatment assignment is unrelated to potential outcomes for never-takers and always-takers. Under this assumption,

$$P(Y_i = y \mid C_i = n, Z_i = z) = P(Y_i = y \mid C_i = n),$$

so that $\hat{\eta}_{n1} = \hat{\eta}_{n0} = \hat{\eta}_n = 0.986$. Substituting into the mixture equation yields $\hat{\eta}_{c0} = 0.9955$, and consequently,

$$\hat{\tau}_{CACE} = 0.0035,$$

indicating that survival increases by approximately 3.5 per 1,000 children due to vitamin A supplementation.[2]

We use conjugate families to perform Bayesian inference in this example. Specifically, we adopt the Bernoulli-Beta model (see Section 3.2) with independent non-informative Beta priors, each having parameters equal to 1. This corresponds to a uniform distribution on the interval $(0, 1)$.

An advantage of the Bayesian formulation is that, after applying data augmentation with the compliance type C_i, all the conditional posterior distributions are Beta (See [203] for details in derivations). In particular:

$$\omega \mid \mathbf{C}, \mathbf{Z}_{\text{obs}}, \mathbf{D}_{\text{obs}}, \mathbf{Y}_{\text{obs}} \sim \text{Beta}(1 + N_c,\ 1 + N_n),$$

[2]CACE and LATE refer to the same estimand -the average causal effect for compliers- under monotonicity and the exclusion restriction. If the exclusion restriction fails, they differ because LATE (as identified by IV) no longer equals the complier-specific causal effect (CACE).

where N_c and N_n denote the number of compliers and never-takers, respectively.

$$\eta_{c1} \mid \mathbf{C}, \mathbf{Z}_{\text{obs}}, \mathbf{D}_{\text{obs}}, \mathbf{Y}_{\text{obs}} \sim \text{Beta}(1 + 9{,}663,\ 1 + 12),$$

$$\eta_{n1} \mid \mathbf{C}, \mathbf{Z}_{\text{obs}}, \mathbf{D}_{\text{obs}}, \mathbf{Y}_{\text{obs}} \sim \text{Beta}(1 + 2{,}385,\ 1 + 34),$$

$$\eta_{c0} \mid \mathbf{C}, \mathbf{Z}_{\text{obs}}, \mathbf{D}_{\text{obs}}, \mathbf{Y}_{\text{obs}} \sim \text{Beta}(1 + N_{c01},\ 1 + N_{c00}),$$

where N_{c01} and N_{c00} are the numbers of compliers in the control group who survived and did not survive, respectively.

$$\eta_{n0} \mid \mathbf{C}, \mathbf{Z}_{\text{obs}}, \mathbf{D}_{\text{obs}}, \mathbf{Y}_{\text{obs}} \sim \text{Beta}(1 + N_{n01},\ 1 + N_{n00}),$$

where N_{n01} and N_{n00} are the numbers of never-takers in the control group who survived and did not survive, respectively.

In addition, the conditional probability of being a complier is given by:

$$P(C_i = c \mid Z_{\text{obs},i}, D_{\text{obs},i}, Y_{\text{obs},i}) = \begin{cases} 0, & i \in \{Z_i = 1, D_i = 0\}, \\ 1, & i \in \{Z_i = 1, D_i = 1\}, \\ \dfrac{\omega g_{c0,i}}{\omega g_{c0,i} + (1-\omega) g_{n0,i}}, & i \in \{Z_i = 0\}, \end{cases}$$

where $g_{c0,i} = \eta_{c0}^{Y_{\text{obs},i}}(1-\eta_{c0})^{1-Y_{\text{obs},i}}$ and $g_{n0,i} = \eta_{n0}^{Y_{\text{obs},i}}(1-\eta_{n0})^{1-Y_{\text{obs},i}}$. Note that $Y_{\text{obs},i} = 1$ implies $g_{c0,i} = \eta_{c0}$ and $g_{n0,i} = \eta_{n0}$, while $Y_{\text{obs},i} = 0$ implies $g_{c0,i} = (1-\eta_{c0})$ and $g_{n0,i} = (1-\eta_{n0})$.

The following code shows the implementation, and the corresponding figure displays the posterior distribution of the CACE, the mean is 0.0024, and the 95% credible intervals is (-0.0012, 0.0071).

R code. Effects of vitamin A supplements on children's survival

```
rm(list = ls()); set.seed(10101)
library(dplyr)
# Simulate data
Nc111 <- 9663
c111 <- cbind(rep(1, Nc111), rep(1, Nc111), rep(1, Nc111))
Nc110 <- 12
c110 <- cbind(rep(1, Nc110), rep(1, Nc110), rep(0, Nc110))
Nn101 <- 2385
n101 <- cbind(rep(1, Nn101), rep(0, Nn101), rep(1, Nn101))
Nn100 <- 34
n100 <- cbind(rep(1, Nn100), rep(0, Nn100), rep(0, Nn100))
Ncn001 <- 11514
cn001 <- cbind(rep(0, Ncn001), rep(0, Ncn001), rep(1, Ncn001
    ))
Ncn000 <- 74
cn000 <- cbind(rep(0, Ncn000), rep(0, Ncn000), rep(0, Ncn000
    ))

mydata <- rbind(c111, c110, n101, n100, cn001, cn000)
mydata <- data.frame(Z = mydata[,1], D = mydata[,2], Y =
    mydata[,3])
N <- dim(mydata)[1]
attach(mydata)

# Sampling function C (type)
SampleType <- function(z, d, y, wc, nc0, nn0){
  if(z == 1 & d == 0){
    pc <- 0
  }else{
    if(z == 1 & d == 1){
      pc <- 1
    }else{
      if(y == 1){
        pc <- (wc * nc0) / (wc * nc0 +  (1 - wc) * nn0)
      }else{
        pc <- (wc * (1 - nc0)) / (wc * (1 - nc0) +  (1 - wc)
     * (1 - nn0))
      }
    }
  }
  rbinom(1, 1, prob = pc) # 1: Complier/ 0: Never taker
}
z = 0; d = 0; y = 0; wc = 0.8; nc0 = 0.9; nn0 = 0.05
SampleType(z = z, d = d, y = y, wc = wc, nc0 = nc0, nn0 =
    nn0)
Clat <- sapply(1:N, function(i){SampleType(z = Z[i], d = D[i
    ], y = Y[i], wc = wc, nc0 = nc0, nn0 = nn0)})
```

R code. Effects of vitamin A supplements on children's survival

```
# Gibbs sampler
a0 <- 1; b0 <- 1 # Hyperparameters beta priors
burnin <- 500; S <- 2000; tot <- S + burnin
PosteriorDraws <- matrix(NA, tot, 5)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)

for(s in 1:tot){
  dataLat <- cbind(mydata, Clat)
  Nc011 <- sum(dataLat$Z == 0 & dataLat$Clat == 1 & dataLat$
    Y == 1)
  Nc010 <- sum(dataLat$Z == 0 & dataLat$Clat == 1 & dataLat$
    Y == 0)
  Nn001 <- sum(dataLat$Z == 0 & dataLat$Clat == 0 & dataLat$
    Y == 1)
  Nn000 <- sum(dataLat$Z == 0 & dataLat$Clat == 0 & dataLat$
    Y == 0)
  Nc <- sum(Clat == 1)
  Nn <- sum(Clat == 0)
  wc <- rbeta(1, 1 + Nc, 1 + Nn)
  nc1 <- rbeta(1, 1 + Nc111, 1 + Nc110)
  nn1 <- rbeta(1, 1 + Nn101, 1 + Nn100)
  nc0 <- rbeta(1, 1 + Nc011, 1 + Nc010)
  nn0 <- rbeta(1, 1 + Nn001, 1 + Nn000)
  Clat <- sapply(1:N, function(i){SampleType(z = Z[i], d = D
    [i], y = Y[i], wc = wc, nc0 = nc0, nn0 = nn0)})
  PosteriorDraws[s, ] <- c(wc, nc1, nc0, nn1, nn0)
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
keep <- seq((burnin+1), tot)
LATE <- PosteriorDraws[keep, 2] - PosteriorDraws[keep, 3]
LATEmean <- mean(LATE)
LATEci <- quantile(LATE, c(0.025, 0.975))

# Plot posterior distribution of CATE
hist(LATE, breaks = 40, freq = FALSE,
main = "Posterior Distribution of CACE",
xlab = "CACE", col = "lightblue", border = "white")
abline(v = LATEmean, col = "red", lwd = 2)
abline(v = LATEci, col = "darkgreen", lty = 2, lwd = 2)

legend("topright", legend = c("Posterior Mean", "95%
    Credible Interval"),
col = c("red", "darkgreen"), lwd = 2, lty = c(1, 2),
bty = "n", cex = 0.8)  # Smaller legend using cex
```

FIGURE 13.12
Posterior distribution: Complier average causal effect of vitamin A supplements.

We can perform inference by imposing the weak exclusion restriction, setting $\eta_n = \eta_{n0} = \eta_{n1}$, so that the posterior distribution is

$$\eta_n \mid \mathbf{C}, \mathbf{Z}_{\text{obs}}, \mathbf{D}_{\text{obs}}, \mathbf{Y}_{\text{obs}} \sim \text{Beta}(1 + N_{n1},\, 1 + N_{n0}),$$

where N_{n1} and N_{n0} denote the numbers of never-takers who survived and did not survive, respectively. This is the only modification to the Gibbs sampler.

The following code shows the implementation, and Figure 13.13 displays the posterior distribution of the CACE (which equals the LATE under the exclusion restriction). The posterior mean is 0.003, and the 95% credible interval is (0.0008, 0.0054). Therefore, imposing the exclusion restriction leaves the posterior mean approximately unchanged but increases the precision and informativeness of the posterior distribution. Without the restriction, the posterior was relatively flat within the 95% credible interval, whereas under the exclusion restriction, the posterior distribution is approximately normal, resulting in a narrower 95% credible interval.

Thus, this example illustrates how the Bayesian framework provides a way to quantify uncertainty and conduct sensitivity analysis regarding the exclusion restriction. See [192] for an extension controlling for covariates.

R code. Effects of vitamin A supplements on children's survival

```
PosteriorDrawsER <- matrix(NA, tot, 4)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)

for(s in 1:tot){
  dataLat <- cbind(mydata, Clat)
  Nc011 <- sum(dataLat$Z == 0 & dataLat$Clat == 1 & dataLat$
    Y == 1)
  Nc010 <- sum(dataLat$Z == 0 & dataLat$Clat == 1 & dataLat$
    Y == 0)
  Nn01 <- sum(dataLat$Clat == 0 & dataLat$Y == 1)
  Nn00 <- sum(dataLat$Clat == 0 & dataLat$Y == 0)
  Nc <- sum(Clat == 1)
  Nn <- sum(Clat == 0)
  wc <- rbeta(1, 1 + Nc, 1 + Nn)
  nc1 <- rbeta(1, 1 + Nc111, 1 + Nc110)
  nn <- rbeta(1, 1 + Nn01, 1 + Nn00)
  nc0 <- rbeta(1, 1 + Nc011, 1 + Nc010)
  Clat <- sapply(1:N, function(i){SampleType(z = Z[i], d = D
    [i], y = Y[i], wc = wc, nc0 = nc0, nn0 = nn)})
  PosteriorDrawsER[s, ] <- c(wc, nc1, nc0, nn)
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
keep <- seq((burnin+1), tot)
LATEER <- PosteriorDrawsER[keep, 2] - PosteriorDrawsER[keep,
     3]
LATEERmean <- mean(LATEER)
LATEERci <- quantile(LATEER, c(0.025, 0.975))
# Plot posterior distribution of ATE
hist(LATE, breaks = 40, freq = FALSE,
main = "Posterior Distribution of CACE",
xlab = "CACE", col = "lightblue", border = "white")
abline(v = LATEmean, col = "red", lwd = 2)
abline(v = LATEci, col = "darkgreen", lty = 2, lwd = 2)
legend("topright", legend = c("Posterior Mean", "95%
    Credible Interval"),
col = c("red", "darkgreen"), lwd = 2, lty = c(1, 2),
bty = "n", cex = 0.8)  # Smaller legend using cex
```

Example: Treatment effect of 401(k) participation on net financial assets

In the example of the effect of *eligibility* on 401(k) net financial assets, we calculate the intention-to-treat (ITT) effect. However, following [77], we

FIGURE 13.13
Posterior distribution: Local average average atreatment effect vitamin A supplements.

use eligibility as an instrument for *participation* and perform inference on the (local) average treatment effect. We adopt the framework described in Section 7.3 for this example. The following code illustrates the procedure, and Figure 13.14 displays the posterior distribution of the (local) average treatment effect of participation. The 95% credible interval is (USD 5,102, USD 12,010), and the posterior mean is USD 8,520, which is higher than the intention-to-treat effect associated with eligibility (USD 5,903).

Exercise 3 asks how to recover the ITT from the LATE and the effect of eligibility on participation. See [96] for practical methods to conduct sensitivity analysis of posterior results when relaxing the exclusion restriction in IV settings. In addition, [97] present a Bayesian inferential framework using a semi-parametric approach in which stochastic errors are modeled using a Dirichlet process, and [332] extend this approach to systems of equations.

R code. Treatment effect: 401(k) participation on net financial assets

```
rm(list = ls()); set.seed(10101)
library(coda); library(ggplot2)
mydata <- read.csv("https://raw.githubusercontent.com/BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/401k.csv", sep = ",", header = TRUE, quote = "")
# Attach variables
attach(mydata)
y <- net_tfa/1000  # Outcome: net financial assets
x <- as.vector(p401) # Endogenous regressor: participation
w <- as.matrix(cbind(1, age, inc, fsize, educ, marr, twoearn, db, pira, hown))  # Exogenous regressors with intercept
z <- as.matrix(e401)  # Instrument: eligibility (NO intercept here)
X <- cbind(x, w); Z <- cbind(z, w)
# Dimensions
k <- ncol(X); kz <- ncol(Z)
# Priors
b0 <- rep(0, k); B0i <- diag(1e-5, k)
g0 <- rep(0, kz); G0i <- diag(1e-5, kz)
nu <- 3; Psi0 <- nu * 1000 * diag(2); Psi0i <- solve(Psi0)
# MCMC parameters
mcmc <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 1
# Auxiliary elements
XtX <- t(X)%*%X; ZtZ <- t(Z)%*%Z; nun <- nu + length(y)
# Gibbs sampling
PostBeta <- function(Sigma, Gamma){
  w1 <- Sigma[1,1] - Sigma[1,2]^2/Sigma[2,2]
  Bn <- solve(w1^(-1)*XtX + B0i)
  yaux <- y - (Sigma[1,2]/Sigma[2,2])*(x - Z%*%Gamma)
  bn <- Bn%*%(B0i%*%b0 + w1^(-1)*t(X)%*%yaux)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostGamma <- function(Sigma, Beta){
  w2 <- Sigma[2,2] - Sigma[1,2]^2/Sigma[1,1]
  Gn <- solve(w2^(-1)*ZtZ + G0i)
  xaux <- x - (Sigma[1,2]/Sigma[1,1])*(y - X%*%Beta)
  gn <- Gn%*%(G0i%*%g0 + w2^(-1)*t(Z)%*%xaux)
  Gamma <- MASS::mvrnorm(1, gn, Gn)
  return(Gamma)
}
PostSigma <- function(Beta, Gamma){
  Uy <- y - X%*%Beta; Ux <- x - Z%*%Gamma
  U <- cbind(Uy, Ux)
  Psin <- solve(Psi0i + t(U)%*%U)
  Sigmai <- rWishart::rWishart(1, df = nun, Sigma = Psin)
  Sigma <- solve(Sigmai[,,1])
  return(Sigma)
}

```

R code. Treatment effect: 401(k) participation on net financial assets

```
PostBetas <- matrix(0, tot, k)
PostGammas <- matrix(0, tot, kz)
PostSigmas <- matrix(0, tot, 2*(2+1)/2)
Beta <- rep(0, k); Gamma <- rep(0, kz)
pb <- winProgressBar(title = "progress bar", min = 0, max =
    tot, width = 300)
for(s in 1:tot){
  Sigma <- PostSigma(Beta = Beta, Gamma = Gamma)
  Beta <- PostBeta(Sigma = Sigma, Gamma = Gamma)
  Gamma <- PostGamma(Sigma = Sigma, Beta = Beta)
  PostBetas[s,] <- Beta
  PostGammas[s,] <- Gamma
  PostSigmas[s,] <- matrixcalc::vech(Sigma)
  setWinProgressBar(pb, s, title=paste( round(s/tot*100, 0),
    "% done"))
}
close(pb)
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]
Gs <- PostGammas[keep,]
Sigmas <- PostSigmas[keep,]
summary(coda::mcmc(Bs))
summary(coda::mcmc(Gs))
summary(coda::mcmc(Sigmas))
# Extract posterior draws for the treatment effect (
    participation = p401)
beta_draws <- Bs[,1]
# Plot posterior distribution of treatment effect
df_beta <- data.frame(effect = as.vector(beta_draws))
ggplot(df_beta, aes(x = effect)) + geom_density(fill = "
    steelblue", alpha = 0.6) +
geom_vline(xintercept = mean(beta_draws), color = "red",
    linetype = "dashed", linewidth = 1) + labs( title = "
    Posterior Distribution of 401(k) Participation Effect",
    x = expression(beta["p401"]), y = "Density") + theme_
    minimal(base_size = 14)
```

13.5 Difference-in-differences (DiD)

Consider a situation with two time periods, $t = 1, 2$, and two groups: a treated group ($D_i = 1$) that receives the treatment between periods $t = 1$ and $t = 2$,

FIGURE 13.14
Posterior distribution: Local average treatment effect 401k participation on net financial assets.

and a control group that never receives the treatment ($D_i = 0$). We observe the outcomes Y_{it} and treatment status D_i for $i = 1, 2, \ldots, N$. Assume a balanced panel data setting.

Let $Y_{it}(0,1)$ denote the potential outcome for unit i in period t if it is not treated in the first period but treated in the second period, whereas $Y_{it}(0,0)$ denotes the potential outcome when it is never treated. Thus, the observed outcome is

$$Y_{it} = Y_{it}(0,1)D_i + Y_{it}(0,0)(1 - D_i).$$

The estimand of primary interest in difference-in-differences (DiD) is the average treatment effect on the treated (ATT),

$$\tau_{ATT} = \mathbb{E}[Y_{it}(0,1) - Y_{it}(0,0) \mid D_i = 1].$$

The key identification assumption in DiD is the *parallel trends* assumption,

$$\mathbb{E}[Y_{i2}(0,0) - Y_{i1}(0,0) \mid D_i = 1] = \mathbb{E}[Y_{i2}(0,0) - Y_{i1}(0,0) \mid D_i = 0],$$

which states that, in the absence of treatment, the average change in outcomes for the treated and control populations would have evolved in the same way.

In addition, identification requires *no anticipation effects*,

$$Y_{i1}(0,0) = Y_{i1}(0,1), \text{for all } D_i = 1.$$

This is, the treatment does not have causal effect prior to its implementation.

[299, 300, 58]

In this framework, there is a model for the assignment mechanism $P(D_i \mid Y_i(1), Y_i(0), \mathbf{X}_i)$, like randomization or propensity score matching [347], supplemented by a model for the data $P(Y_i(1), Y_i(0) \mid \mathbf{X}_i)$. A causal inference is delivered by conditioning in what is observed to calculate the posterior distribution of the causal estimand given also the models for the assignment mechanisms and data. Therefore, there is a model to input the potential outcome missing value that allows to get the predictive distribution, and then, the posterior predictive distribution of the treatment effect $(Y_i(1) - Y_i(0))$ is calculated. We should take into account that the parameters of the models for predicting the missing potential outcomes are generally not causal effects. Due to using models for the data based on simulation, the Bayesian approach is by far the most direct and flexible of the modes of inference for causal effects. However, it relies on assumptions on the assignment mechanism and data.

Given pretreatment variables, it improves the predictive power of the potential outcome missing data

There, no compliance is explicitly treated as missing data. A nice advantage of the Bayesian approach is to relax the exclusion restriction [359].

Propensity matching should be used to create "assigned treatment" and "assigned control" groups that are well balanced on all covariates. The covariate modeling should be applied to estimate the effects of "received treatment" for the subgroup of true compliers [359].

Simultaneous causality is conceptually challenging because causality is often associated with chronological order, cause preceding effect. In simultaneous systems, such as supply and demand, outcomes are jointly determined in equilibrium, so this temporal order is not explicit in the observed data. Nevertheless, the underlying structural equations still encode causal relationships, even though their effects are realized simultaneously.

In general, association does not imply causation, although causation does imply association. Two variables can be associated for several reasons: one may cause the other, they may share common causes, or they may share a common effect when the analysis is restricted to a specific level of that effect (or one of its descendants) [190]. For this reason, causal graphs are valuable tools for explicitly representing the assumptions about the causal relationships under study and should be among the first steps in establishing an identification strategy.

Introduce maximum entropy using identification conditions to build likelihood functions.

Instrumental variables change the level of the treatment, without affecting the potential outcomes associated with these treatment levels [201].

The gold standard in identification of causal effects is *randomized controlled trial (RCT)*, where assignment to treatment is random, independent of the outcome and other potential exogenous determinants of the outcome. However, most of the time practitioners use *observational data*, where the treatment status/level is *endogenous* due to units actively defining the treat-

ment that they receive [201]. Thus the assignment mechanism in RCTs are given by chance, whereas in observational studies by choice. This makes a substantial methodological difference such that the latter implies looking a source of *exogenous variation* that influences the *treatment status/level*.

Two critical aspects in the identification of causal effects are: (i) the presence of a strong source of *exogenous variation* that influences the *endogenous regressors*, which are often the primary variables of interest to researchers, as they may directly affect the outcome or response variables and can be influenced by policy decisions, for example, identifying the causal effects of social programs on income or education, or evaluating strategic interventions in industry, such as estimating the price elasticity of demand for a specific product; and (ii) the effective *control of other relevant exogenous covariates*, such as pre-treatment characteristics or external factors in a demand model.

In this context, the use of *non-parametric models* (see Chapters 13 and 12) is valuable due to their flexibility and weaker structural assumptions. Therefore, non-parametric and machine learning approaches serve as *powerful tools* that can be combined with strong exogenous variation to robustly identify causal effects [76, 79].

Read this reference [199] before begin working in this chapter! Also read [203].

13.6 Sample selection

[173]

13.7 Regression discontinuity design

[88, 87, 235]

13.8 Regression kink design

[72]

13.9 Synthetic control

[6, 229]

13.10 Event Analysis

13.11 Bayesian exponential tilted empirical likelihood

Bayesian parametric approaches are often criticized on the basis that they require arbitrary distribution assumptions which often are not examined. Partial information approaches are based only on certain moment assumptions without making specific distributional assumptions. However, there are no free lunch, as these methods imply efficiency losses.

The point of departure of Bayesian exponential tilted empirical likelihood (BETEL) are moment conditions that are used to build the likelihood function.

13.12 A general framework for updating belief distributions

Introduce [43] as a generalization of [80].

13.13 Bayesian model averaging

We can use BMA as a sensible way to perform robustness analysis regarding model specification (regressors uncertainty) in performing inference of treatment effects.

13.14 Summary

13.15 Exercises

1. Show that the Average Treatment Effect (ATE) in the simple linear regression framework

$$Y_i = \beta_0 + \tau D_i + \mu_i,$$

assuming non-informative prior distributions, so that the posterior mean of the location parameter coincides with the maximum likelihood estimator, is equal to

$$\bar{y}_1 - \bar{y}_0.$$

2. Some readers may question the assumption that potential outcomes are normally distributed. However, it is important to note that the normal distribution is the *maximum entropy* continuous distribution given a specified mean μ and finite variance σ^2. In other words, among all distributions with the same mean and variance, the normal distribution represents the one with the greatest level of uncertainty or unpredictability [98].

   Show that the normal distribution is the *maximum entropy* continuous distribution given a specified mean μ and finite variance σ^2 by considering the formal definition of entropy:

$$H(f) = -\int_{-\infty}^{\infty} f(y) \log f(y)\, dy,$$

   where $f(y)$ is a probability density function.

3. **401(k) participation on net financial assets continues I**

   Apply the framework from this example to compute the intention-to-treat effect, the local average treatment effect, and the effect of eligibility on participation.

4. **401(k) participation on net financial assets continues II**

   Use the function *rivDP* from the *bayesm* package to perform inference on 401(k) participation and its effect on net financial assets using the same specification as in the main text, and plot the LATE.

14

Approximate Bayesian methods

Approximate Bayesian methods are a family of techniques designed to handle situations where the likelihood function lacks an analytical expression, is highly complex, or the problem is high-dimensional, whether due to a large parameter space or a massive dataset [273]. In the former case, traditional Markov Chain Monte Carlo (MCMC) and importance sampling algorithms fail to provide a solution. In the latter, these algorithms struggle to produce accurate estimates within a reasonable timeframe, unless users modify them (see Chapter 12).

However, there is no free lunch, *Approximate Bayesian methods* address these challenges at the cost of providing an approximation to the posterior distribution rather than the *exact* posterior. Nonetheless, asymptotic results show that the approximation improves as the sample size increases.

In this chapter, I first present *simulation-based approaches*, which are designed to address situations where the likelihood is highly complex and may lack an analytical solution. In the second part, I introduce *optimization approaches*, which are intended to handle high-dimensional problems. Specifically, I discuss approximate Bayesian computation (ABC) and Bayesian synthetic likelihood (BSL), the two most common *simulation-based approaches*. Then, I present integrated nested Laplace approximations (INLA) and *variational Bayes* (VB), the two most common *optimization approaches* for high-dimensional problems.

14.1 Simulation-based approaches

Taking into account the fundamental equation for performing parameter inference in the Bayesian framework,

$$\pi(\boldsymbol{\theta} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \boldsymbol{\theta}) \times \pi(\boldsymbol{\theta}),$$

we see in Section 4.1 that MCMC algorithms, such as the Gibbs sampler (Section 4.1.1) and Metropolis-Hastings (Section 4.1.2), require evaluation of the likelihood function $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ in the posterior conditional distribution or the acceptance probability, respectively. This is also the case for importance sampling when calculating the importance weights (Section 4.2).

Thus, what happens when the likelihood function does not have an analytical expression? This situation arises in many models involving unobserved heterogeneity (i.e., unobserved taste preferences), models defined by quantile functions (e.g., the g-and-k distribution), or dynamic equilibrium models (e.g., repeated game models).

Simulation-based algorithms provide a Bayesian solution when we face this situation, namely, when the likelihood function lacks an analytical expression or is highly complex. The only requirement is that we must be able to simulate synthetic data from the model conditional on the parameters. Therefore, these algorithms obtain an approximation to the posterior draws by simulating from the prior distribution $\pi(\boldsymbol{\theta})$ and then using these draws to simulate from the likelihood $p(\mathbf{y} \mid \boldsymbol{\theta})$.

14.1.1 Approximate Bayesian computation

Approximate Bayesian Computation (ABC) is designed to handle inferential situations where the likelihood function $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ is intractable or highly complex, $\boldsymbol{y} \in \mathbb{R}^N$. It was introduced in population genetics by [394, 315] and later generalized by [31]. The basic intuitive origin of ABC appears to have been introduced by [356]. A growing body of literature explores its applications in biology, cosmology, finance, economics, and other fields.

The requirement in ABC is the ability to simulate from the parametric model. The process begins by drawing samples from the prior distribution $\pi(\boldsymbol{\theta})$ multiple times, $\boldsymbol{\theta} \in \mathbb{R}^K$, and then simulating data from the model given each $\boldsymbol{\theta}^{(s)}, s = 1, 2, \ldots, S$. The resulting synthetic data, $\boldsymbol{z}^{(s)} \in \mathbb{R}^N$ is used to compute summary statistics $\boldsymbol{\eta}(\boldsymbol{z}^{(s)}) \in \mathbb{R}^L, L \geq K$. These summary statistics are crucial for the performance of ABC and should be selected based on a thorough understanding of the model.

Next, we compare the synthetic summary statistics with the observed summary statistics $\boldsymbol{\eta}(\boldsymbol{y})$ using a distance metric $d\left\{\boldsymbol{\eta}(\boldsymbol{y}), \boldsymbol{\eta}(\boldsymbol{z}^{(s)})\right\}$, typically the Euclidean distance. We retain the prior draws that generate synthetic summary statistics closest to the observed ones, that is, $d\left\{\boldsymbol{\eta}(\boldsymbol{y}), \boldsymbol{\eta}(\boldsymbol{z}^{(s)})\right\} \leq \epsilon$, forming an approximation of the posterior distribution $\pi_\epsilon(\boldsymbol{\theta}, \boldsymbol{z} \mid \boldsymbol{\eta}(\boldsymbol{y}))$.

The simplest algorithm is the accept/reject approximate Bayesian computation (ABC-AR) (see Algorithm A35).

Algorithm A35 Accept/reject ABC

```
for s = 1, ..., S do
    Draw θ^s from π(θ),
    Simulate z^s = (z_1^s, z_2^s, ..., z_n^s)^⊤ from the model, p(·|θ^s)
    Calculate d_(s) = d{η(y), η(z^s)}
end for
Order the distances d_(1) ≤ ··· ≤ d_(S)
Select all θ^s such that d_(i) ≤ ϵ, where ϵ > 0 is the tolerance level.
```

Note that the posterior distribution is conditional on the summary statistics $\boldsymbol{\eta}(\boldsymbol{y})$ and the tolerance parameter ϵ. This implies that we obtain an approximation to the target distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$, that is $\pi(\boldsymbol{\theta} \mid \boldsymbol{\eta}(\boldsymbol{y}))$, because $\boldsymbol{\eta}(\boldsymbol{y})$ is not a sufficient statistic in most cases, and $\epsilon > 0$, these conditions introduce bias [48]. However, ABC performs well compared to full-likelihood approaches in low-dimensional parameter spaces [31].

Furthermore, [134] show in Theorems 1 and 2 that Bayesian consistency and asymptotic normality hold, provided that $\epsilon \to 0$ fast enough as $N \to +\infty$. In particular, the requirement is that the proportion of accepted draws converges to 0 at a rate faster than $N^{-K/2}$. Additionally, Theorem 2 in [134] shows that $100(1-\alpha)\%$ Bayesian credible regions using ABC have frequentist coverage of $100(1-\alpha)\%$.

We should note from these asymptotic results that ABC suffers from the *curse of dimensionality*. Specifically, given a sample size of 1,000 and two parameters, the proportion of accepted draws should be 0.1%, meaning we would require one million prior draws to obtain 1,000 posterior draws. On the other hand, if the number of parameters is three, we would require 31.62 million prior draws. This limitation of ABC has attracted attention; see Chapter 8 of [379] for some potential solutions.

It is a common practice in ABC to perform a regression adjustment after retaining the draws [31, 248, 379]. This adjustment reduces bias in posterior draws by performing a simple linear regression between the selected draws and the discrepancy between the observed and simulated summary statistics, $\theta_k^{(s)} = \alpha_k + \left(\boldsymbol{\eta}(\boldsymbol{y}) - \boldsymbol{\eta}(\boldsymbol{z}^{(s)})\right)^\top \boldsymbol{\beta}_k + \mu_k^{(s)},\ k = 1, 2, \ldots, K$. Then, the posterior draws are adjusted using the slope estimate $\theta_k^{\mathrm{adj},(s)} = \theta_k^{(s)} - \left(\boldsymbol{\eta}(\boldsymbol{y}) - \boldsymbol{\eta}(\boldsymbol{z}^{(s)})\right)^\top \hat{\boldsymbol{\beta}}_k$. Other regression adjustment strategies are also used, such as local linear regression, ridge regression, and neural networks. See the *abc* package in **R**.

The favorable asymptotic sampling properties of ABC rely on correct model specification. [135] demonstrate that when the assumed model is misspecified, the asymptotic behavior of ABC can deteriorate. In particular, the posterior shape becomes asymptotically non-Gaussian, and the behavior of the posterior mean remains generally unknown. Additionally, regression adjustment approaches can produce posteriors that differ significantly from their simpler accept/reject counterparts.

Given these concerns, testing model specification in ABC is essential. This can be done using simulated goodness-of-fit statistics [40, 256], predictive p-values [40], discrepancy diagnostics [135], and asymptotic tests [331] to evaluate model adequacy.

The accept/reject ABC algorithm is inefficient, as all draws are independent; thus, there is no learning from previous draws. This intensifies the computational burden. Therefore, [269, 413] introduced Markov Chain Monte Carlo ABC (ABC-MCMC) algorithms, and [380, 117, 107, 246] proposed sequential Monte Carlo approaches (ABC-SMC). However, results comparing

ABC-MCMC and ABC-SMC with ABC-AR are controversial regarding computational efficiency [40]. In addition, ABC-AR is very simple and easily allows parallel computing [133]. Nevertheless, ABC-SMC is now the recommended approach, as it does not require tuning the algorithm's tolerance [273], and there are open-source implementations that facilitate its use.

New developments in ABC have focused on using empirical measures calculated from the observed ($\hat{\mu}_n$) and synthetic ($\hat{\mu}_{\boldsymbol{\theta}}^{(s)}$) data to replace summary statistics. Thus, $d\left\{\boldsymbol{\eta}(\boldsymbol{y}), \boldsymbol{\eta}(\boldsymbol{z}^{(s)})\right\}$ is replaced by $\mathcal{D}\left\{\hat{\mu}_n, \hat{\mu}_{\boldsymbol{\theta}}^{(s)}\right\}$, where the latter is a discrepancy measure, such as the Kullback-Leibler divergence [214]. However, [116] found in their simulation exercises that the best-performing summary statistics approach performs at least as well as the best discrepancy-measure approaches. The key point is to select informative summary statistics.

Example: g-and-k distribution for financial returns

The g-and-k distribution is a highly flexible distribution capable of capturing skewness and heavy tails through its parameters. This makes it particularly useful for modeling real-world data that deviate from normality, especially in fields like finance, where outliers are common. However, this distribution lacks a closed-form expression for its density function.

The g-and-k distribution is defined by its quantile function [118]. Specifically, it is specified through its inverse cumulative distribution function,

$$Q(p \mid \theta) = F^{-1}(p \mid \theta),$$

where $F = P(U \leq u)$, and Q represents the p-quantile [337]. The quantile function of the g-and-k distribution is given by

$$Q^{gk}\left\{z(p) \mid a, b, c, g, k\right\} = a + b\left[1 + c\frac{1 - \exp\left\{-gz(p)\right\}}{1 + \exp\left\{-gz(p)\right\}}\right]\left\{1 + z(p)^2\right\}^k z(p),$$

where $z(p)$ is the standard normal quantile function, and $c = 0.8$ is a commonly suggested value.

In the g-and-k distribution, a is the location parameter, and b is the scale parameter, controlling the dispersion. The parameters g and k determine the levels of skewness and kurtosis, respectively, while c modifies the impact of skewness, and is typically set to 0.8.

[118] propose a moving average of order one using a g-and-k distribution to model exchange rate log returns. In particular,

$$z_t = \epsilon_t + \theta_1 \epsilon_{t-1}, \quad t = 1, \ldots, 524,$$

where $\epsilon_t \sim N(0, 1)$.

The values of z_t are then divided by $(1 + \theta_1^2)^{1/2}$ to ensure that they marginally follow a standard normal distribution. Thus, simulating g-and-k data requires only substituting z_t into the quantile function.

We model exchange rate log daily returns from USD/EUR one year before and after the WHO declared the COVID-19 pandemic on 11 March 2020. We use the dataset *ExchangeRate.csv* from our GitHub repository. Our ABC implementation uses twelve summary statistics: the seven octiles, the interquartile range, robust measures of skewness and kurtosis, and the autocorrelations of order one and two (see [118] and code below). We adopt the prior distributions proposed by [331],

$$\theta_1 \sim U(-1,1),\ a \sim U(0,5)\ b \sim U(0,5)$$
$$g \sim U(-5,5),\ k \sim U(-0.5,5).$$

We use the *EasyABC* package in **R** to implement the ABC accept/reject (ABC-AR) algorithm using 150,000 prior draws with an acceptance rate of 0.67%. We also apply the ABC Markov chain Monte Carlo (ABC-MCMC) method [269] and the sequential Monte Carlo ABC (ABC-SMC) method [246] to compare the results across different ABC algorithms.[1] We generate 100,000 samples and retain 1% in ABC-MCMC, and 30,000 samples, keeping 3.4%, with a stopping criterion of 5% in ABC-SMC. These settings imply that the three algorithms require approximately the same computational time. As ABC is a simulation-based method, the computational burden is mostly driven by the speed of simulating the process; thus, the *EasyABC* package has parallel computing algorithms to speed up the processes. Users can refer to the cited references for algorithmic details, and the *EasyABC* package for parallel computing implementation.

In Exercise 1, we ask to program the ABC accept/reject algorithm (Algorithm A35) from scratch and compare the results with those obtained using the ABC-AR implementation in the *EasyABC* package.

The following code presents the results, and Figure 14.1 compares the posterior distributions of θ_1, g, and k using the three methods. In this figure, we observe that ABC-MCMC (red) and ABC-SMC (green) exhibit similar performance, and both approaches provide more information than ABC-AR (blue). There is marginal positive evidence that the moving average coefficient is positive, and the three algorithms yield similar means, although ABC-AR exhibits lower precision. Since the posterior distribution of g is centered around zero, the distribution appears to be symmetric around its median. Meanwhile, a positive k indicates that the distribution has heavier tails than a normal distribution, implying a higher likelihood of extreme values (outliers) in the exchange rate USD/EURO.

[1]Note that this setting does not satisfy the asymptotic requirements for Bayesian consistency. However, it serves as a pedagogical exercise.

R code. Exchange rate log returns: Approximate Bayesian computation

```
######### ABC Exchange rate og returns: USD/EURO
rm(list = ls()); set.seed(010101)
library(EasyABC)
dfExcRate <- read.csv(file = "https://raw.githubusercontent.com/BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/ExchangeRate.csv", sep = ",", header = T)
attach(dfExcRate); n <- length(USDEUR)
# Summary statistics
SumSt <- function(y) {
  Oct <- quantile(y, c(0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875))
  eta1 <- Oct[6] - Oct[2]
  eta2 <- (Oct[6] + Oct[2] - 2 * Oct[4]) / eta1
  eta3 <- (Oct[7] - Oct[5] + Oct[3] - Oct[1]) / eta1
  autocor <- acf(y, lag = 2, plot = FALSE)
  autocor[["acf"]][2:3]
  Etay <- c(Oct, eta1, eta2, eta3, autocor[["acf"]][2:3])
  return(Etay)
}
# g-and-k distribution
RGKnewSum <- function(par) {
  z <- NULL
  theta <- par[1]; a <- par[2]; b <- par[3]; g <- par[4]; k <- par[5]
  e <- rnorm(n + 1)
  for(t in 2:(n + 1)){
    zt <- e[t] + theta * e[t-1]
    z <- c(z, zt)
  }
  zs <- z / (1 + theta^2)^0.5
  x <- a + b * (1 + 0.8 * (1 - exp(-g * zs)) / (1 + exp(-g * zs))) * (1 + zs^2)^k * zs
  Etaz <- SumSt(x)
  return(Etaz)
}
toy_prior <- list(c("unif",-1,1), c("unif",0,5), c("unif",0,5), c("unif", -5,5), c("unif", -0.5,5))
sum_stat_obs <- SumSt(USDEUR)
tick <- Sys.time()
ABC_AR <- ABC_rejection(model=RGKnewSum, prior=toy_prior, summary_stat_target = sum_stat_obs, nb_simul=150000, tol = 0.0067,
progress_bar = TRUE)
tock <- Sys.time()
tock - tick
PostABCAR <- coda::mcmc(ABC_AR$param)
summary(PostABCAR)
tick <- Sys.time()
ABC_MCMC <- ABC_mcmc(method="Marjoram", model=RGKnewSum, prior=toy_prior, summary_stat_target=sum_stat_obs, n_rec = 100000, progress_bar = TRUE)
tock <- Sys.time()
tock - tick
PostABCMCMC <- coda::mcmc(ABC_MCMC[["param"]][order(ABC_MCMC[["dist"]])[1:1000],])
summary(PostABCMCMC)
tick <- Sys.time()
```

R code. Exchange rate log returns: Approximate Bayesian computation

```
ABC_SMC<-ABC_sequential(method="Lenormand", model=RGKnewSum,
     prior=toy_prior, summary_stat_target=sum_stat_obs, nb_
    simul = 30000, alpha = 0.034, p_acc_min = 0.05,
progress_bar = TRUE)
summary(PostABCSMC)
tock <- Sys.time()
tock - tick
PostABCSMC <- coda::mcmc(ABC_SMC[["param"]])
# Figures
library(ggplot2); library(latex2exp)
Sp <- 1000

df1 <- data.frame(Value = c(PostABCAR[1:Sp,1], PostABCMCMC
    [1:Sp,1], PostABCSMC[1:Sp,1]),
Distribution = factor(c(rep("AR", Sp), rep("MCMC", Sp), rep(
    "SMC", Sp))))
dentheta <- ggplot(df1, aes(x = Value, color = Distribution)
    ) +   geom_density(linewidth = 1) + labs(title = TeX("
    Posterior density plot: $theta$"), x = TeX("$theta$"), y
     = "Posterior density") + scale_color_manual(values = c(
    "blue", "red", "green")) +  theme_minimal() + theme(
    legend.title = element_blank())

df2 <- data.frame(Value = c(PostABCAR[1:Sp,4], PostABCMCMC
    [1:Sp,4], PostABCSMC[1:Sp,4]),
Distribution = factor(c(rep("AR", Sp), rep("MCMC", Sp), rep(
    "SMC", Sp))))
deng <- ggplot(df2, aes(x = Value, color = Distribution)) +
      geom_density(linewidth = 1) + labs(title = "Posterior
    density plot: g", x = "g", y = "Posterior density") +
scale_color_manual(values = c("blue", "red", "green")) +
    theme_minimal() + theme(legend.title = element_blank())

df3 <- data.frame(Value = c(PostABCAR[1:Sp,5], PostABCMCMC
    [1:Sp,5], PostABCSMC[1:Sp,5]),
Distribution = factor(c(rep("AR", Sp), rep("MCMC", Sp), rep(
    "SMC", Sp))))
denk <- ggplot(df3, aes(x = Value, color = Distribution)) +
      geom_density(linewidth = 1) + labs(title = "Posterior
    density plot: k", x = "k", y = "Posterior density") +
scale_color_manual(values = c("blue", "red", "green")) +
    theme_minimal() + theme(legend.title = element_blank())

library(ggpubr)
ggarrange(dentheta, deng, denk, labels = c("A", "B", "C"),
    ncol = 3, nrow = 1,
legend = "bottom", common.legend = TRUE)
```

FIGURE 14.1
Posterior distributions using approximate Bayesian computation: Exchange rate log returns USD/EURO.

14.1.2 Bayesian synthetic likelihood

Note that in ABC, in most cases, we target the posterior distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{\eta}(\boldsymbol{y}))$ rather than $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$, as $\boldsymbol{\eta}(\boldsymbol{y})$ is not a sufficient statistic, $\boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{\theta} \in \mathbb{R}^K$, $\boldsymbol{\eta}(\boldsymbol{y}) \in \mathbb{R}^L$, $L \geq K$. This can be beneficial since discarding information may improve the behavior of the likelihood or make the inference more robust to model misspecification [314]. Given the intractability of $p(\boldsymbol{y} \mid \boldsymbol{\theta})$, it is highly likely that $p(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta})$ is also intractable. [418] addressed this issue by introducing an auxiliary model for the summary statistics, assuming $p_a(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta}) = N(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$.

Bayesian synthetic likelihood (BSL) arises when this auxiliary likelihood is combined with a prior distribution on the parameter [119, 314],

$$\pi_a(\boldsymbol{\theta} \mid \boldsymbol{\eta}(\boldsymbol{y})) \propto p_a(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}),$$

where the subscript a indicates that this is an approximation due to the Gaussian assumption. This is referred to as the idealized BSL posterior. However, note that $p_a(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta})$ is rarely available, as $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ are generally unknown. Therefore, we estimate these quantities using simulations from the

model given a realization of $\boldsymbol{\theta}$:[2]

$$\widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\eta}(\boldsymbol{z}^{(m)}),$$
$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\theta}} = \frac{1}{M-1}\sum_{m=1}^{M} (\boldsymbol{\eta}(\boldsymbol{z}^{(m)}) - \widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}})(\boldsymbol{\eta}(\boldsymbol{z}^{(m)}) - \widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}})^{\top}.$$

Then, we have

$$\pi_{a,M}(\boldsymbol{\theta} \mid \boldsymbol{\eta}(\boldsymbol{y})) \propto p_{a,M}(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}),$$

where $p_{a,M}(\boldsymbol{\eta}(\boldsymbol{y}))$ uses the estimates, which in turn depends on the number of draws M to calculate $\widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\theta}}$. Note that although we can have unbiased estimators of these object, in general, $N(\boldsymbol{\eta}(\boldsymbol{y}) \mid \widehat{\boldsymbol{\mu}}_{\boldsymbol{\theta}}, \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\theta}})$ is not an unbiased estimator of $N(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. [9] show an unbiased estimator for BSL.

[131] show that $\pi_{a,M}(\boldsymbol{\theta})$ converges asymptotically to a Gaussian distribution and that the $100(1-\alpha)\%$ Bayesian credible regions using BSL have frequentist coverage of $100(1-\alpha)\%$. In addition, the posterior mean from BSL is also asymptotically Gaussian. These results require convenient estimation of the covariance matrix and that $M \to \infty$ as $N \to \infty$, $M = C\lfloor N^{\gamma} \rfloor$, $C > 0$, $\gamma > 0$ and $\lfloor x \rfloor$ denoting the integer floor of x. Thus, the choice of M does not drastically affect the desirable asymptotic properties of BSL [131]. In addition, [314] find in their examples that posterior inference depends only weakly on M. Thus, M can be chosen to balance computational efficiency such that the standard deviation of the log synthetic likelihood is between 1 and 3 [10].

A critical aspect of BSL is the estimation of the covariance matrix, which can be computationally demanding in high-dimensional settings. However, [131] propose an adjusted approach to BSL that allows the use of a simple, though potentially misspecified, estimator of the covariance matrix (see Equation 5 and the related discussion in their paper for details).

If the normality assumption for summary statistics is too restrictive, [8] propose a robust BSL method based on a semi-parametric approach. Additionally, [132] show that when the model is misspecified, i.e., there is no compatibility between the assumed statistical model and the true data-generating process, BSL can lead to unreliable parameter inference. To address this issue, they propose a new BSL method that detects model misspecification and provides more reliable inference.

We can perform BSL using the Algorithm A36. We can use a random walk Metropolis-Hastings to set the proposal distribution. The covariance matrix of the random walk proposal can be tuned using an initial pilot run.

An advantage of BSL over ABC is that it does not require selecting a tolerance parameter ϵ. Furthermore, BSL is more computationally efficient

[2]There are better ways to calculate the covariance matrix, see [131].

Algorithm A36 Bayesian synthetic likelihood

for $s = 1, \ldots, S$ **do**
  Draw $\boldsymbol{\theta}^c \sim q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{s-1})$
  for $m = 1, \ldots, M$ **do**
    Simulate $\boldsymbol{z}^m = (z_1^m, z_2^m, ..., z_n^m)^\top$ from the model, $p(\cdot \mid \boldsymbol{\theta}^c)$
    Calculate $\boldsymbol{\eta}(\boldsymbol{z}^{(m)})$
  end for
  Calculate $\boldsymbol{\mu}_{\boldsymbol{\theta}^c}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}^c}$
  Compute $p_a^c(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta}^c) = N(\boldsymbol{\mu}_{\boldsymbol{\theta}^c}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}^c})$ and $p_a^{s-1}(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta}^{s-1}) = N(\boldsymbol{\mu}_{\boldsymbol{\theta}^{s-1}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}^{s-1}})$
  Compute the acceptance probability

$$\alpha(\boldsymbol{\theta}^{s-1}, \boldsymbol{\theta}^c) = \min\left\{1, \frac{p_a^c(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c)q(\boldsymbol{\theta}^{s-1} \mid \boldsymbol{\theta}^c)}{p_a^{s-1}(\boldsymbol{\eta}(\boldsymbol{y}) \mid \boldsymbol{\theta}^{s-1})\pi(\boldsymbol{\theta}^{s-1})q(\boldsymbol{\theta}^c \mid \boldsymbol{\theta}^{s-1})}\right\}$$

  Draw u from $U(0, 1)$
  if $u < \alpha$ **then**
    Set $\boldsymbol{\theta}^s = \boldsymbol{\theta}^c$, $\boldsymbol{\mu}_{\boldsymbol{\theta}^s} = \boldsymbol{\mu}_{\boldsymbol{\theta}^c}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}^s} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}^c}$
  else
    Set $\boldsymbol{\theta}^s = \boldsymbol{\theta}^{s-1}$, $\boldsymbol{\mu}_{\boldsymbol{\theta}^s} = \boldsymbol{\mu}_{\boldsymbol{\theta}^{s-1}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}^s} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}^{s-1}}$
  end if
end for

than ABC when dealing with a high-dimensional vector of summary statistics as the acceptance rate of the former is asymptotically non-vanishing [131].

On the other hand, ABC is asymptotically more efficient than BSL, and it imposes very weak requirements on the choice of summary statistics, whereas BSL requires summary statistics that satisfy central limit theorems (CLTs), and consistent estimators of the covariance matrix. However, asymptotically, both approaches provide reliable inference, provided their respective requirements are met [273].

In practice, BSL may be more convenient than ABC when the summary statistics are high-dimensional and satisfy CLT conditions. Otherwise, ABC may be the better alternative.

Example: Simulation exercise

We simulate a dataset following the specification of the g-and-k distribution for the financial returns example, setting $\theta_1 = 0.8$, $a = 1$, $b = 0.5$, $g = -1$, and $k = 1$, with a sample size of 500. We use the same priors and summary statistics as in that example.

We use the *BSL* package in **R** to perform posterior inference using BSL. Additionally, we implement our own BSL sampler to compare the results with the vanilla algorithm from the package. We run the BSL algorithms with $M = 200$, $S = 11,000$, a burn-in of 1,000, and a thinning parameter of 10,

using a random walk proposal distribution. This setting results in similar computational times between the two implementations (scratch and package) and is also comparable to the ABC implementation in Exercise 1. Take into account that BSL is a simulation-based method; therefore, the computational burden is mostly driven by the speed of simulating the process. The *BSL* package has parallel computing algorithms to speed up the processes, and users should refer to the *BSL* package for details.

The following code demonstrates this procedure. The summary statistics appear to be approximately normally distributed, as shown in Figure 14.2, where the red line represents the normal density and the black line represents the estimated density of the summary statistic. Figure 14.3 displays the posterior distributions from both algorithms for θ_1, g, and k. In general, the 95% credible intervals encompass the true parameter values. However, the posterior draws from the *BSL* package are more informative compared to our implementation of Algorithm A36. Additionally, BSL outperforms the results of ABC in Exercise 1, yielding more precise posterior distributions. This pattern has been observed in other settings [116, 273]. Both ABC and BSL produce posterior distributions centered at the true parameter values, although the posterior distribution of θ_1 deviates from this pattern.

R code. g-and-k simulation: Bayesian synthetic likelihood

```
######## BSL: g-and-k simulation ############
rm(list = ls()); set.seed(010101); library(BSL)
# Simulate g-and-k data
RGKnew <- function(par) {
  z <- NULL
  theta <- par[1]; a <- par[2]; b <- par[3]; g <- par[4]; k <- par[5]
  e <- rnorm(n + 1)
  for(t in 2:(n + 1)){
    zt <- e[t] + theta * e[t-1]
    z <- c(z, zt)
  }
  zs <- z / (1 + theta^2)^0.5
  x <- a + b * (1 + 0.8 * (1 - exp(-g * zs)) / (1 + exp(-g * zs))) * (1 + zs^2)^k * zs
  return(x)
}
# Summary statistics
SumSt <- function(y) {
  Oct <- quantile(y, c(0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875))
  eta1 <- Oct[6] - Oct[2]
  eta2 <- (Oct[6] + Oct[2] - 2 * Oct[4]) / eta1
  eta3 <- (Oct[7] - Oct[5] + Oct[3] - Oct[1]) / eta1
  autocor <- acf(y, lag = 2, plot = FALSE)
  autocor[["acf"]][2:3]
  Etay <- c(Oct, eta1, eta2, eta3, autocor[["acf"]][2:3])
  return(Etay)
}
# Prior function
LogPrior <- function(par){
  LogPi <- log(par[1] > -1 & par[1] < 1 & par[2] > 0 & par[2] < 5 & par[3] > 0 & par[3] < 5 & par[4] > -5 & par[4] < 5 & par[5] > -0.5 & par[5] < 5)
  return(LogPi)
}
# Population parameters
theta1 <- 0.8; a <- 1; b <- 0.5; g <- -1; k <- 1
parpop <- c(theta1, a, b, g, k)
K <- 5
n <- 500
y <- RGKnew(par = parpop)
# Algorithm parameters
M <- 200 # Number of iterations to calculate mu and sigma
S <- 11000 # Number of MCMC iterations
burnin <- 1000 # Burn in iterations
thin <- 10 # Thining parameter
keep <- seq(burnin + 1, S, thin)
par0 <- c(0.5, 2, 1, 0, 1)
Modelgk <- newModel(fnSim = RGKnew, fnSum = SumSt, theta0 = par0, fnLogPrior = LogPrior, verbose = FALSE)
validObject(Modelgk)
# Check if the summary statistics are roughly normal
simgk <- simulation(Modelgk, n = M, theta = par0, seed = 10)
par(mfrow = c(4, 3))
```

R code. g-and-k simulation: Bayesian synthetic likelihood

```
for (i in 1:12){
  eval <- seq(min(simgk$ssx[, i]), max(simgk$ssx[, i]), 0.001)
  densnorm <- dnorm(eval, mean = mean(simgk$ssx[, i]), sd(simgk$ssx[, i]))
  plot(density(simgk$ssx[, i]), main = "", xlab = "")
  lines(eval, densnorm, col = "red")
}
##### Program from scratch #######
Lims <- matrix(c(-1, 0, 0, -5, -0.5, 1, rep(5, 4)), 5, 2)
parPost <- matrix(NA, S, K); parPost[1,] <- par0 ; EtaY <- SumSt(y = y)
Zsl <- replicate(M, RGKnew(par = parPost[1,]))
EtasZsl <- t(apply(Zsl, 2, SumSt))
Usl <- colMeans(EtasZsl); SIGMAsl <- var(EtasZsl)
dnormsl <- mvtnorm::dmvnorm(EtaY, Usl, SIGMAsl, log = TRUE)
tune <- 0.005; CoVarRW <- tune*diag(K)
accept <- rep(0, S); tick1 <- Sys.time()
pb <- winProgressBar(title = "progress bar", min = 0, max = S, width = 300)
for (s in 2:S){
  parc <- MASS::mvrnorm(1, mu = parPost[s-1,], Sigma = CoVarRW)
  RestCheck <- NULL
  for(j in 1:K){
    if(parc[j] < Lims[j,1] | parc[j] > Lims[j,2]){
      Rej <- 1
    }else{Rej <- 0}
    RestCheck <- c(RestCheck, Rej)
  }
  if(sum(RestCheck) != 0){
    parPost[s,] <- parPost[s-1,]; accept[s] <- 0
  }else{
    Z <- replicate(M, RGKnew(par = parc))
    EtasZ <- t(apply(Z, 2, SumSt))
    Um <- colMeans(EtasZ); SIGMAm <- var(EtasZ)
    dnormc <- mvtnorm::dmvnorm(EtaY, Um, SIGMAm, log = TRUE)
    if(s > round(0.1*S, 0) & s < round(0.2*S, 0)){
      CoVarRW <- var(parPost, na.rm = TRUE)
    }
    if(dnormc == -Inf){
      parPost[s,] <- parPost[s-1,]; accept[s] <- 0
    }else{
      alpha <- min(1, exp(dnormc - dnormsl)); U <- runif(1)
      if(U <= alpha){
        parPost[s,] <- parc; Usl <- Um; SIGMAsl <- SIGMAm
        dnormsl <- dnormc; accept[s] <- 1
      }else{
        parPost[s,] <- parPost[s-1,]; accept[s] <- 0
      }
    }
  }
  setWinProgressBar(pb, s, title=paste( round(s/S*100, 0),"% done"))
}
```

R code. g-and-k simulation: Bayesian synthetic likelihood

```
close(pb); tock1 <- Sys.time(); tock1 - tick1
mean(accept)
PostChainOwn <- coda::mcmc(parPost[keep,])
summary(PostChainOwn)
#### BSL package
tick <- Sys.time()
Resultsgk <- bsl(y = y, n = M, M = S, model = Modelgk, covRandWalk = CoVarRW,
method = "BSL", thetaNames = expression(theta, a, b, g, k), plotOnTheFly = TRUE)
tock <- Sys.time(); tock - tick
PostChain <- coda::mcmc(Resultsgk@theta[keep,])
summary(PostChain)
Resultsgk@acceptanceRate
plot(Resultsgk@loglike[keep], type = "l")
sd(Resultsgk@loglike[keep])
# Figures
library(ggplot2); library(latex2exp)
Sp <- length(keep)
df1 <- data.frame(Value = c(PostChain[,1], PostChainOwn[,1]), Distribution = factor(c(rep("BSL", Sp), rep("BSLscratch", Sp))))
dentheta <- ggplot(df1, aes(x = Value, color = Distribution)) + geom_density(linewidth = 1) + labs(title = TeX("Posterior density plot: $theta$"), x = TeX("$theta$"), y = "Posterior density") + geom_vline(xintercept = theta1, linetype = "dashed", color = "red", linewidth = 1) + scale_color_manual(values = c("blue", "red")) +  theme_minimal() +
theme(legend.title = element_blank())
df2 <- data.frame(Value = c(PostChain[,4], PostChainOwn[,4]), Distribution = factor(c(rep("BSL", Sp), rep("BSLscratch", Sp))))
deng <- ggplot(df2, aes(x = Value, color = Distribution)) + geom_density(linewidth = 1) + labs(title = "Posterior density plot: g", x = "g", y = "Posterior density") +
geom_vline(xintercept = g, linetype = "dashed", color = "red", linewidth = 1) +
scale_color_manual(values = c("blue", "red")) + theme_minimal() + theme(legend.title = element_blank())
df3 <- data.frame(Value = c(PostChain[,5], PostChainOwn[,5]), Distribution = factor(c(rep("BSL", Sp), rep("BSLscratch", Sp))))
denk <- ggplot(df3, aes(x = Value, color = Distribution)) + geom_density(linewidth = 1) + labs(title = "Posterior density plot: k", x = "k", y = "Posterior density") + geom_vline(xintercept = k, linetype = "dashed", color = "red", linewidth = 1) + scale_color_manual(values = c("blue", "red")) +  theme_minimal() + theme(legend.title = element_blank())
library(ggpubr)
ggarrange(dentheta, deng, denk, labels = c("A", "B", "C"), ncol = 3, nrow = 1,
legend = "bottom", common.legend = TRUE)
```

FIGURE 14.2
Density estimates of summary statistics: g-and-k simulation.

FIGURE 14.3
Posterior distributions using Bayesian synthetic likelihood: g-and-k simulation.

14.2 Optimization approaches

Traditional MCMC and importance sampling (IS) algorithms require pointwise evaluation of the likelihood function, which entails a massive number of operations when applied to very large datasets. Unfortunately, these algorithms are not designed to be *scalable*, at least in their standard form (see Chapter 12 for alternatives). Moreover, when the parameter space is large, they also lack *scalability* with respect to the number of parameters. Therefore, approximation methods should be considered even when the likelihood function has an analytical expression.

Optimization approaches are designed to scale efficiently with high-dimensional parameter spaces and large datasets. The key idea is to replace simulation with optimization. In this section, we introduce the most common optimization approaches within the Bayesian inferential framework.

14.2.1 Integrated nested Laplace approximations

Integrated Nested Laplace Approximations (INLA) is a deterministic approach for Bayesian inference in latent Gaussian models (LGMs) [361]. In particular, INLA approximates the marginal posterior distributions using a combination of Laplace approximations, low-dimensional deterministic integration and optimization steps in sparse covariance settings [362]. The advantages of INLA compared to MCMC are that it is fast and does not suffer from poor mixing.

The point of departure is a structured additive regression model, where the response variable y_i belongs to the exponential family such that the mean μ_i is linked to a linear predictor η_i through the link function $g(\cdot)$, that is, $\eta_i = g(\mu_i)$, where

$$\mu_i = \alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i + \sum_{j=1}^{J} f^{(j)}(u_{ji}) + \epsilon_i,$$

where α is the general intercept, $f^{(j)}$ are unknown functions of the covariates $\boldsymbol{u}_i$, $\boldsymbol{\beta}$ is a K-dimensional vector of linear effects associated with regressors $\boldsymbol{x}_i$, and ϵ_i is the unstructured error.

Note that latent Gaussian models encompass a wide range of relevant empirical models depending on the specific elements and structure involved in $f^{(j)}$, such as generalized linear models with unobserved heterogeneity, spatial and/or temporal dependence, and semi-parametric models.

Latent Gaussian models assign a Gaussian prior to α, $\boldsymbol{\beta}$, $f^{(j)}$, and $(\epsilon_i$. Let $\boldsymbol{z}$ denote the vector of all latent Gaussian variables $\left\{\alpha, \boldsymbol{\beta}, f^{(j)}, \eta_i\right\}$, where the dimension is potentially $P = 1 + K + J + N$ (although this is not always the case), and let $\boldsymbol{\theta}$ be the m-dimensional vector of hyperparameters. Then, the density $\pi(\boldsymbol{z} \mid \boldsymbol{\theta}_1)$ is Gaussian with mean zero and precision matrix (i.e., the inverse of the covariance matrix) $\boldsymbol{Q}(\boldsymbol{\theta}_1)$.

The distribution of $\boldsymbol{y}$ is $p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}_2)$ such that y_i are conditional independent given $z_i, \boldsymbol{\theta}_2$, $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top \ \boldsymbol{\theta}_2^\top]^\top$, $i = 1, 2, \ldots, N$. Thus, the posterior distribution is

$$\begin{aligned}\pi(\boldsymbol{z}, \boldsymbol{\theta} \mid \boldsymbol{y}) &\propto \pi(\boldsymbol{\theta}) \times \pi(\boldsymbol{z} \mid \boldsymbol{\theta}_1) \times \prod_{i=1}^{N} p(y_i \mid \boldsymbol{z}, \boldsymbol{\theta}_2) \\ &\propto \pi(\boldsymbol{\theta}) |\boldsymbol{Q}(\boldsymbol{\theta}_1)|^{1/2} \exp\left\{ -\frac{1}{2} \boldsymbol{z}^\top \boldsymbol{Q}(\boldsymbol{\theta}_1) \boldsymbol{z} + \sum_{i=1}^{N} \log p(y_i \mid z_i, \boldsymbol{\theta}_2) \right\}.\end{aligned} \tag{14.1}$$

Most models in INLA assume a conditional independence structure within the high-dimensional latent Gaussian field; that is, $\boldsymbol{z}$ is a Gaussian Markov random field (GMRF) with a sparse precision matrix $\boldsymbol{Q}(\boldsymbol{\theta}_1)$. A second key assumption is that the dimension of $\boldsymbol{\theta}$ is small, for instance, $m < 15$. These assumptions are essential for enabling fast approximate inference.

The main aim in INLA is to approximate the marginal posterior distributions $\pi(z_i \mid \boldsymbol{y})$ and $\pi(\theta_l \mid \boldsymbol{y})$, $l = 1, 2, \ldots, m$. The posterior distribution of $\boldsymbol{\theta}$ is

$$\begin{aligned}\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) &= \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{p(\boldsymbol{y})} \\ &= \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{p(\boldsymbol{y})} \times \frac{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})} \\ &= \frac{p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{y}) \pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})} \\ &\propto \frac{p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})} \\ &\propto \frac{p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}) \pi(\boldsymbol{z} \mid \boldsymbol{\theta}) \pi(\boldsymbol{\theta})}{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})}.\end{aligned}$$

The numerator in the previous expression is easy to calculate (see Equation 14.1), but the denominator is generally not available in closed form and is difficult to compute. Thus, INLA approximates it at specific values $\boldsymbol{\theta}_g$,

$$\pi_a(\boldsymbol{\theta}_g \mid \boldsymbol{y}) \propto \frac{p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}_g) \pi(\boldsymbol{z} \mid \boldsymbol{\theta}_g) \pi(\boldsymbol{\theta}_g)}{\pi_{a,G}(\boldsymbol{z} \mid \boldsymbol{\theta}_g, \boldsymbol{y})},$$

where $\pi_{a,G}(z_i \mid \boldsymbol{\theta}_g, \boldsymbol{y})$ is a Gaussian approximation that matches the mode and covariance matrix of the full posterior $\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})$.

The approximation error of $\pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$ is $O(N^{-1})$ under standard conditions, meaning that the error, when multiplied by N, remains bounded [399, 361]. However, in many applications using INLA, the dimension of the latent Gaussian variables, P, increases with the sample size. In such cases, the error rate becomes $O(P/N)$. When $P/N \to 1$, which occurs in many models, the approximation error becomes $O(1)$: bounded, but potentially large.

Therefore, it is important to check the effective number of parameters, as the asymptotic error of $\pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$ depends on the dimension of $\boldsymbol{z}$. According to [361], in most of their applications the effective number of parameters is small relative to the sample size in regions near the mode of $\pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$.

The marginal posterior $\pi(z_i \mid \boldsymbol{\theta}, \boldsymbol{y})$ is more challenging to compute due to the potentially high dimension of $\boldsymbol{z}$. It may seem intuitive to use the Gaussian approximation $\pi_{a,G}(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})$; however, this is often not sufficiently accurate due to its lack of skewness. An alternative is to use the following expression

$$\begin{aligned}
\pi(z_i \mid \boldsymbol{\theta}, \boldsymbol{y}) &= \frac{p(z_i, \boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{\theta}, \boldsymbol{y})} \\
&= \frac{p(z_i, \boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{\theta}, \boldsymbol{y})} \times \frac{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})} \\
&= \frac{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{y})} \times p(z_i, \boldsymbol{\theta}, \boldsymbol{y}) \\
&= \frac{\pi(\boldsymbol{z} \mid \boldsymbol{\theta}, \boldsymbol{y})}{\pi(\boldsymbol{z}_{-i} \mid z_i, \boldsymbol{\theta}, \boldsymbol{y})} \\
&\propto \frac{p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}) \pi(\boldsymbol{z} \mid \boldsymbol{\theta}) \pi(\boldsymbol{\theta})}{\pi(\boldsymbol{z}_{-i} \mid z_i, \boldsymbol{\theta}, \boldsymbol{y})},
\end{aligned}$$

where the second-to-last equality follows from the identity $\pi(\boldsymbol{z}_{-i} \mid z_i, \boldsymbol{\theta}, \boldsymbol{y}) = \frac{p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{y})}{p(z_i, \boldsymbol{\theta}, \boldsymbol{y})}$, $-i$ denotes all elements of $\boldsymbol{z}$ except the i-th, and approximating the denominator in the last expression using a simplified Laplace approximation, which corrects the Gaussian approximation for location and skewness via a Taylor series expansion about the mode of the Laplace approximation [361]. This follows the spirit of the approximations developed by [399, 400] for posterior moments and marginal densities, and is similar to those used in Chapter 10 when performing Bayesian model averaging (BMA) using the BIC information criterion. The discussion about the asymptotic error of the approximation $\pi_a(z_i \mid \boldsymbol{\theta}, \boldsymbol{y})$ follows the same arguments as those for $\pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$.

We can obtain the marginal posterior distributions integrating with respect to $\boldsymbol{\theta}$,

$$\begin{aligned}
\pi(z_i \mid \boldsymbol{y}) &= \int_{\Theta} \pi(z_i \mid \boldsymbol{\theta}, \boldsymbol{y}) \pi(\boldsymbol{\theta} \mid \boldsymbol{y}) d\boldsymbol{\theta} \\
\pi(\theta_l \mid \boldsymbol{y}) &= \int_{\Theta} \pi(\boldsymbol{\theta} \mid \boldsymbol{y}) d\boldsymbol{\theta}_{-l}.
\end{aligned}$$

This integrals are solve numerically using a smart grid around the mode of $\pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$. In particular,

$$\pi_a(z_i \mid \boldsymbol{y}) = \sum_{g=1}^{G} \pi_a(z_i \mid \boldsymbol{\theta}_g, \boldsymbol{y}) \pi_a(\boldsymbol{\theta}_g \mid \boldsymbol{y}) \Delta_g,$$

where $\pi_a(z_i \mid \boldsymbol{\theta}_g, \boldsymbol{y})$ is the approximation of $\pi(z_i \mid \boldsymbol{\theta}, \boldsymbol{y})$ evaluated at $\boldsymbol{\theta}_g$. Then,

we have the sum over the values of $\boldsymbol{\theta}$ with area weights Δ_g. If all support points are equidistant, then $\Delta_g = 1$.

Algorithm A37 presents the INLA algorithm. Note that stages 2 and 3 correspond to the nested Laplace approximations, whereas stage 4 involves the integration step.

Algorithm A37 Integrated nested Laplace approximations (INLA)

1: Obtain the mode of $\pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$, that is $\boldsymbol{\theta}^*$, by maximizing $\log \pi_a(\boldsymbol{\theta} \mid \boldsymbol{y})$
2: Compute $\pi_a(\boldsymbol{\theta}_g \mid \boldsymbol{y})$ for a set of high density points $\boldsymbol{\theta}_g$, $g = 1, 2, \ldots, G$
3: Compute the approximation $\pi_a(z_i \mid \boldsymbol{\theta}_g, \boldsymbol{y})$, $g = 1, 2, \ldots, G$
4: Compute $\pi_a(z_i \mid \boldsymbol{y})$ and $\pi_a(\theta_l \mid \boldsymbol{y})$ using numerical integration

Thus, INLA can be used in LGMs that satisfy [362, 277]:

1. There is conditional independence, that is, $y_i \perp y_s \mid \eta_i, \boldsymbol{\theta}$, such that $p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i \mid \eta_i, \boldsymbol{\theta})$.
2. The dimension of $\boldsymbol{\theta}$ is small, less than 15.
3. $\boldsymbol{z}$ is a GMRF with sparse precision matrix.
4. The linear predictor depends linearly on the smooth unknown functions of covariates.
5. The inference is on the marginal posterior distributions $\pi(z_i \mid \boldsymbol{y})$ and $\pi(\theta_l \mid \boldsymbol{y})$.

[361] also discuss how to approximate the marginal likelihood and compute the deviance information criterion [386] for model selection, as well as how to perform predictive analysis.

The point of departure for INLA is the class of latent Gaussian models (LGMs); consequently, discrete latent classes are not supported. In addition, since INLA is based on approximations around the mode, it may struggle with models exhibiting multimodality, as there is no global exploration of the parameter space.

Implementing INLA from scratch is a complex task [277]; thus, applications are generally limited to the models implemented in the `INLA` package in **R**.[3] However, there are new packages that implement specialized models, as well as recent developments that combine INLA with MCMC algorithms (see Table 2 in [277] for details).

Example: Poisson model with unobserved heterogeneity

Let's simulate the model $Y_i \sim \text{Poisson}(\lambda_i)$, where $\lambda_i = \exp\{1 + x_i + \epsilon_i\}$, with $\epsilon_i \sim \mathcal{N}(0, 0.5^2)$ and $x_i \sim \mathcal{N}(0, 1^2)$ for $i = 1, 2, \ldots, 10{,}000$. Note that ϵ_i represents the unobserved heterogeneity.

The following code demonstrates how to perform inference on this model

[3] Visit https://www.r-inla.org/ for documentation.

using the *INLA* package. Keep in mind that INLA specifies Gaussian priors in terms of precision, which is the inverse of the variance.

We present the results obtained using the three approximation strategies available in INLA: Simplified Laplace (default), Gaussian, and full Laplace. In this example, the results are practically identical across methods (see Figures 14.4 and 14.5), and all 95% credible intervals contain the true population parameters. Despite the large sample size, INLA performs inference in a matter of seconds.

R code. Integrated Nested Laplace Approximations: Poisson model with unobserved heterogeneity

```
# Install Rtools according to your R version
# Check Rtolls is properly installed
# Install INLA
# install.packages("INLA",repos=c(getOption("repos"),INLA="https://inla.r-inla-download.org/R/stable"), dep=TRUE)
rm(list = ls()); set.seed(010101); library(INLA)
n <- 10000; x <- rnorm(n, sd = 1); u <- rnorm(n, sd = 0.5)
intercept <- 1; beta <- 1; id <- 1:n
y <- rpois(n, lambda = exp(intercept + beta * x + u))
my.data <- data.frame(y, x, id)
formula <- y ~ 1 + x + f(id, model="iid")
inla.sla <- inla(formula, data = my.data, family = "poisson", control.compute=list(return.marginals.predictor=TRUE))
inla.ga <- inla(formula, data = my.data, family = "poisson", control.inla = list(strategy = "gaussian", int.strategy = "eb"), control.compute=list(return.marginals.predictor=TRUE))
inla.la <- inla(formula, data = my.data, family = "poisson", control.inla = list(strategy = "laplace", int.strategy = "grid", dz=0.1, diff.logdens=20),
control.compute=list(return.marginals.predictor=TRUE))
summary(inla.sla)
marg_sla <- inla.sla$marginals.fixed$x
marg_ga  <- inla.ga$marginals.fixed$x
marg_la  <- inla.la$marginals.fixed$x
plot(marg_sla, type = "l", col = "blue", lwd = 2, xlab = expression(beta[x]), ylab = "Density", main = "Posterior of slope under different INLA strategies")
lines(marg_ga, col = "green", lwd = 2, lty = 2)
lines(marg_la, col = "red", lwd = 2, lty = 3)
abline(v = 1, col = "black", lty = 4, lwd = 2)
legend("topright", legend = c("Simplified Laplace", "Gaussian", "Full Laplace", "True = 1"), col = c("blue", "green", "red", "black"), lwd = 2, lty = c(1, 2, 3, 4), bty = "n")
# Summary beta sla
inla.qmarginal(c(0.025, 0.5, 0.975), marg_sla) # 95% credible interval
# Variance
marg.prec.sla <- inla.sla$marginals.hyperpar[["Precision for id"]]
marg.prec.ga  <- inla.ga$marginals.hyperpar[["Precision for id"]]
marg.prec.la  <- inla.la$marginals.hyperpar[["Precision for id"]]
marg.var.sla <- inla.tmarginal(function(x) 1/x, marg.prec.sla)
marg.var.ga  <- inla.tmarginal(function(x) 1/x, marg.prec.ga)
marg.var.la  <- inla.tmarginal(function(x) 1/x, marg.prec.la)
```

R code. Integrated Nested Laplace Approximations: Poisson model with unobserved heterogeneity

```
# Base plot
plot(marg.var.sla, type = "l", col = "blue", lwd = 2, xlab =
     expression(sigma^2), ylab = "Density", main = "
    Posterior of Random Effect Variance")
lines(marg.var.ga, col = "green", lwd = 2, lty = 2)
lines(marg.var.la, col = "red", lwd = 2, lty = 3)
abline(v = 0.25, col = "black", lty = 4, lwd = 2)
legend("topright", legend = c("Simplified Laplace", "
    Gaussian", "Full Laplace", "True Variance (0.25)"), col
    = c("blue", "green", "red", "black"), lwd = 2, lty = c
    (1, 2, 3, 4), bty = "n")
# Summary variance sla
inla.qmarginal(c(0.025, 0.5, 0.975), marg.var.sla) # 95%
    credible interval
```

FIGURE 14.4
Density estimates of slope parameter: Three approximations in INLA.

Example: Spatial econometrics model

The starting point of spatial econometrics is the *contiguity* or *adjacency* matrix W, which defines which spatial polygons (regions) are considered

FIGURE 14.5
Density estimates of variance: Three approximations in INLA.

neighbors. This is an $N \times N$ matrix, where each row and column corresponds to a spatial polygon, and a non-zero element indicates that two polygons are neighbors. By construction, the main diagonal is equal to zero, meaning that no polygon is a neighbor to itself. Given its structure, the contiguity matrix is sparse.

There are various ways to define contiguity between spatial units, with two of the most commonly used criteria being the *queen* and *rook* criteria, inspired by chess. In the queen criterion, two spatial units are considered neighbors if they share any part of a boundary or a point. In contrast, under the rook criterion, two units are considered neighbors only if they share a common boundary edge; they must touch along at least one side, and corner contact is not sufficient to establish a neighbor relationship.

However, users may apply any contiguity criteria they deem appropriate, depending on the context. For example, contiguity can be defined based on factors like the travel time by car between centroids or the proximity of main towns between municipalities. Typically, the contiguity matrix W is binary, with a value of 1 indicating that two regions are neighbors and 0 otherwise. Additionally, contiguity matrices are often row-standardized to facilitate the analysis of spatial stationarity.

[247, 44] present the most commonly used specifications in spatial econometrics, namely the spatial error model (SEM), the spatial autoregressive model (SAR), and the spatial Durbin model (SDM), which can, in turn, be combined to form the general nesting spatial (GNS) model [125, 326]. In par-

ticular, the SEM sets

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\mu} \\ \boldsymbol{\mu} &= \lambda \boldsymbol{W}\boldsymbol{\mu} + \boldsymbol{\epsilon},\end{aligned}$$

where $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}_N, \sigma^2 \boldsymbol{I}_N)$. Then, $\boldsymbol{\mu} = (\boldsymbol{I}_N - \rho \boldsymbol{W})^{-1}\boldsymbol{\epsilon}$, which implies that

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e},$$

where $\boldsymbol{e} \sim N(\boldsymbol{0}, \sigma^2(\boldsymbol{I}_N - \rho\boldsymbol{W})^{-1}(\boldsymbol{I}_N - \rho\boldsymbol{W}^\top)^{-1})$. In this model λ controls the degree of spatial dependence.

The SAR sets

$$\begin{aligned}\boldsymbol{y} &= \rho\boldsymbol{W}\boldsymbol{y} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\mu} \\ &= (\boldsymbol{I}_N - \rho\boldsymbol{W})^{-1}\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},\end{aligned}$$

where $\boldsymbol{\epsilon} = (\boldsymbol{I}_N - \rho\boldsymbol{W})^{-1}\boldsymbol{\mu}$.

The SDM sets

$$\begin{aligned}\boldsymbol{y} &= \rho\boldsymbol{W}\boldsymbol{y} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{W}\boldsymbol{X}\boldsymbol{\delta} + \boldsymbol{\mu} \\ &= (\boldsymbol{I}_N - \rho\boldsymbol{W})^{-1}(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{W}\boldsymbol{X}\boldsymbol{\delta}) + \boldsymbol{\epsilon},\end{aligned}$$

where $\boldsymbol{\epsilon} = (\boldsymbol{I}_N - \rho\boldsymbol{W})^{-1}\boldsymbol{\mu}$.

The degree of spatial dependence in the SAR and SDM models is determined by the parameter ρ. Note that the SDM model is similar to the SAR model, except that it also includes spatial lags of the regressors as additional covariates.

[302] and [18] show that a necessary condition for weak stationarity in spatial autoregressive processes with a row-standardized contiguity matrix is that the spatial autocorrelation coefficient must lie between $1/\omega_{\min}$ and 1, where $\omega_{\min}$ is the smallest (most negative) eigenvalue of the contiguity matrix.[4]

Note that these three models, conditional on the spatial parameter, are standard linear regressions, which can be easily estimated using *INLA*. [44] use the *INLABMA* package to estimate these models conditional on different values of the spatial parameters, and then perform Bayesian Model Averaging (BMA) using the resulting estimates. In particular, it is necessary to define a grid over the spatial parameters, perform Bayesian inference using INLA for each value in the grid, and then aggregate the posterior results using BMA.

We now perform Bayesian inference on a spatial econometric model using the *INLA* and *INLABMA* packages, based on the dataset provided by [333], who conducted a Bayesian analysis of electricity demand in the department of

[4]This is a necessary condition to ensure weak stationarity but is not sufficient due to edge and corner effects [178,].

Antioquia (Colombia), accounting for spatial dependence between municipalities in the region using a Conditional Autoregressive (CAR) spatial model. In particular, we conduct inference using the following specification:

$$\log(\text{Electricity}_i) = \beta_1 + \beta_2 \log(\text{Elect. price}_i) + \beta_3 \log(\text{Income}_i) + \beta_4 \log(\text{Subs. price}_i) + \mu_i,$$

where $\boldsymbol{\mu} = \lambda \boldsymbol{W} \boldsymbol{\mu} + \epsilon$; *Electricity* is the average per capita annual consumption of electricity by individuals living in households of stratum one in each municipality of the department of Antioquia; *Elect. price* is the average price of electricity (per kWh); *Income* is the average per capita annual income; and *Subs. price* is the average price of an electricity substitute (see [333] for details).

The following code illustrates how to carry out a spatial econometric analysis. First, we download the files needed to plot the maps and construct the contiguity matrix. These files are available in the folder *DataApp/Antioquia* of our GitHub repository: **https://github.com/besmarter/BSTApp**. The initial part of the code demonstrates how to download the *GitHub* repository and extract the necessary files for mapping.

Figure 14.6 displays the average electricity consumption per municipality. We observe the presence of spatial clusters, particularly in the northwestern region, which corresponds to a low-altitude area along the Caribbean coast.

The second part of the code constructs the contiguity matrix using the queen criterion. Figure 14.7 illustrates the spatial links between municipalities. Following this, we estimate a standard Ordinary Least Squares (OLS) regression and conduct spatial autocorrelation tests on the residuals. The null hypothesis of both the global and local Moran's I tests is the absence of spatial autocorrelation in the OLS residuals. We reject the null hypothesis.

Finally, we perform Bayesian inference over a predefined grid of values for the spatial coefficient and apply Bayesian Model Averaging (BMA) using the *INLABMA* package. Figure 14.8 shows the BMA posterior distribution of the spatial coefficient in the SEM, which indicates the presence of spatial dependence. Figure 14.9 displays the posterior density of the own-price elasticity of electricity demand.

R code. Spatial econometric analysis: Spatial error model using INLA

```
rm(list = ls()); set.seed(010101)
library(sf); library(tmap); library(classInt)
library(RColorBrewer); library(spdep); library(parallel)
library(INLABMA); library(sf); library(INLA)
zip_url <- "https://github.com/BEsmarter-consultancy/BSTApp/
    archive/refs/heads/master.zip"
temp_zip <- tempfile(fileext = ".zip")
download.file(zip_url, temp_zip, mode = "wb")
temp_dir <- tempdir()
unzip(temp_zip, exdir = temp_dir)
antioquia_path <- file.path(temp_dir, "BSTApp-master", "
    DataApp", "Antioquia")
list.files(antioquia_path)
antioquia_path <- file.path(temp_dir, "BSTApp-master", "
    DataApp", "Antioquia")
shp_file <- file.path(antioquia_path, "Antioquia.shp")
antioquia <- st_read(shp_file)
print(antioquia)
plot(st_geometry(antioquia), main = "Map of Antioquia")
interval <- classIntervals(antioquia$CONS_OLD, 5, style = "
    quantile")
antioquia$cons_class <- cut(antioquia$CONS_OLD,breaks =
    interval$brks,include.lowest = TRUE)
plotcolors <- brewer.pal(5, "Reds")
tmap_mode("plot")
tm_shape(antioquia) +
tm_fill(fill = "cons_class", fill.scale = tm_scale(values =
    plotcolors), fill.legend = tm_legend(title = "
    Electricity consumption")) + tm_borders(col = "grey90")
    + tm_compass(type = "8star", position = c("right", "top"
    )) + tm_layout(legend.outside = TRUE)
nb_object <- poly2nb(antioquia, queen = TRUE)
centroids <- st_centroid(st_geometry(antioquia))
coords <- st_coordinates(centroids)
plot(st_geometry(antioquia), border = "grey")
plot(nb_object, coords, add = TRUE, col = "red")
attach(antioquia)
fform <- L_CONS_OLD ~ L_P_OLD + L_ING_US + L_P_SUST
RegOLS <- lm(fform)
summary(RegOLS)
res <- RegOLS$residuals
NBList <- nb2listw(nb_object, style = "B")
moran_mc<- moran.mc(res, listw = NBList, 10000)
LM<-localmoran(as.vector(res), NBList)
sum(LM[,5]<0.05)
# Bayesian estimation
zero.variance <- list(prec = list(initial = 25, fixed = TRUE
    ))
ant.mat <- nb2mat(nb_object)
bmsp <- as(ant.mat, "CsparseMatrix")
antioquia$idx <- 1:nrow(antioquia)
rrho1 <- seq(0.5, 0.95, len = 10)
```

R code. Spatial econometric analysis: Spatial error model using INLA

```
semmodels <- mclapply(rrho1, function(rho) {
  sem.inla(fform, d = as.data.frame(antioquia), W = bmsp,
    rho = rho,
  family = "gaussian", impacts = FALSE,
  control.family = list(hyper = zero.variance),
  control.predictor = list(compute = TRUE),
  control.compute = list(dic = TRUE, cpo = TRUE),
  control.inla = list(print.joint.hyper = TRUE))
})
bmasem <- INLABMA(semmodels, rrho1, 0, impacts = FALSE)
#Display results
plot(bmasem$rho$marginal, type="l", col = "blue", lwd = 2,
    xlab = expression(lambda), ylab = "Density", main = "
    Spatial error model: Posterior spatial coefficient")
bmasem[["rho"]][["quantiles"]]
marg_sla <- bmasem[["marginals.fixed"]][["L_P_OLD"]]
plot(marg_sla, type = "l", col = "blue", lwd = 2, xlab =
    expression(beta[x]), ylab = "Density", main = "Spatial
    error model: Posterior price elasticity")
bmasem[["summary.fixed"]]
```

FIGURE 14.6
Per capita annual average electricity consumption: Stratum 1, Antioquia (Colombia).

FIGURE 14.7
Contiguity map: Queen criterion, municipalities (Antioquia, Colombia).

FIGURE 14.8
BMA posterior density: Spatial error coefficient.

FIGURE 14.9
BMA posterior density: Own-price elasticity of electricity.

14.2.2 Variational Bayes

Variational Bayes (VB) is a method from machine learning [217, 408] that replaces $\pi(\boldsymbol{\theta} \mid \mathbf{y})$ with an approximation obtained through optimization using the calculus of variations, hence the name *variational* Bayes. This approach is useful when the posterior distribution is complex (e.g., multimodal) or when the parameter space is high-dimensional, making MCMC or IS algorithms computationally expensive.

The goal in VB is to approximate the posterior distribution using a distribution $q(\boldsymbol{\theta})$ from a variational family $\mathcal{Q}$ —a class of distributions that is computationally convenient yet flexible enough to closely approximate the true posterior [47]-. The distribution q is called the *variational approximation* to the posterior, and a particular q in $\mathcal{Q}$ is defined by a specific set of *variational parameters*. Typically, this approximation is obtained by minimizing the Kullback–Leibler (KL) divergence between $q(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta} \mid \mathbf{y})$.

$$q^*(\boldsymbol{\theta}) := \underset{q \in \mathcal{Q}}{\arg\min}\ \mathrm{KL}(q(\boldsymbol{\theta}) || \pi(\boldsymbol{\theta} \mid \mathbf{y})), \tag{14.2}$$

where $\mathrm{KL}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta} \mid \mathbf{y})) = \mathbb{E}_q \left[\log \left(\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta} | \mathbf{y})} \right) \right]$.[5]

Note that in relatively complex models, the optimization in Equation 14.2 is not computable because it depends on the marginal likelihood $p(\boldsymbol{y})$, which is typically unknown due to the intractability of the integral involved. However,

[5]The Kullback–Leibler (KL) divergence, also known as relative entropy, is non-negative: it equals 0 when $q(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta} \mid \mathbf{y})$, and is positive otherwise. However, it does not satisfy the triangle inequality and is not symmetric, that is, $\mathrm{KL}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta} \mid \mathbf{y})) \neq \mathrm{KL}(\pi(\boldsymbol{\theta} \mid \mathbf{y}) \| q(\boldsymbol{\theta}))$. Therefore, it is not a true distance (metric).

there is a solution to this problem. Let's see:

$$\begin{aligned}
\log(p(\boldsymbol{y})) &= \log\left(\int_{\boldsymbol{\Theta}} p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}\right) \\
&= \log\left(\int_{\boldsymbol{\Theta}} p(\boldsymbol{y}, \boldsymbol{\theta})d\boldsymbol{\theta}\right) \\
&= \log\left(\int_{\boldsymbol{\Theta}} \frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} q(\boldsymbol{\theta})d\boldsymbol{\theta}\right) \\
&= \log \mathbb{E}_q\left(\frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right) \\
&\geq \mathbb{E}_q \log\left(\frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right) \\
&= \mathbb{E}_q \log(p(\boldsymbol{y}, \boldsymbol{\theta})) - \mathbb{E}_q \log(q(\boldsymbol{\theta})) \\
&= \mathrm{ELBO}(q(\boldsymbol{\theta})),
\end{aligned}$$

where the inequality follows from Jensen's inequality, since $\log(\cdot)$ is concave. The last term is the *evidence lower bound* (ELBO), which serves as a lower bound for the marginal likelihood. Note that the gap between the marginal likelihood and the ELBO is given by

$$\begin{aligned}
\log(p(\boldsymbol{y})) - \mathrm{ELBO}(q(\boldsymbol{\theta})) &= \log(p(\boldsymbol{y})) - \mathbb{E}_q \log(p(\boldsymbol{y}, \boldsymbol{\theta})) + \mathbb{E}_q \log(q(\boldsymbol{\theta})) \\
&= \mathbb{E}_q\left(\log(q(\boldsymbol{\theta})) - \log\left(\frac{p(\boldsymbol{y}, \boldsymbol{\theta})}{p(\boldsymbol{y})}\right)\right) \\
&= \mathbb{E}_q\left(\log(q(\boldsymbol{\theta})) - \log\left(\frac{p(\boldsymbol{y} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{p(\boldsymbol{y})}\right)\right) \\
&= \mathbb{E}_q\left(\log(q(\boldsymbol{\theta})) - \log(\pi(\boldsymbol{\theta} \mid \boldsymbol{y}))\right) \\
&= \mathrm{KL}(q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta} \mid \mathbf{y})).
\end{aligned}$$

Then,

$$\log(p(\boldsymbol{y})) = \mathrm{KL}(q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta} \mid \mathbf{y})) + \mathrm{ELBO}(q(\boldsymbol{\theta})),$$

which implies that maximizing the ELBO with respect to $q(\boldsymbol{\theta})$ is equivalent to minimizing the KL divergence, since $\log(p(\boldsymbol{y}))$ does not depend on $q(\boldsymbol{\theta})$. This avoids the need to compute the marginal likelihood and, consequently, makes the variational problem easier to solve. In addition, it provides a lower bound for the marginal likelihood, which can potentially be used for model selection. Thus, solving problem 14.2 is equivalent to solving

$$q^*(\boldsymbol{\theta}) := \underset{q \in \mathcal{Q}}{\arg\max}\ \mathrm{ELBO}(q(\boldsymbol{\theta})). \tag{14.3}$$

Note that the ELBO can be also expressed as

$$\begin{aligned}
\mathrm{ELBO}(q(\boldsymbol{\theta})) &= \mathbb{E}_q \log(p(\boldsymbol{y}, \boldsymbol{\theta})) - \mathbb{E}_q \log(q(\boldsymbol{\theta})) \\
&= \mathbb{E}_q \log(p(\boldsymbol{y} \mid \boldsymbol{\theta})) + \mathbb{E}_q \log(\pi(\boldsymbol{\theta})) - \mathbb{E}_q \log(q(\boldsymbol{\theta})) \\
&= \mathbb{E}_q \log(p(\boldsymbol{y} \mid \boldsymbol{\theta})) - \mathrm{KL}(q(\boldsymbol{\theta})||\pi(\boldsymbol{\theta})).
\end{aligned}$$

This means that when maximizing the ELBO, we seek the distribution $q(\boldsymbol{\theta})$ that both maximizes the likelihood and minimizes the KL divergence between the variational distribution and the prior. In other words, we aim to strike a balance between the prior and the likelihood, which aligns with the core principle of Bayesian inference deriving the posterior distribution.

The most common approach for specifying $q(\boldsymbol{\theta})$ is to assume independence across blocks of $\boldsymbol{\theta}$, i.e., $q(\boldsymbol{\theta}) = \prod_{l=1}^{K} q_l(\boldsymbol{\theta}_l)$. This is known as the *mean-field variational family*, a term that originates from statistical physics. Each $q_l(\boldsymbol{\theta}_l)$ is parameterized by a set of *variational parameters*, and optimization is performed with respect to these parameters. Note that the mean-field approximation does not capture dependencies between parameters, although it can approximate the marginal distributions.

Let us now decompose the ELBO under the mean-field variational family [296],

$$
\begin{aligned}
\mathrm{ELBO}(q(\boldsymbol{\theta})) &= \mathbb{E}_q \log(p(\boldsymbol{y}, \boldsymbol{\theta})) - \mathbb{E}_q \log(q(\boldsymbol{\theta})) \\
&= \int_{\mathbb{R}^K} \log(p(\boldsymbol{y}, \boldsymbol{\theta})) q(\boldsymbol{\theta}) d\boldsymbol{\theta} - \int_{\mathbb{R}^K} \log(q(\boldsymbol{\theta})) q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&= \int_{\mathbb{R}^K} \log\left(p(\boldsymbol{y}, \boldsymbol{\theta})\right) \prod_{l=1}^{K} q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l - \int_{\mathbb{R}^K} \log\left(\prod_{l=1}^{K} q_l(\boldsymbol{\theta}_l)\right) \prod_{l=1}^{K} q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l \\
&= \int_{\mathbb{R}} \underbrace{\left(\int_{\mathbb{R}^{K-1}} \log\left(p(\boldsymbol{y}, \boldsymbol{\theta})\right) \prod_{l \neq k} q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l\right)}_{\mathbb{E}_{-k}(\log(p(\boldsymbol{y}, \boldsymbol{\theta})))} q_k(\boldsymbol{\theta}_k) d\boldsymbol{\theta}_k \\
&\quad - \int_{\mathbb{R}} \log(q_k(\boldsymbol{\theta}_k)) q_k(\boldsymbol{\theta}_k) d\boldsymbol{\theta}_k - \sum_{l \neq k} \int_{\mathbb{R}} \log(q_l(\boldsymbol{\theta}_l)) q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l \\
&= \int_{\mathbb{R}} \log\left\{\exp\left(\mathbb{E}_{-k}(\log\left(p(\boldsymbol{y}, \boldsymbol{\theta})\right))\right)\right\} q_k(\boldsymbol{\theta}_k) d\boldsymbol{\theta}_k \\
&\quad - \int_{\mathbb{R}} \log(q_k(\boldsymbol{\theta}_k)) q_k(\boldsymbol{\theta}_k) d\boldsymbol{\theta}_k - \sum_{l \neq k} \int_{\mathbb{R}} \log(q_l(\boldsymbol{\theta}_l)) q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l \\
&= \int_{\mathbb{R}} \log\left\{\frac{\exp\left(\mathbb{E}_{-k}(\log\left(p(\boldsymbol{y}, \boldsymbol{\theta})\right))\right)}{q_k(\boldsymbol{\theta}_k)}\right\} q_k(\boldsymbol{\theta}_k) d\boldsymbol{\theta}_k \\
&\quad - \sum_{l \neq k} \int_{\mathbb{R}} \log(q_l(\boldsymbol{\theta}_l)) q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l \\
&= -\mathrm{KL}(q_k(\boldsymbol{\theta}_k) || \exp\left(\mathbb{E}_{-k}(\log\left(p(\boldsymbol{y}, \boldsymbol{\theta})\right))\right)) - \sum_{l \neq k} \int_{\mathbb{R}} \log(q_l(\boldsymbol{\theta}_l)) q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l \\
&\leq -\sum_{l \neq k} \int_{\mathbb{R}} \log(q_l(\boldsymbol{\theta}_l)) q_l(\boldsymbol{\theta}_l) d\boldsymbol{\theta}_l,
\end{aligned}
$$

where $\mathbb{E}_{-k}$ denotes expectation with respect to the distribution $\prod_{l \neq k} q(\boldsymbol{\theta}_l)$.

Note that we maximize the ELBO when $\mathrm{KL}(q_k(\boldsymbol{\theta}_k)||\exp(\mathbb{E}_{-k}(\log(p(\boldsymbol{y},\boldsymbol{\theta})))))$ equals 0, that is, when

$$\begin{aligned} q_k^*(\boldsymbol{\theta}_k) &\propto \exp(\mathbb{E}_{-k}(\log(p(\boldsymbol{y},\boldsymbol{\theta})))) \\ &= \exp(\mathbb{E}_{-k}(\log(p(\boldsymbol{y},\boldsymbol{\theta}_{-k},\boldsymbol{\theta}_k)))) \\ &= \exp(\mathbb{E}_{-k}(\log(p(\boldsymbol{\theta}_k \mid \boldsymbol{y},\boldsymbol{\theta}_{-k})p(\boldsymbol{y},\boldsymbol{\theta}_{-k})))) \\ &\propto \exp(\mathbb{E}_{-k}(\log(p(\boldsymbol{\theta}_k \mid \boldsymbol{y},\boldsymbol{\theta}_{-k})))), k = 1,2,\ldots,K. \end{aligned} \tag{14.4}$$

Note the circular dependency inherent in $q_k^*(\boldsymbol{\theta}_k)$: it depends on $q_{-k}^*(\boldsymbol{\theta}_{-k})$. Therefore, this situation must be addressed algorithmically. One of the most common algorithms used to solve the optimization problem in 14.3 using 14.4 is the *coordinate ascent variational inference* (CAVI) algorithm [42]. The algorithm starts from an initial solution and iteratively cycles through each $q_k^*(\boldsymbol{\theta}_k)$, for $k = 1,2,\ldots,K$, updating each component in turn. Algorithm A38 outlines the basic structure of the CAVI algorithm.

Algorithm A38 Variational Bayes: Coordinate ascent variational inference

```
Initialize the variational factors q_l^*(θ_l), l = 1,2,...,K
while ELBO > ϵ, ϵ small do
    for l = 1,2,...,L do
        Set q_l^*(θ_l) ∝ exp(E_{-l}(log(p(θ_l | y, θ_{-l}))))
    end for
    Compute ELBO(q) = E_q log(p(y, θ)) − E_q log(q(θ))
end while
Return q(θ)
```

We should note that the CAVI algorithm is sensitive to the initial variational factors because it guarantees convergence to a local maximum, which may depend heavily on the initialization point. This issue can be mitigated by using multiple starting points and selecting the solution with the highest ELBO [46]. In addition, it is well known that VB tends to overestimate the precision of the posterior distribution, although it does not necessarily suffer in terms of accuracy [47].

An important feature of the method is that it is deterministic rather than stochastic; that is, it does not require approximating the posterior distribution via sampling from simpler distributions. This often makes VB faster than MCMC methods, which are inherently stochastic. Instead, VB solves a deterministic optimization problem to find the *best variational distribution* within a chosen family, typically from the exponential family, by optimizing the variational parameters to minimize the KL divergence from the posterior distribution. Once the *variational parameters* are obtained, we can sample from the *variational distribution* to conduct estimation, hypothesis testing, prediction, and other tasks.

A limitation of the CAVI algorithm is that it requires evaluation at each data point, making it non-scalable in large data settings. In such situations,

we can use *stochastic variational inference* (SVI) [195], an algorithm that optimizes the ELBO using natural gradients combined with stochastic optimization. SVI is particularly effective when each complete conditional belongs to the exponential family (see Section 3.1), which includes most of the models discussed in this book. It is especially useful for conditionally conjugate models that include *local* latent variables ($\boldsymbol{z}_i$) associated with specific data points, and *global* parameters ($\boldsymbol{\phi}$) shared across the entire dataset. That is, we define $\boldsymbol{\theta} = [\boldsymbol{z}^\top \ \boldsymbol{\phi}^\top]^\top$. An example is the probit model using data augmentation [393].

Given the global-local exchangeable structure, the joint distribution can be expressed as $p(\boldsymbol{\phi}, \boldsymbol{z}, \boldsymbol{y}) = \pi(\boldsymbol{\phi} \mid \boldsymbol{\alpha}) \prod_{i=1}^N p(\boldsymbol{z}_i, \boldsymbol{y}_i \mid \boldsymbol{\phi})$, where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1^\top \ \alpha_2]^\top$ represents the hyperparameters of the prior distribution of the global parameters $\boldsymbol{\phi}$.

Assuming that the joint distribution $p(\boldsymbol{z}_i, \boldsymbol{y}_i \mid \boldsymbol{\phi})$ is in the canonical form of the exponential family, and that each prior distribution is conjugate, the complete conditional distribution of the global parameters is also in the exponential family. The posterior parameters are given by $\boldsymbol{\alpha}_n = [\boldsymbol{\alpha}_1^\top + \sum_{i=1}^N t(\boldsymbol{z}_i, \boldsymbol{y}_i)^\top \ \alpha_2 + N]^\top$, where $t(\boldsymbol{z}_i, \boldsymbol{y}_i)$ is a sufficient statistic of $\boldsymbol{z}_i$ and $\boldsymbol{y}_i$ (see Section 3.2).

Additionally, the structure of the model implies that $\boldsymbol{z}_i$ is independent of $\boldsymbol{z}_{-i}$ and $\boldsymbol{y}_{-i}$ given $\boldsymbol{y}_i$ and $\boldsymbol{\phi}$. Thus, if $p(\boldsymbol{z}_i \mid \boldsymbol{y}_i, \boldsymbol{\phi})$ is in the canonical form of the exponential family, then the local variational update is given by $\boldsymbol{\psi}_i = \mathbb{E}_{\boldsymbol{\xi}}[\boldsymbol{\eta}(\boldsymbol{\phi}, \boldsymbol{y}_i)]$, where the parameter set $\boldsymbol{\eta}(\boldsymbol{\phi}, \boldsymbol{y}_i)$ is a function of the conditional set, $\boldsymbol{\xi}$, and $\boldsymbol{\psi}_i$ are the variational parameters of the variational approximations $q(\boldsymbol{\phi} \mid \boldsymbol{\xi})$ and $q(\boldsymbol{z}_i \mid \boldsymbol{\psi}_i)$ for the posterior distributions of $\boldsymbol{\phi}$ and $\boldsymbol{z}_i$, respectively. The global variational update is given by $\boldsymbol{\xi} = [\boldsymbol{\alpha}_1^\top + \sum_{i=1}^N \mathbb{E}_{\boldsymbol{\psi}_i} t(\boldsymbol{z}_i, \boldsymbol{y}_i)^\top \ \alpha_2 + N]^\top$.

SVI focuses on the global parameters, updating them using the ELBO natural gradients with respect to $\boldsymbol{\xi}$, where natural gradients are the usual gradients premultiplied by the inverse covariance matrix of the sufficient statistic. These gradients are easily calculated in exponential families. In particular, the updates are given by

$$\boldsymbol{\xi}_t = \boldsymbol{\xi}_{t-1} + \epsilon_t g(\boldsymbol{\xi}_{t-1}),$$

where ϵ_t is the step size, and $g(\boldsymbol{\xi}_t) = \mathbb{E}_{\boldsymbol{\psi}_i}[\boldsymbol{\alpha}_n - \boldsymbol{\xi}_t]$ is the natural gradient of the global variational parameters [47]. Consequently,

$$\boldsymbol{\xi}_t = (1 - \epsilon_t)\boldsymbol{\xi}_{t-1} + \epsilon_t \mathbb{E}_{\boldsymbol{\psi}}[\boldsymbol{\alpha}_n].$$

However, calculating this update requires using the entire dataset, which is computationally burdensome. Therefore, we should use stochastic optimization, which follows noisy but cheap-to-compute unbiased gradients to optimize the function. The key idea is to construct the natural gradient using only one random draw from the dataset, and then scale it (by multiplying it with the

sample size) to approximate the sample information:

$$\hat{\boldsymbol{\xi}} = \boldsymbol{\alpha} + N(\mathbb{E}_{\boldsymbol{\psi}_i^*}[t(\boldsymbol{z}_i, \boldsymbol{y}_i)]^\top, 1)^\top$$
$$\boldsymbol{\xi}_t = (1 - \epsilon_t)\boldsymbol{\xi}_{t-1} + \epsilon_t \hat{\boldsymbol{\xi}},$$

where $(\boldsymbol{z}_i, \boldsymbol{y}_i)$ is a random draw from the sample and its corresponding latent variable. Finally, the step size schedule to update the global parameters are given by

$$\epsilon_t = t^{-\kappa},\ 0.5 < \kappa \leq 1.$$

This step size schedule satisfies the Robbins and Monro conditions necessary for stochastic optimization [341].

Algorithm A39 shows the stochastic variational inference algorithm.

Algorithm A39 Variational Bayes: Stochastic variational inference

1: Initialize the variational global parameter $\boldsymbol{\phi}_0$
2: Set the step size schedule $\epsilon_t = t^{-\kappa},\ 0.5 < \kappa \leq 1$
3: **while** TRUE **do**
4: Choose randomly a data point $\boldsymbol{y}_i$ using a discrete uniform distribution, $i \sim U(1, 2, \ldots, N)$
5: Optimize its local variational parameters $\boldsymbol{\psi}_i^* = \mathbb{E}_{\boldsymbol{\xi}_{t-1}}[\boldsymbol{\eta}(\boldsymbol{\phi}, \boldsymbol{y_i})]$
6: Compute the coordinates updates as though $\boldsymbol{y}_i$ were repeated N times, $\hat{\boldsymbol{\xi}} = \boldsymbol{\alpha} + N(\mathbb{E}_{\boldsymbol{\psi}_i^*}[t(\boldsymbol{z}_i, \boldsymbol{y}_i)]^\top, 1)^\top$
7: Update the global variational parameters, $\boldsymbol{\xi}_t = (1 - \epsilon_t)\boldsymbol{\xi}_{t-1} + \epsilon_t \hat{\boldsymbol{\xi}}$
8: **end while**
9: Return $q_{\boldsymbol{\xi}}(\boldsymbol{\phi})$

[431] analyzes the asymptotic properties of the VB posterior by decomposing the convergence rates of VB into the convergence rate of the true posterior and the approximation error induced by the variational family. These authors show that the VB posterior concentrates entirely in a neighborhood of the true posterior distribution. In addition, if the loss function is convex, there exists a point estimator that converges at the same rate. [431] also shows that, for specific cases such as sparse linear models, the concentration rate of the VB posterior is faster than the concentration rate of the exact posterior.

Variational Bayes (VB) shares some similarities with Gibbs sampling. In Gibbs sampling, we iteratively sample from the conditional posterior distributions, whereas in VB we iteratively update the variational parameters of the variational family. The former is stochastic, while the latter is deterministic, and consequently faster in complex models or large datasets. VB also bears resemblance to Expectation Propagation (EP): both are deterministic algorithms that approximate the posterior distribution by minimizing the Kullback-Leibler (KL) divergence. However, VB minimizes $\mathrm{KL}(q(\boldsymbol{\theta}) \| \pi(\boldsymbol{\theta} \mid \boldsymbol{y}))$, whereas EP minimizes $\mathrm{KL}(\pi(\boldsymbol{\theta} \mid \boldsymbol{y}) \| q(\boldsymbol{\theta}))$. As a result, VB tends to approximate the mode of the posterior by maximizing the ELBO, while EP focuses

on matching the mean and variance through moment matching [42, 150]. Although EP often provides better uncertainty quantification, it tends to be less stable and does not scale well to large datasets.

VB also shares some conceptual features with the Expectation-Maximization (EM) algorithm. In both methods, there is an initial step involving the computation of expectations —used in VB to derive the variational distributions— and an iterative step that maximizes a target function (the ELBO in VB, the expected complete-data log-likelihood in EM). However, EM yields point estimates, whereas VB yields full posterior approximations.

Example: Linear regression

Let's perform variational Bayes inference in the linear regression model with conjugate family. In particular,

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\mu},$$

where $\boldsymbol{\mu} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. This implies that:

$$\boldsymbol{y} \sim N(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}).$$

The conjugate priors for the parameters are

$$\begin{aligned}\boldsymbol{\beta} \mid \sigma^2 &\sim N(\boldsymbol{\beta}_0, \sigma^2\boldsymbol{B}_0),\\ \sigma^2 &\sim IG(\alpha_0/2, \delta_0/2).\end{aligned}$$

Then, the posterior distributions are $\boldsymbol{\beta} \mid \sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \sigma^2\boldsymbol{B}_n)$, and $\sigma^2 \mid \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2)$, where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}})$, $\hat{\boldsymbol{\beta}}$ is the MLE, $\alpha_n = \alpha_0 + N$ and $\delta_n = \delta_0 + \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\beta}_0^\top\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 - \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$ (see Section 3.3).

Let's use the mean-field variational family $q(\boldsymbol{\beta}, \sigma^2) = q(\boldsymbol{\beta})q(\sigma^2) \approx \pi(\boldsymbol{\beta}, \sigma^2 \mid \boldsymbol{y}, \boldsymbol{X})$.[6] Then,

$$\begin{aligned}\log q^*(\boldsymbol{\beta}) &\propto \mathbb{E}_{\sigma^2}[\log p(\boldsymbol{y}, \boldsymbol{\beta}, \sigma^2 \mid \boldsymbol{X})]\\ &= \mathbb{E}_{\sigma^2}[\log p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X}) + \log \pi(\boldsymbol{\beta} \mid \sigma^2) + \log \pi(\sigma^2)]\\ &= \mathbb{E}_{\sigma^2}\left(-\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) - \frac{K}{2}\log\sigma^2\right.\\ &\quad\left.-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top\boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)\right) + c_1\\ &= -0.5\mathbb{E}_{\sigma^2}\left(\frac{1}{\sigma^2}\right)[(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)] + c_2.\end{aligned}$$

The last equality follows the same steps to obtain the posterior distribution

[6] We can also use the variational family $q(\boldsymbol{\beta}, \sigma^2) = q(\boldsymbol{\beta} \mid \sigma^2)q(\sigma^2)$, which may be more straightforward to manipulate.

of $\boldsymbol{\beta}$, c_1 and c_2 are arbitrary constants. This expression implies that $q^*(\boldsymbol{\beta})$ is $N\left(\boldsymbol{\beta}_n, \left(\mathbb{E}_{\sigma^2}\left(\frac{1}{\sigma^2}\right)\right)^{-1}\boldsymbol{B}_n\right)$.

In addition,

$$\begin{aligned}
\log q^*(\sigma^2) &\propto \mathbb{E}_{\boldsymbol{\beta}}[\log p(\boldsymbol{y}, \boldsymbol{\beta}, \sigma^2 \mid \boldsymbol{X})] \\
&= \mathbb{E}_{\boldsymbol{\beta}}[\log p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X}) + \log \pi(\boldsymbol{\beta} \mid \sigma^2) + \log \pi(\sigma^2)] \\
&= \mathbb{E}_{\boldsymbol{\beta}}\left(-\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) - \frac{K}{2}\log\sigma^2\right. \\
&\left.-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\boldsymbol{\beta}_0)^\top\boldsymbol{B}_0^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_0) - (\alpha_0/2+1)\log\sigma^2 - \frac{\delta_0}{2\sigma^2}\right) + c_1 \\
&= -\mathbb{E}_{\boldsymbol{\beta}}\left[\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{\beta}-\boldsymbol{\beta}_0)^\top\boldsymbol{B}_0^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_0) + \delta_0\right] \\
&- \left(\frac{N+K+\alpha_0}{2}+1\right)\log\sigma^2 + c_1.
\end{aligned}$$

This means that $q^*(\sigma^2)$ is $IG(\alpha_n/2, \delta_n/2)$, where $\alpha_n = N + K + \alpha_0$, and

$$\begin{aligned}
\delta_n &= \mathbb{E}_{\boldsymbol{\beta}}\left[(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{\beta}-\boldsymbol{\beta}_0)^\top\boldsymbol{B}_0^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_0) + \delta_0\right] \\
&= \mathbb{E}_{\boldsymbol{\beta}}\left[(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)\right] - \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\beta}_0^\top\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \delta_0.
\end{aligned}$$

This result implies that $\mathbb{E}_{\sigma^2}\left(\frac{1}{\sigma^2}\right) = \alpha_n/\delta_n$, since the inverse of a gamma-distributed random variable (in the rate parametrization) follows an inverse-gamma distribution. Therefore, $\mathrm{Var}(\boldsymbol{\beta}) = \frac{\delta_n}{\alpha_n}\boldsymbol{B}_n$. Note that

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\beta}}\left[(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)\right] &= tr\left\{\mathbb{E}_{\boldsymbol{\beta}}\left[(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)\right]\right\} \\
&= \mathbb{E}_{\boldsymbol{\beta}}\left\{tr\left[(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)\right]\right\} \\
&= \mathbb{E}_{\boldsymbol{\beta}}\left\{tr\left[(\boldsymbol{\beta}-\boldsymbol{\beta}_n)(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}\right]\right\} \\
&= tr\left\{\mathbb{E}_{\boldsymbol{\beta}}\left[(\boldsymbol{\beta}-\boldsymbol{\beta}_n)(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\right]\boldsymbol{B}_n^{-1}\right\} \\
&= tr\left\{Var(\boldsymbol{\beta})\boldsymbol{B}_n^{-1}\right\},
\end{aligned}$$

where we use that the trace of a scalar is the scalar itself, that expectation and trace can be interchanged since both are linear operators, and that the trace operator is invariant under cyclic permutations. Then,

$$\begin{aligned}
\delta_n &= tr\left\{Var(\boldsymbol{\beta})\boldsymbol{B}_n^{-1}\right\} - \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\beta}_0^\top\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \delta_0 \\
&= \left(\frac{\alpha_0+N+K}{\alpha_0+K}\right)\left(-\boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{y}^\top\boldsymbol{y} + \boldsymbol{\beta}_0^\top\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \delta_0\right),
\end{aligned}$$

where the last equality follows from $tr\left\{Var(\boldsymbol{\beta})\boldsymbol{B}_n^{-1}\right\} = \delta_n/\alpha_n tr\left\{\boldsymbol{I}_K\right\}$.

In addition (Exercise 4),

$$\begin{aligned}\text{ELBO}(q(\boldsymbol{\beta},\sigma^2)) &= \mathbb{E}_{\boldsymbol{\beta},\sigma^2}[\log p(\boldsymbol{y},\boldsymbol{\beta},\sigma^2 \mid \boldsymbol{X})] - \mathbb{E}_{\boldsymbol{\beta},\sigma^2}[\log q(\boldsymbol{\beta},\sigma^2)] \\ &= -\frac{N}{2}\log(2\pi) + \frac{\alpha_0}{2}\log(\delta_0/2) - \frac{\alpha_n}{2}\log(\delta_n/2) - 0.5\log|\boldsymbol{B}_0| \\ &+ 0.5\log|\boldsymbol{B}_n| - \log\Gamma(\alpha_0/2) + \log\Gamma(\alpha_n/2) \\ &- K/2\log(\alpha_n/\delta_n) + 0.5tr\left\{Var(\boldsymbol{\beta})\boldsymbol{B}_n^{-1}\right\}.\end{aligned}$$

Note that the first two lines of the ELBO share the same structure as the log marginal likelihood $p(\boldsymbol{y})$ in Section 3.3.

The following code presents a simulation setting with a sample size of 500 and two regressors drawn from standard normal distributions. The population parameters are set to 1, and we use non-informative priors with 10,000 posterior draws.

First, we perform VB inference using the *LaplacesDemon* package, and then implement it from scratch. Although we have analytical solutions in this setting, we apply our own CAVI algorithm to assess its performance. We also compare the results with those from the Gibbs sampler, the marginal likelihood, and the ELBO.

In general, all calculations seem to perform well: the 95% credible intervals contain the population parameters, and the posterior means are very close to them.

R code. Variational Bayes: Linear regression model

```
set.seed(010101)
library(LaplacesDemon)
N <- 500; K <- 2
sig2 <- 1; B <- rep(1, K + 1)
X <- cbind(1, matrix(rnorm(N*K), N, K))
e <- rnorm(N, 0, sig2^0.5)
y <- X%*%B + e
######################### Data List Preparation #########################
mon.names <- "mu[1]"
parm.names <- as.parm.names(list(beta=rep(0,K + 1), sigma2=0))
pos.beta <- grep("beta", parm.names)
pos.sigma2 <- grep("sigma2", parm.names)
PGF <- function(Data) {
  beta <- rnorm(Data$K)
  sigma2 <- runif(1)
  return(c(beta, sigma2))
}
MyData <- list(K=K, PGF=PGF, X=X, mon.names=mon.names,
parm.names=parm.names, pos.beta=pos.beta, pos.sigma2=pos.sigma2, y=y)
########################## Model Specification ##########################
b0 <- 0; B0 <- 1000; a0 <- 0.01; d0 <- 0.01
Model <- function(parm, Data)
{
  ### Parameters
  beta <- parm[Data$pos.beta]
  sigma2 <- interval(parm[Data$pos.sigma2], 1e-100, Inf)
  parm[Data$pos.sigma2] <- sigma2
  ### Log-Prior
  beta.prior <- sum(dnormv(beta, b0, B0, log=TRUE))
  sigma2.prior <- dinvgamma(sigma2, a0/2, d0/2, log=TRUE)
  ### Log-Likelihood
  mu <- tcrossprod(Data$X, t(beta))
  LL <- sum(dnorm(Data$y, mu, sigma2^0.5, log=TRUE))
  ### Log-Posterior
  LP <- LL + beta.prior + sigma2.prior
  Modelout <- list(LP=LP, Dev=-2*LL, Monitor=mu[1],
  yhat=rnorm(length(mu), mu, sigma2^0.5), parm=parm)
  return(Modelout)
}
Initial.Values <- rep(0,K+2); S <- 10000
Fit <- VariationalBayes(Model, Initial.Values, Data=MyData, Covar=NULL,
Iterations=S, Method="Salimans2", Stop.Tolerance=1e-2, CPUs=1)
print(Fit)
PosteriorChecks(Fit)
caterpillar.plot(Fit, Parms="beta")
plot(Fit, MyData, PDF=FALSE)
Pred <- predict(Fit, Model, MyData, CPUs=1)
summary(Pred, Discrep="Chi-Square")
```

R code. Variational Bayes: Linear regression model

```
####### MCMC #######
# Posterior parameters
b0 <- rep(b0, K+1); B0 <- B0*diag(K+1)
bhat <- solve(t(X)%*%X)%*%t(X)%*%y
Bn <- as.matrix(Matrix::forceSymmetric(solve(solve(B0) + t(X)%*%X)))
bn <- Bn%*%(solve(B0)%*%b0 + t(X)%*%X%*%bhat)
dn <- as.numeric(d0 + t(y)%*%y+t(b0)%*%solve(B0)%*%b0-t(bn)%*%solve(Bn)%*%bn)
an <- a0 + N
# Posterior draws
sig2 <- MCMCpack::rinvgamma(S,an/2,dn/2)
summary(coda::mcmc(sig2))
Betas <- t(sapply(1:S, function(s){MASS::mvrnorm(1, bn, sig2[s]*Bn)}))
summary(coda::mcmc(Betas))
####### VB from scratch #######
dnVB <- ((a0+N+K)/(a0+N))*dn; anVB <- a0 + N + K
BnVB <- Bn; bnVB <- bn
sig2VB <- MCMCpack::rinvgamma(S,anVB/2,dnVB/2)
BetasVB <- MASS::mvrnorm(S, mu = bnVB, Sigma = (dn/an)*BnVB)
summary(coda::mcmc(sig2VB)); summary(coda::mcmc(BetasVB))
ELBO <- -N/2*log(2*pi) + a0/2*log(d0/2) - anVB/2*log(dnVB/2) - 0.5*log(det(B0)) + 0.5*log(det(BnVB)) - lgamma(a0/2) + lgamma(anVB/2) - K/2*log(anVB/dnVB) + K/2
LogMarLik <- -N/2*log(2*pi) + a0/2*log(d0/2) - an/2*log(dn/2) - 0.5*log(det(B0)) + 0.5*log(det(Bn)) - lgamma(a0/2) + lgamma(an/2)
ELBO; LogMarLik; ELBO < LogMarLik
# CAVI
ELBOfunc <- function(d,B){
  ELBOi <- -N/2*log(2*pi) + a0/2*log(d0/2) - anVB/2*log(d/2) - 0.5*log(det(B0)) + 0.5*log(det(B)) - lgamma(a0/2) + lgamma(anVB/2) - K/2*log(anVB/d) + 0.5*(anVB/d)*sum(diag(B%*%solve(Bn)))
  return(ELBOi)
}
d <- 100; B <- diag(K)
Esig2inv <- anVB/d; ELBOs <- rep(-Inf, S); epsilon <- 1e-5
for(s in 2:S){
  B <- Esig2inv^(-1)*Bn
  EbQb <- sum(diag(B%*%solve(Bn))) - t(bn)%*%solve(Bn)%*%bn + t(y)%*%y + t(b0)%*%solve(B0)%*%b0
  d <- EbQb + d0
  Esig2inv <- as.numeric(anVB/d)
  ELBOs[s] <- ELBOfunc(d = d, B = B)
  if (ELBOs[s] < ELBOs[s - 1]) { message("Lower bound decreases!\n")}
  # Check for convergence
  if (ELBOs[s] - ELBOs[s - 1] < epsilon) { break }
  # Check if VB converged in the given maximum iterations
  if (s == S) {warning("VB did not converge!\n")}
}
sig2VBscratch <- MCMCpack::rinvgamma(S,anVB/2,d/2)
BetasVBscratch <- MASS::mvrnorm(S, mu = bnVB, Sigma = B)
summary(coda::mcmc(sig2VBscratch)); summary(coda::mcmc(BetasVBscratch))
```

14.3 Summary

In this chapter, we present approximate methods designed for situations where the likelihood function does not have a closed-form expression (e.g., ABC and BSL), or where the sample size and parameter space are large (e.g., INLA and VI), making traditional MCMC and importance sampling (IS) methods ineffective. We provide the theoretical foundations and include applications to illustrate the potential of these methods.

Simulation-based algorithms suffer from the *curse of dimensionality* in the parameter space, while *optimization-based approaches* typically require explicit evaluation of the likelihood function. To address situations where both challenges arise simultaneously, recent developments, referred to as *hybrid methods*, combine these strategies with the *pseudo-marginal* MCMC approach [13, 273]. For instance, when pseudo-marginal MCMC is combined with variational Bayes, it becomes possible to perform inference in high-dimensional spaces even when the likelihood function is intractable [401].

14.4 Exercises

1. **g-and-k distribution for financial returns continues**

   Simulate a dataset following the specification given in the g-and-k distribution for financial returns example. Set $\theta_1 = 0.8$, $a = 1$, $b = 0.5$, $g = -1$, and $k = 1$, with a sample size of 500, and use the same priors as in that example. Implement the ABC accept/reject algorithm from scratch using one million prior draws, selecting the 1,000 draws with the smallest distance.[7]

   - Perform a linear regression adjustment using the posterior draws of our ABC-AR algorithm (ABC-AR-Adj).
   - Compare the results with those obtained using the ABC-AR implementation in the *EasyABC* package, ensuring that the computational time is relatively similar between both implementations.
   - Compare the posterior results of ABC-AR, ABC-AR-Adj and EasyABC with the population values.

2. **Simulation: g-and-k distribution continues**

[7] Note that this setting does not satisfy the asymptotic requirements for Bayesian consistency. However, it serves as a pedagogical exercise.

Perform the simulation example of the Bayesian synthetic likelihood presented in the book, using the same population parameters and setting $M = 500$, $S = 6{,}000$, with burn-in and thinning parameters set to 1,000 and 5, respectively. Use the *BSL* package in **R** to perform inference using the vanilla, unbiased, semi-parametric, and misspecified (mean and variance) versions of BSL. Compare the posterior distributions of the methods with the true population parameters.

3. Simulate a multinomial logit model (see Section 6.5 in the book) with 3 alternatives, 2 alternative-specific regressors, and 1 individual-specific regressor. The population parameters for the alternative-specific regressors are -0.3 and 1.2, while the population values for the individual-specific regressor are 0.3, 0, and 0.5. All regressors are assumed to follow a standard normal distribution, and the sample size is 1,000.

   Perform inference using the *INLA* package, and note that the Poisson trick should be used for multinomial models in this exercise (see [371] for details).

4. Get the expression for the ELBO in the linear regression model with conjugate family.

5. **Linear regression continues**

   Perform inference in the linear regression example using stochastic variational inference via the automatic differentiation variational inference (ADVI) approach [238], implemented in `rstan` package. In particular, consider the model $y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \mu_i$, assuming non-informative independent priors: $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{B}_0)$ and $\sigma^2 \sim \text{IG}(\alpha_0/2, \delta_0/2)$, where $\boldsymbol{\beta}_0 = \mathbf{0}_3$, $\boldsymbol{B}_0 = 1000\boldsymbol{I}_3$, and $\alpha_0 = \delta_0 = 0.01$. The sample size is one million.

6. Let's retake the mixture regression model of Chapter 11, that is, the simple regression mixture with two components such that $z_{i1} \sim \text{Ber}(0.5)$, consequently, $z_{i2} = 1 - z_{i1}$, and assume one regressor, $x_i \sim N(0, 1)$, $i = 1, 2, \ldots, 1{,}000$. Then,

$$p(y_i \mid \boldsymbol{x}_i) = 0.5\phi(y_i \mid 2 + 1.5x_i, 1^2) + 0.5\phi(y_i \mid -1 + 0.5x_i, 0.8^2).$$

   Let's set $\alpha_{h0} = \delta_{h0} = 0.01$, $\boldsymbol{\beta}_{h0} = \mathbf{0}_2$, $\boldsymbol{B}_{h0} = \boldsymbol{I}_2$, and $\boldsymbol{a}_0 = [1/2\ 1/2]^\top$.

   Perform VB inference in this model using the CAVI algorithm.

Bibliography

[1] Arild Aakvik, James J. Heckman, and Edward J. Vytlacil. Estimating treatment effects for discrete outcomes when responses to treatment vary: An application to norwegian vocational rehabilitation programs. *Journal of Econometrics*, 125(1-2):15–51, 2005.

[2] Alberto Abadie, Joshua D. Angrist, and Guido W. Imbens. Instrumental variables estimates of the effect of subsidized training on the quantiles of trainee earnings. *Econometrica*, 70(1):91–117, 2002.

[3] D. Acemoglu, S. Johnson, and J. Robinson. The colonial origins of comparative development: An empirical investigation. *The American Economic Review*, 91(5):1369–1401, 2001.

[4] J. H. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993.

[5] Jim Albert. *Bayesian Computation with R*. Use R! Springer, New York, NY, 2nd edition, 2009.

[6] Muhammad Amjad, Devavrat Shah, and Dennis Shen. Robust synthetic control. *Journal of Machine Learning Research*, 19(22):1–51, 2018.

[7] Sungbae An and Frank Schorfheide. Bayesian analysis of dsge models. *Econometric reviews*, 26(2-4):113–172, 2007.

[8] Ziwen An, David J Nott, and Christopher Drovandi. Robust bayesian synthetic likelihood via a semi-parametric approach. *Statistics and Computing*, 30(3):543–557, 2020.

[9] Ziwen An, Leah F South, and Christopher Drovandi. Bsl: An r package for efficient parameter estimation for simulation-based models via bayesian synthetic likelihood. *Journal of Statistical Software*, 101:1–33, 2022.

[10] Ziwen An, Leah F South, David J Nott, and Christopher C Drovandi. Accelerating bayesian synthetic likelihood with the graphical lasso. *Journal of Computational and Graphical Statistics*, 28(2):471–475, 2019.

[11] C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[12] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[13] Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient monte carlo computations. *Annals of Statistics*, 37(2):697–725, 2009.

[14] Joshua D. Angrist, Guido W. Imbens, and Donald B. Rubin. Identification of causal effects using instrumental variables. *Journal of the American Statistical Association*, 91(434):444–455, 1996.

[15] Joshua D. Angrist and Jörn-Steffen Pischke. Better late than nothing: Some comments on deaton (2009) and heckman and urzua (2009). *Journal of Economic Literature*, 48(2):399–423, 2010.

[16] Joshua D. Angrist and Jörn-Steffen Pischke. *Mastering 'Metrics: The Path from Cause to Effect*. Princeton University Press, Princeton, NJ, 2014.

[17] Joshua D. Angrist and Jörn-Steffen Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, Princeton, NJ, 2009.

[18] Luc Anselin. A note of small sample properties of estimators in a first order spatial autoregressive model. *Environment and Planning*, 14:1023–1030, 1982.

[19] C. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[20] R. Baath. *Package bayesboot*, 2018.

[21] Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. sgmcmc: An r package for stochastic gradient markov chain monte carlo. *Journal of Statistical Software*, 91:1–27, 2019.

[22] M. Barbieri and J. Berger. Optimal predictive model selection. *The Annals of Statistics*, 32(3):870–897, 2004.

[23] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up markov chain monte carlo: an adaptive subsampling approach. In *International conference on machine learning*, pages 405–413. PMLR, 2014.

[24] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On markov chain monte carlo methods for tall data. *Journal of Machine Learning Research*, 18(47):1–43, 2017.

[25] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[26] Sanjib Basu and Siddhartha Chib. Marginal likelihood and bayes factors for dirichlet process mixture models. *Journal of the American Statistical Association*, 98(461):224–235, 2003.

[27] M. Bayarri and J. Berger. P–values for composite null models. *Journal of American Statistical Association*, 95:1127–1142, 2000.

[28] M. J. Bayarri and J. Berger. The interplay of bayesian and frequentist analysis. *Statistical Science*, 19(1):58–80, 2004.

[29] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–416, 1763.

[30] Thomas Bayes. LII. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, 53:370–418, 1763.

[31] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[32] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[33] Daniel J Benjamin, James O Berger, Magnus Johannesson, Brian A Nosek, E-J Wagenmakers, Richard Berk, Kenneth A Bollen, Björn Brembs, Lawrence Brown, Colin Camerer, et al. Redefine statistical significance. *Nature human behaviour*, 2(1):6–10, 2018.

[34] J. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, third edition edition, 1993.

[35] J. Berger. The case for objective bayesian analysis. *Bayesian Analysis*, 1(3):385–402, 2006.

[36] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013.

[37] James O Berger and Luis R Pericchi. The intrinsic bayes factor for model selection and prediction. *Journal of the American Statistical Association*, 91(433):109–122, 1996.

[38] J. Bernardo and A. Smith. *Bayesian Theory*. Wiley, Chichester, 1994.

[39] José M Bernardo and Adrian FM Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

[40] Giorgio Bertorelle, Andrea Benazzo, and S Mona. ABC as a flexible framework to estimate demography over space and time: Some cons, many pros. *Molecular Ecology*, 19(13):2609–2625, 2010.

[41] Peter J Bickel and Joseph A Yahav. Some contributions to the asymptotic theory of bayes solutions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 11(4):257–276, 1969.

[42] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006.

[43] Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78(5):1103–1130, 2016.

[44] Roger Bivand, Virgilio Gómez-Rubio, and Håvard Rue. Spatial data analysis with r-inla with some extensions. *Journal of statistical software*, 63:1–31, 2015.

[45] D. Blackwell and J. MacQueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, 1:353–355, 1973.

[46] David M Blei and Michael I Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

[47] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[48] Michael GB Blum. Approximate Bayesian computation: a nonparametric perspective. *Journal of the American Statistical Association*, 105(491):1178–1187, 2010.

[49] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[50] Howard D. Bondell and Brian J. Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123, 2008.

[51] G. E. P. Box. Science and statistics. *Journal of the American Statistical Association*, 71:791–799, 1976.

[52] George E. P. Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.

[53] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco, 1st edition, 1976.

[54] George EP Box. Robustness in the strategy of scientific model building. In *Robustness in statistics*, pages 201–236. Elsevier, 1979.

[55] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[56] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[57] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[58] Christoph Breunig, Ruixuan Liu, and Zhengfei Yu. Semiparametric bayesian difference-in-differences. *arXiv preprint arXiv:2412.04605*, 2024.

[59] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[60] Kenneth P Burnham and David R Anderson. Multimodel inference: understanding aic and bic in model selection. *Sociological methods & research*, 33(2):261–304, 2004.

[61] Colin Cameron and Pravin Trivedi. *Microeconometrics: Methods and Applications*. Cambridge, 2005.

[62] Olivier Cappé, Simon J Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[63] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[64] Bradley P Carlin, Alan E Gelfand, and Adrian FM Smith. Hierarchical bayesian analysis of changepoint problems. *Journal of the royal statistical society: series C (applied statistics)*, 41(2):389–405, 1992.

[65] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76:1–32, 2017.

[66] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, 146(1):2–7, 1999.

[67] Chris K Carter and Robert Kohn. On gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

[68] G. Casella and E. Moreno. Objective Bayesian variable selection. *Journal of the American Statistical Association*, 101(473):157–167, 2006.

[69] George Casella and Roger Berger. *Statistical inference*. CRC Press, 2024.

[70] DM Ceperley and Mark Dewing. The penalty method for random walks with uncertain energies. *The Journal of chemical physics*, 110(20):9812–9820, 1999.

[71] Joshua Chan, Gary Koop, Dale J Poirier, and Justin L Tobias. *Bayesian econometric methods*, volume 7. Cambridge University Press, 2019.

[72] Marc K Chan and Akbar Zamanzadeh. Minimum wage and employment in the us: an application of bayesian quantile kink regression. *Econometric Reviews*, pages 1–23, 2025.

[73] W. Chang. *Web Application Framework for R: Package shiny*. R Studio, 2018.

[74] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

[75] V. Chernozhukov and H. Hong. An MCMC approach to classical estimation. *Journal of Econometrics*, 115:293–346, 2003.

[76] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21:C1–C68, 2018.

[77] Victor Chernozhukov and Christian Hansen. The effects of 401(k) participation on the wealth distribution: An instrumental quantile regression analysis. *The Review of Economics and Statistics*, 86(3):735–751, 2004.

[78] Victor Chernozhukov and Christian Hansen. An iv model of quantile treatment effects. *Econometrica*, 73(1):245–261, 2005.

[79] Victor Chernozhukov, Christian Hansen, Nathan Kallus, Martin Spindler, and Vasilis Syrgkanis. Applied causal inference powered by ml and ai. arXiv preprint 2403.02467, arXiv, 2024.

[80] Victor Chernozhukov and Han Hong. An mcmc approach to classical estimation. *Journal of econometrics*, 115(2):293–346, 2003.

[81] S. Chib. Bayes inference in the Tobit censored regression model. *Journal of Econometrics*, 51:79–99, 1992.

[82] S. Chib and B. Carlin. On MCMC sampling in hierarchical longitudinal models. *Statistics and Computing*, 9:17–26, 1999.

[83] Siddhartha Chib. Bayes regression with autoregressive errors: A gibbs sampling approach. *Journal of econometrics*, 58(3):275–294, 1993.

[84] Siddhartha Chib. Marginal likelihood from the gibbs output. *Journal of the american statistical association*, 90(432):1313–1321, 1995.

[85] Siddhartha Chib and Edward Greenberg. Bayes inference in regression models with arma (p, q) errors. *Journal of Econometrics*, 64(1-2):183–206, 1994.

[86] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[87] Siddhartha Chib, Edward Greenberg, and Anna Simoni. Nonparametric bayes analysis of the sharp and fuzzy regression discontinuity designs. *Econometric Theory*, 39(3):481–533, 2023.

[88] Siddhartha Chib and Liana Jacobi. Bayesian fuzzy regression discontinuity analysis and returns to compulsory schooling. *Journal of Applied Econometrics*, 31(6):1026–1047, 2016.

[89] Siddhartha Chib and Ivan Jeliazkov. Marginal likelihood from the metropolis–hastings output. *Journal of the American Statistical Association*, 96(453):270–281, 2001.

[90] Siddhartha Chib and Todd A Kuffner. Bayes factor consistency. *arXiv preprint arXiv:1607.00292*, 2016.

[91] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

[92] Hugh A Chipman, Edward I George, and Robert E McCulloch. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[93] Taeryon Choi and Mark J. Schervish. On posterior consistency in nonparametric regression problems. *Journal of Multivariate Analysis*, 98(10):1969–1987, 2007.

[94] Gerda Claeskens and Nils Lid Hjort. Model selection and model averaging. *Cambridge books*, 2008.

[95] M. Clyde and E. George. Model uncertatinty. *Statistical Science*, 19(1):81–94, 2004.

[96] T. Conley, C. Hansen, and P. Rossi. Plausibly exogenous. *The Review of Economics and Statistics*, 94(1):260–272, 2012.

[97] Timothy G Conley, Christian B Hansen, Robert E McCulloch, and Peter E Rossi. A semi-parametric bayesian approach to the instrumental variable problem. *Journal of Econometrics*, 144(1):276–305, 2008.

[98] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2006.

[99] John G. Cragg and Stephen G. Donald. Testing identifiability and specification in instrumental variable models. *Econometric Theory*, 9(2):222–240, 1993.

[100] Scott Cunningham. *Causal Inference: The Mixtape*. Yale University Press, New Haven, CT, 2021.

[101] Johan Dahlin and Thomas B Schön. Getting started with particle metropolis-hastings for inference in nonlinear dynamical models. *Journal of Statistical Software*, 88:1–41, 2019.

[102] A. P. Dawid, M. Musio, and S. E. Fienberg. From statistical evidence to evidence of causality. *Bayesian Analysis*, 11(3):725–752, 2016.

[103] Bruno De Finetti. Foresight: its logical laws, its subjective sources. In H. E. Kyburg and H. E. Smokler, editors, *Studies in Subjective Probability*. Krieger, New York, 1937. p.55–118.

[104] Piet De Jong and Neil Shephard. The simulation smoother for time series models. *Biometrika*, 82(2):339–350, 1995.

[105] Angus Deaton. Instruments, randomization, and learning about development. *Journal of Economic Literature*, 48(2):424–455, 2010.

[106] M. H. DeGroot. *Probability and statistics*. Addison-Wesley Publishing Co., London, 1975.

[107] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22:1009–1020, 2012.

[108] Marco Del Negro and F. Schorfheide. Forecasting with bayesian var models. In John Geweke, Gary Koop, and Herman van Dijk, editors, *The Oxford Handbook of Bayesian Econometrics*, pages 224–254. Oxford University Press, 2011.

[109] Persi Diaconis and Donald Ylvisaker. Conjugate priors for exponential families. *The Annals of statistics*, 7(2):269–281, 1979.

[110] J. M. Dickey and E. Gunel. Bayes factors from mixed probabilities. *Journal of the Royal Statistical Society: Series B (Methodology)*, 40:43–46, 1978.

[111] James M Dickey. The weighted likelihood ratio, linear hypotheses on normal location parameters. *The Annals of Mathematical Statistics*, pages 204–223, 1971.

[112] Peter. Diggle, P. Heagerty, Liang K-Y., and S. Zeger. *Analysis of longitudinal data*. Oxford University Press, 2002.

[113] Thomas Doan, Robert Litterman, and Christopher Sims. Forecasting and conditional projection using realistic prior distributions. *Econometric reviews*, 3(1):1–100, 1984.

[114] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. *Sequential Monte Carlo methods in practice*, pages 3–14, 2001.

[115] Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[116] Christopher Drovandi and David T Frazier. A comparison of likelihood-free methods with and without summary statistics. *Statistics and Computing*, 32(3):42, 2022.

[117] Christopher C Drovandi and Anthony N Pettitt. Estimation of parameters for macroparasite population evolution using approximate Bayesian computation. *Biometrics*, 67(1):225–233, 2011.

[118] Christopher C Drovandi and Anthony N Pettitt. Likelihood-free bayesian estimation of multivariate quantile distributions. *Computational Statistics & Data Analysis*, 55(9):2541–2556, 2011.

[119] Christopher C Drovandi, Anthony N Pettitt, and Anthony Lee. Bayesian indirect inference using a parametric auxiliary model. *Statistical Science*, 30(1):72—95, 2015.

[120] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[121] Y. D. Edwards and G. M. Allenby. Multivariate analysis of multiple response data. *Journal of Marketing Research*, 40:321–334, 2003.

[122] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7:1–26, 1979.

[123] Bradley Efron and Trevor Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

[124] Bradley Efron and Trevor Hastie. *Computer age statistical inference, student edition: algorithms, evidence, and data science*, volume 6. Cambridge University Press, 2021.

[125] J Paul Elhorst et al. *Spatial econometrics: from cross-sectional data to spatial panels*, volume 479. Springer, 2014.

[126] Walter Enders. *Applied Econometric Time Series*. Wiley, Hoboken, NJ, 4th edition, 2014.

[127] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

[128] T. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[129] Carmen Fernandez, Eduardo Ley, and Mark FJ Steel. Benchmark priors for bayesian model averaging. *Journal of Econometrics*, 100(2):381–427, 2001.

[130] R. Fisher. *Statistical Methods for Research Workers*. Hafner, New York, 13th edition, 1958.

[131] David Frazier, David J Nott, Christopher Drovandi, and Robert Kohn. Bayesian inference using synthetic likelihood: Asymptotics and adjustments. *Journal of the American Statistical Association*, 118(544):2821––2832, 2023.

[132] David T Frazier and Christopher Drovandi. Robust approximate bayesian inference with synthetic likelihood. *Journal of Computational and Graphical Statistics*, 30(4):958–976, 2021.

[133] David T Frazier, Worapree Maneesoonthorn, Gael M Martin, and Brendan PM McCabe. Approximate Bayesian forecasting. *International Journal of Forecasting*, 35(2):521–539, 2019.

[134] David T Frazier, Gael M Martin, Christian P Robert, and Judith Rousseau. Asymptotic properties of approximate Bayesian computation. *Biometrika*, 105(3):593–607, 2018.

[135] David T Frazier, Christian P Robert, and Judith Rousseau. Model misspecification in approximate Bayesian computation: consequences and diagnostics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(2):421–444, 2020.

[136] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[137] Jerome H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–67, 1991.

[138] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[139] Sylvia Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of time series analysis*, 15(2):183–202, 1994.

[140] Sylvia Frühwirth-Schnatter. *Finite mixture and Markov switching models*. Springer, 2006.

[141] Wenjiang J. Fu. Penalized regression: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

[142] George M. Furnival and R. W. Wilson. Regressions by leaps and bounds. *Technometrics*, 16(4):499–511, 1974.

[143] George M. Furnival and Robert W. Wilson. Regressions by leaps and bounds. *Technometrics*, 16(4):499–511, 1974.

[144] Jacob R. Gardner, Geoff Pleiss, David Bindel Wu, Kilian Q. Weinberger, and Andrew Gordon Wilson. Product kernel interpolation for scalable gaussian processes. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84, pages 1407–1416. PMLR, 2018.

[145] P. Garthwaite, J. Kadane, and A. O'Hagan. Statistical methods for eliciting probability distributions. *Journal of American Statistical Association*, 100(470):680–701, 2005.

[146] A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

[147] Alan E Gelfand and Dipak K Dey. Bayesian model choice: asymptotics and exact calculations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(3):501–514, 1994.

[148] A. Gelman and X. Meng. Model checking and model improvement. In Gilks, Richardson, and Speigelhalter, editors, *In Markov chain Monte Carlo in practice*. Springer US, 1996. Chapter 6, pp. 157–196.

[149] A. Gelman, X. Meng, and H. Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pages 733–760, 1996.

[150] Andrew Gelman, John B Carlin, Hal S Stern, David Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2021.

[151] Andrew Gelman et al. Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian analysis*, 1(3):515–534, 2006.

[152] Andrew Gelman, Walter R Gilks, and Gareth O Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.

[153] Andrew Gelman and Guido Imbens. Why ask why? forward causal inference and reverse causal questions. Technical report, National Bureau of Economic Research, 2013.

[154] Andrew Gelman and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992.

[155] S Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[156] E. George and R. McCulloch. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

[157] E. George and R. McCulloch. Approaches for Bayesian variable selection. *Statistica Sinica*, 7:339–373, 1997.

[158] Edward I George and Robert E McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

[159] Edward I George and Robert E McCulloch. Approaches for Bayesian variable selection. *Statistica Sinica*, 7:339–373, 1997.

[160] J. Geweke. *Bayesian Statistics*, chapter Evaluating the accuracy of sampling-based approaches to calculating posterior moments. Clarendon Press, Oxford, UK., 1992.

[161] John Geweke. Using simulation methods for bayesian econometric models: inference, development, and communication. *Econometric reviews*, 18(1):1–73, 1999.

[162] John Geweke. Getting it right: Joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, 99(467):799–804, 2004.

[163] John Geweke. *Contemporary Bayesian econometrics and statistics*, volume 537. John Wiley & Sons, 2005.

[164] John Geweke, Gary Koop, and Herman K van Dijk. *The Oxford handbook of Bayesian econometrics*. Oxford University Press, USA, 2011.

[165] Charles J Geyer. Practical markov chain monte carlo. *Statistical science*, pages 473–483, 1992.

[166] Richard D Gill and James M Robins. Causal inference for complex longitudinal data: the continuous case. *Annals of Statistics*, pages 1785–1811, 2001.

[167] Ryan Giordano, Runjing Liu, Michael I Jordan, and Tamara Broderick. Evaluating sensitivity to the stick-breaking prior in bayesian nonparametrics. *Bayesian Analysis*, 1(1):1–34, 2022.

[168] I. J. Good. The bayes/non bayes compromise: A brief review. *Journal of the American Statistical Association*, 87(419):597–606, September 1992.

[169] S. N. Goodman. Toward evidence-based medical statistics. 1: The P value fallacy. *Annals of internal medicine*, 130(12):995–1004, 1999.

[170] N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993.

[171] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.

[172] David S Greenberg, Marcel Nonnenmacher, and Jakob H Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414, 2019.

[173] Edward Greenberg. *Introduction to Bayesian econometrics*. Cambridge University Press, 2012.

[174] Damodar N. Gujarati and Dawn C. Porter. *Basic Econometrics*. McGraw-Hill Education, New York, NY, 5th edition, 2009.

[175] Paul Gustafson. Local robustness in bayesian analysis. In *Robust Bayesian Analysis*, pages 71–88. Springer, 2000.

[176] Trygve Haavelmo. The statistical implications of a system of simultaneous equations. *Econometrica*, 11(1):1–12, 1943.

[177] P Richard Hahn, Jared S Murray, and Carlos M Carvalho. Bayesian regression tree models for causal inference: Regularization, confounding, and heterogeneous effects (with discussion). *Bayesian Analysis*, 15(3):965–1056, 2020.

[178] Robert Haining. *Spatial data analysis in the social and environmental sciences*. Cambridge University Press, United of Kingdom, first edition, 1990.

[179] Johannes Edmund Handschin and David Q Mayne. Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International journal of control*, 9(5):547–559, 1969.

[180] Lars Peter Hansen. Large sample properties of generalized method of moments estimators. *Econometrica*, 50(4):1029–1054, 1982.

[181] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50(2):949, 2022.

[182] Trevor Hastie and Robert Tibshirani. Bayesian backfitting (with discussion). *Journal of the American Statistical Association*, 95(452):1228–1240, 2000.

[183] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2 edition, 2009.

[184] W. Hastings. Monte Carlo sampling methods using Markov chains and their application. *Biometrika*, 57:97–109, 1970.

[185] Fumio Hayashi. *Econometrics*. Princeton University Press, 2000.

[186] James J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–161, 1979.

[187] James J. Heckman, Hedibert F. Lopes, and Rémi Piatek. Treatment effects: A bayesian perspective. *Econometric Reviews*, 33(1-4):36–67, 2014.

[188] P. Heidelberger and P. D. Welch. Simulation run length control in the presence of an initial transient. *Operations Research*, 31(6):1109–1144, 1983.

[189] Lütkepohl Helmut. *New introduction to multiple time series analysis*. Springer, 2005.

[190] Miguel A. Hernán and James M. Robins. *Causal Inference: What If*. Chapman & Hall/CRC, Boca Raton, FL, 2020.

[191] Jennifer L Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.

[192] Keisuke Hirano, Guido W. Imbens, Donald B. Rubin, and Xiaohong Zhou. Assessing the effect of an influenza vaccine in an encouragement design. *Biostatistics*, 1(1):69–88, 2000.

[193] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[194] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

[195] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *the Journal of machine Learning research*, 14(1):1303–1347, 2013.

[196] Paul W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960, 1986.

[197] Daniel Hosszejni and Gregor Kastner. Modeling univariate and multivariate stochastic volatility in r with `stochvol` and `factorstochvol`. *Journal of Statistical Software*, 100(12):1–34, 2021.

[198] Zhen Huang and Andrew Gelman. Sampling for bayesian computation with large datasets. Technical report, Department of Statistics, Columbia University, 2005. Technical Report, Department of Statistics, Columbia University.

[199] Leonardo Iacovone, David J McKenzie, and Rachael Meager. Bayesian impact evaluation with informative priors: An application to a colombian management and export improvement program. Technical report, World Bank, 2023.

[200] Joseph G. Ibrahim and Purushottam W. Laud. On bayesian analysis of generalized linear models using jeffreys's prior. *Journal of the American Statistical Association*, 86(416):981–986, 1991.

[201] Guido W. Imbens. Instrumental variables: An econometrician's perspective. *Statistical Science*, 29(3):323–358, 2014. Also IZA Discussion Paper No. 8048 and NBER Working Paper No. 19983.

[202] Guido W. Imbens and Joshua D. Angrist. Identification and estimation of local average treatment effects. *Econometrica*, 62(2):467–475, 1994.

[203] Guido W Imbens and Donald B Rubin. Bayesian inference for causal effects in randomized experiments with noncompliance. *The annals of statistics*, pages 305–327, 1997.

[204] Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction.* Cambridge University Press, New York, NY, 2015.

[205] H. Ishwaran and J. S. Rao. Spike and slab variable selection: Frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.

[206] Liana Jacobi, Chun Fung Kwok, Andrés Ramírez-Hassan, and Nhung Nghiem. Posterior manifolds over prior parameter regions: Beyond pointwise sensitivity assessments for posterior statistics from mcmc inference. *Studies in Nonlinear Dynamics & Econometrics*, 28(2):403–434, 2024.

[207] Liana Jacobi, Dan Zhu, and Mark Joshi. Estimating posterior sensitivities with application to structural analysis of bayesian vector autoregressions, 2022.

[208] H. Jeffreys. Some test of significance, treated by the theory of probability. *Proccedings of the Cambridge philosophy society*, 31:203–222, 1935.

[209] H. Jeffreys. *Theory of Probability*. Oxford University Press, London, 1961.

[210] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946.

[211] M. Jetter and A. Ramírez Hassan. Want export diversification? Educate the kids first. *Economic Inquiry*, 53(4):1765–1782, 2015.

[212] Michael Jetter, Rafat Mahmood, Christopher F. Parmeter, and Andrés Ramírez-Hassan. Post-cold war civil conflict and the role of history and religion: A stochastic search variable selection approach. *Economic Modelling*, 114:105907, 2022.

[213] V. E. Jhonson and D. Rossell. Bayesian model selection in high-dimensional settings. *Journal of the American Statistical Association*, 107(498):649–660, 2012.

[214] Bai Jiang. Approximate bayesian computation with kullback-leibler divergence as data discrepancy. In *International conference on artificial intelligence and statistics*, pages 1711–1721. PMLR, 2018.

[215] Valen E Johnson and David Rossell. Bayesian model selection in high-dimensional settings. *Journal of the American Statistical Association*, 107(498):649–660, 2012.

[216] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[217] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

[218] J. B. Kadane. Predictive and structural methods for eliciting prior distributions. In A Zellner, editor, *Bayesian Analysis in Econometrics and Statistics: Essays in honor of Harold Jeffreys,*, pages 89–93. North–Holland Publishing Company,, Amsterdam, 1980.

[219] Joseph Kadane and Lara Wolfson. Experiences in elitation. *The Statiscian*, 47(1):3–19, 1998.

[220] Daniel Kahneman. *Thinking, fast and slow.* Macmillan, 2011.

[221] N. Kantas, A. Doucet, S. S. Singh, and J. M. Maciejowski. An overview of sequential monte carlo methods for parameter estimation in general state–space models. *IFAC Proceedings Volumes*, 42(10):774–785, 2009.

[222] Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

[223] G. Karabatsos. A menu-driven software package of Bayesian nonparametric (and parametric) mixed models for regression analysis and density estimation. *Behavior Research Methods*, 49:335–362, 2016.

[224] A. Karl and A. Lenkoski. Instrumental variable Bayesian model averaging via conditional Bayes factor. Technical report, Heidelberg University, 2012.

[225] R. Kass. Statistical inference: the big picture. *Statistical science*, 26(1):1–9, 2011.

[226] Robert E. Kass and Adrian E. Raftery. Bayes factorss. *Journal of American Statistical Association*, 90(430):773–795, 1995.

[227] Gregor Kastner and Sylvia Frühwirth-Schnatter. Ancillarity-sufficiency interweaving strategy (asis) for boosting mcmc estimation of stochastic volatility models. *Computational Statistics & Data Analysis*, 76:408–423, 2014.

[228] Chang-Jin Kim and Charles R Nelson. Has the us economy become more stable? a bayesian approach based on a markov-switching model of the business cycle. *Review of Economics and Statistics*, 81(4):608–616, 1999.

[229] Sungjin Kim, Clarence Lee, and Sachin Gupta. Bayesian synthetic control methods. *Journal of Marketing Research*, 57(5):831–852, 2020.

[230] Frank Kleibergen and Richard Paap. Generalized reduced rank tests using the singular value decomposition. *Journal of Econometrics*, 133(1):97–126, 2006.

[231] Augustine Kong, Jun S Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

[232] G Koop, R León-Gonzalez, and R Strachan. Bayesian model averaging in the instrumental variable regression model. *Journal of Econometrics*, 171:237–250, 2012.

[233] Gary Koop, Dimitris Korobilis, et al. Bayesian multivariate time series methods for empirical macroeconomics. *Foundations and Trends® in Econometrics*, 3(4):267–358, 2010.

[234] Gary M Koop. *Bayesian econometrics*. John Wiley & Sons Inc., 2003.

[235] Julia Kowalska, Mark van de Wiel, and StÃŠphanie van der Pas. Bayesian regression discontinuity design with unknown cutoff. *arXiv preprint arXiv:2406.11585*, 2024.

[236] Hideo Kozumi and Genya Kobayashi. Gibbs sampling methods for Bayesian quantile regression. *Journal of Statistical Computation and Simulation*, 81(11):1565—-1578, 2011.

[237] Fabian Krueger. *bvarsv: Bayesian Analysis of a Vector Autoregressive Model with Stochastic Volatility and Time-Varying Parameters*, 2022. R package version 1.1.

[238] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of machine learning research*, 18(14):1–45, 2017.

[239] Minjung Kyung, Jeff Gill, Malay Ghosh, and George Casella. Penalized regression, standard errors, and bayesian lassos. *Bayesian Analysis*, 5(2):369–411, 2010.

[240] Tony Lancaster. *An introduction to modern Bayesian econometrics*. Blackwell Oxford, 2004.

[241] P. Laplace. *Théorie Analytique des Probabilités*. Courcier, 1812.

[242] Pierre Simon Laplace. Mémoire sur la probabilité de causes par les évenements. *Mémoire de l'académie royale des sciences*, 1774.

[243] E.L. Lehmann and George Casella. *Theory of Point Estimation*. Springer, second edition edition, 2003.

[244] Alex Lenkoski, Theo S. Eicher, and Adrian Raftery. Two-stage Bayesian model averaging in endogeneous variable models. *Econometric Reviews*, 33, 2014.

[245] Alex Lenkoski, Anna Karl, and Andreas Neudecker. *Package ivbma*, 2013.

[246] Maxime Lenormand, Franck Jabot, and Guillaume Deffuant. Adaptive approximate Bayesian computation for complex models. *Computational Statistics*, 28(6):2777–2796, 2013.

[247] James LeSage and Robert Kelley Pace. *Introduction to spatial econometrics*. Chapman and Hall/CRC, 2009.

[248] Christoph Leuenberger and Daniel Wegmann. Bayesian computation and model selection without likelihoods. *Genetics*, 184(1):243–252, 2010.

[249] Arthur Lewbel and Krishna Pendakur. Tricks with hicks: The easi demand system. *American Economic Review*, 99(3):827–863, 2009.

[250] Eduardo Ley and Mark FJ Steel. On the effect of prior assumptions in Bayesian model averaging with applications to growth regression. *Journal of Applied Econometrics*, 24(4):651–674, 2009.

[251] Wei Li, Sangwoo Ahn, and Max Welling. Scalable mcmc for mixed membership stochastic blockmodels. In JMLR Workshop and Conference Proceedings, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 51, pages 723–731, 2016.

[252] D. V. Lindley. The philosophy of statistics. *The Statistician*, 49(3):293–337, 2000.

[253] D. V. Lindley and L. D. Phillips. Inference for a Bernoulli process (a Bayesian view). *American Statistician*, 30:112–119, 1976.

[254] Dennis V Lindley. A statistical paradox. *Biometrika*, 44(1/2):187–192, 1957.

[255] Antonio R Linero. Bayesian regression trees for high-dimensional prediction and variable selection. *Journal of the American Statistical Association*, 113(522):626–636, 2018.

[256] Jarno Lintusaari, Michael U Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic biology*, 66(1):e66–e82, 2017.

[257] Robert B Litterman. Forecasting with bayesian vector autoregressions—five years of experience. *Journal of Business & Economic Statistics*, 4(1):25–38, 1986.

[258] J. S. Liu and R. Chen. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576, 1995.

[259] Jan-Matthis Lueckmann, Pedro J Goncalves, Gabriele Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems*, 30, 2017.

[260] Gruber Luis and Kastner Gregor. *bayesianVARs: MCMC Estimation of Bayesian Vectorautoregressions*, 2024. R package version 1.0.5.

[261] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2917–2925, 2015.

[262] Dougal Maclaurin and Ryan P. Adams. Firefly monte carlo: Exact mcmc with subsets of data. In *The Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 4279–4295. IJCAI, 2015.

[263] D. Madigan and A. E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546, 1994.

[264] D. Madigan, J. C. York, and D. Allard. Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232, 1995.

[265] David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.

[266] Charles F. Manski. Nonparametric bounds on treatment effects. *American Economic Review*, 80(2):319–323, 1990.

[267] Charles F. Manski. *Identification Problems in the Social Sciences*. Harvard University Press, Cambridge, MA, 1995.

[268] Charles F. Manski. *Partial Identification of Probability Distributions*. Springer, New York, 2003.

[269] Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the USA*, 100(26):15324–15328, 2003.

[270] Jacob Marschak and William H. Andrews. Random simultaneous equations and the theory of production. *Econometrica*, 12(3-4):143–205, 1944.

[271] Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. MCMCpack: Markov chain Monte Carlo in R. *Journal of Statistical Software*, 42(9):1–21, 2011.

[272] Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. *Package MCMCpack*, 2018.

[273] Gael M Martin, David T Frazier, and Christian P Robert. Approximating bayes in the 21st century. *Statistical Science*, 39(1):20–45, 2024.

[274] Gael M Martin, David T Frazier, and Christian P Robert. Computing bayes: From then 'til now. *Statistical Science*, 39(1):3–19, 2024.

[275] Gael M Martin, Rubén Loaiza-Maya, Worapree Maneesoonthorn, David T Frazier, and Andrés Ramírez-Hassan. Optimal probabilistic forecasts: When do they work? *International Journal of Forecasting*, 38(1):384–406, 2022.

[276] Osvaldo A. Martin, Ravin Kumar, and Junpeng Lao. *Bayesian Modeling and Computation in Python*. publisher=Chapman and Hall/CRC, Boca Raton, December 2021.

[277] Sara Martino and Andrea Riebler. Integrated nested laplace approximations (inla). *arXiv preprint arXiv:1907.01248*, 2019.

[278] Tyler H McCormick, Adrian E Raftery, David Madigan, and Randall S Burd. Dynamic logistic regression and dynamic model averaging for binary classification. *Biometrics*, 68(1):23–30, 2012.

[279] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. London: Chapman and Hall, 1989.

[280] R. McCulloch and P. Rossi. An exact likelihood analysis of the multinomial probit model. *Journal of Econometrics*, 64:207–240, 1994.

[281] Robert E McCulloch, Nicholas G Polson, and Peter E Rossi. A bayesian analysis of the multinomial probit model with fully identified parameters. *Journal of econometrics*, 99(1):173–193, 2000.

[282] Sharon Bertsch McGrayne. *The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, & Emerged Triumphant from Two Centuries of C*. Yale University Press, 2011.

[283] Karel Mertens and Morten O Ravn. A reconciliation of svar and narrative estimates of tax multipliers. *Journal of Monetary Economics*, 68:S1–S19, 2014.

[284] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys*, 21:1087–1092, 1953.

[285] J. Miller and M. Harrison. Inconsistency of Pitman-Yor process mixtures for the number of components. *Journal of Machine Learning Research*, 15:3333–3370, 2014.

[286] Stanislav Minsker. Scalable and robust bayesian inference via the median posterior. *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 37:1656–1664, 2015.

[287] Stanislav Minsker, Subhabrata Srivastava, Lin Lin, and David B. Dunson. Robust and scalable bayes via a median of subset posterior measures. *Journal of Machine Learning Research*, 18(1):4488–4527, 2017.

[288] P. Müller, F. Quintana, A. Jara, and T. Hanson. *Bayesian Nonparametric Data Analysis*. Springer, New York, 2015.

[289] Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, 1996.

[290] Radford M. Neal. Mcmc using hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, chapter 5, pages 113–162. Chapman and Hall/CRC, 2011.

[291] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[292] Willie Neiswanger, Chong Wang, and Eric P Xing. Asymptotically exact, embarrassingly parallel mcmc. In *Proceedings of the Thirtieth International Conference on Machine Learning*, pages 623–632. PMLR, 2013.

[293] J.A. Nelder and R.W.M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[294] Christopher Nemeth and Paul Fearnhead. Stochastic gradient markov chain monte carlo. *Journal of the American Statistical Association*, 116(533):433–450, 2021.

[295] J. Neyman and E. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society, Series A*, 231:289–337, 1933.

[296] Duy Nguyen. An in depth introduction to variational bayes note. *Available at SSRN 4541076*, 2023.

[297] Kuschnig Nikolas, Vashold Lukas, Tomass Nirai, McCracken Michael, and Ng Serena. *BVAR: Hierarchical Bayesian Vector Autoregression*, 2022. R package version 1.0.5.

[298] Agostino Nobile. Comment: Bayesian multinomial probit models with a normalization constraint. *Journal of Econometrics*, 99(2):335–345, 2000.

[299] James Normington, Eric Lock, Caroline Carlin, Kevin Peterson, and Bradley Carlin. A bayesian difference-in-difference framework for the impact of primary care redesign on diabetes outcomes. *Statistics and Public Policy*, 6(1):55–66, 2019.

[300] James P Normington, Eric F Lock, Thomas A Murray, and Caroline S Carlin. Bayesian variable selection in hierarchical difference-in-differences models. *Statistical methods in medical research*, 31(1):169–183, 2022.

[301] Robert B. O'Hara and Mikko J. Sillanpää. A review of bayesian variable selection methods: What, how and which. *Bayesian Analysis*, 4(1):85–118, 2009.

[302] K. Ord. Estimation methods for models of spatial interaction. *Journal of the American Statistical Association*, 70(349):120–126, 1975.

[303] George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[304] T. Park and G. Casella. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[305] G. Parmigiani and L. Inoue. *Decision theory principles and approaches*. John Wiley & Sons, 2008.

[306] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–710, 1995.

[307] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, New York, 2018.

[308] Luis Pericchi and Carlos Pereira. Adaptative significance levels using optimal decision rules: Balancing by weighting the error probabilities. *Brazilian Journal of Probability and Statistics*, 2015.

[309] Giovanni Petris, Sonia Petrone, and Patrizia Campagnoli. Dynamic linear models. In *Dynamic Linear Models with R*, pages 31–84. Springer, 2009.

[310] Michael K Pitt, Ralph dos Santos Silva, Paolo Giordani, and Robert Kohn. On some properties of markov chain monte carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

[311] Geoff Pleiss, Jacob R. Gardner, Kilian Q. Weinberger, and Andrew Gordon Wilson. Constant-time predictive distributions for gaussian processes. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 4111–4120. PMLR, 2018.

[312] Martyn Plummer, Nicky Best, Kate Cowles, Karen Vines, Deepayan Sarkar, Douglas Bates, Russell Almond, and Arni Magnusson. *Output Analysis and Diagnostics for MCMC*, 2016.

[313] Andy Pole, Mike West, and Jeff Harrison. *Applied Bayesian forecasting and time series analysis*. Chapman and Hall/CRC, 2018.

[314] Leah F Price, Christopher C Drovandi, Anthony Lee, and David J Nott. Bayesian synthetic likelihood. *Journal of Computational and Graphical Statistics*, 27(1):1–11, 2018.

[315] Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun, and Marcus W Feldman. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798, 1999.

[316] Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up mcmc by efficient data subsampling. *Journal of the American Statistical Association*, 2019.

[317] Matias Quiroz, Mattias Villani, Robert Kohn, Minh-Ngoc Tran, and Khue-Dung Dang. Subsampling mcmc-an introduction for the survey statistician. *Sankhya: The Indian Journal of Statistics, Series A*, 80:33–69, 2018.

[318] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[319] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023.

[320] A. Raftery. Bayesian model selection in social research. *Sociological Methodology*, 25:111–163, 1995.

[321] Adrian Raftery, Jennifer Hoeting, Chris Volinsky, Ian Painter, and Ka Yee Yeung. *Package BMA*, 2012.

[322] Adrian E Raftery. Bayesian model selection in structural equation models. *Sage Focus Editions*, 154:163–163, 1993.

[323] Adrian E Raftery, Miroslav Kárnỳ, and Pavel Ettler. Online prediction under model uncertainty via dynamic model averaging: Application to a cold rolling mill. *Technometrics*, 52(1):52–66, 2010.

[324] Adrian E. Raftery, David Madigan, and Jennifer A. Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92(437):179–191, 1997.

[325] A.E. Raftery and S.M. Lewis. One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. *Statistical Science*, 7:493–497, 1992.

[326] A. Ramírez Hassan. The interplay between the Bayesian and frequentist approaches: A general nesting spatial panel data model. *Spatial Economic Analysis*, 12(1):92–112, 2017.

[327] A. Ramírez Hassan, J. Cardona Jiménez, and R. Cadavid Montoya. The impact of subsidized health insurance on the poor in Colombia: Evaluating the case of Medellín. *Economia Aplicada*, 17(4):543–556, 2013.

[328] A. Ramírez-Hassan and R. Guerra-Urzola. Bayesian treatment effects due to a subsidized health program: The case of preventive health care utilization in medellín (Colombia). *Empirical Economics*, 60:1477–1506, 2021.

[329] Andrés Ramírez-Hassan. Dynamic variable selection in dynamic logistic regression: an application to internet subscription. *Empirical Economics*, 59(2):909–932, 2020.

[330] Andrés Ramírez-Hassan and Daniela A. Carvajal-Rendón. Specification uncertainty in modeling internet adoption: A developing city case analysis. *Utilities Policy*, 70:101218, 2021.

[331] Andrés Ramírez-Hassan and David T. Frazier. Testing model specification in approximate bayesian computation using asymptotic properties. *Journal of Computational and Graphical Statistics*, 33(3):1–14, 2024.

[332] Andrés Ramírez-Hassan and Alejandro López-Vera. Welfare implications of a tax on electricity: A semi-parametric specification of the incomplete easi demand system. *Energy Economics*, 131:1–13, 2024.

[333] Andrés Ramírez Hassan and Santiago Montoya Blandón. Welfare gains of the poor: An endogenous bayesian approach with spatial random effects. *Econometric Reviews*, 38(3):301–318, 2019.

[334] F. Ramsey. Truth and probability. In Routledge and Kegan Paul, editors, *The Foundations of Mathematics and other Logical Essays*. New York: Harcourt, Brace and Company, London, 1926. Ch. VII, p.156–198.

[335] A. Ramírez-Hassan and M. Graciano-Londoño. A GUIded tour of Bayesian regression. *The R Journal*, 13(2):135–152, 2020.

[336] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[337] Glen D Rayner and Helen L MacGillivray. Numerical maximum likelihood estimation for the g-and-k and generalized g-and-h distributions. *Statistics and Computing*, 12(1):57–75, 2002.

[338] Lewis J Rendell, Adam M Johansen, Anthony Lee, and Nick Whiteley. Global consensus monte carlo. *Journal of Computational and Graphical Statistics*, 30(2):249–259, 2020.

[339] Silvio R Rendon. Fixed and random effects in classical and bayesian regression. *Oxford Bulletin of Economics and Statistics*, 75(3):460–476, 2013.

[340] Sylvia Richardson and Peter J Green. On bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: series B (statistical methodology)*, 59(4):731–792, 1997.

[341] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[342] Christian P Robert and George Casella. *Introducing monte carlo methods with R*, volume 18. Springer, 2010.

[343] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2nd edition, 2011.

[344] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[345] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[346] Veronika Ročková. On semi-parametric inference for BART. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8137–8146. PMLR, 2020.

[347] Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[348] P. Rossi. *Package bayesm*, 2017.

[349] Peter E. Rossi. *Bayesian Non- and Semi-Parametric Methods and Applications*. Princeton University Press, Princeton, NJ, 2014.

[350] Peter E Rossi, Greg M Allenby, and Rob McCulloch. *Bayesian statistics and marketing*. John Wiley & Sons, 2012.

[351] Veronika Ročková and Edward I. George. The spike-and-slab lasso. *Journal of the American Statistical Association*, 113(521):431–444, 2018.

[352] Veronika Ročková and Enakshi Saha. On theory for bart. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2839–2848. PMLR, 2019.

[353] Veronika Ročková and Stephanie van der Pas. Posterior concentration for bayesian regression trees and forests. *The Annals of Statistics*, 48(4):2103–2130, 2020.

[354] Donald B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974.

[355] Donald B. Rubin. Bayesian inference for causal effects: The role of randomization. *The Annals of Statistics*, 6(1):34–58, 1978.

[356] Donald B Rubin. Bayesianly justifiable and relevant frequency calculations for the applies statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

[357] Donald B. Rubin. Using the sir algorithm to simulate posterior distributions. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 3*, pages 395–402. Oxford University Press, 1988.

[358] Donald B. Rubin. Neyman (1923) and causal inference in experiments and observational studies. *Statistical Science*, 5(4):472–480, 1990.

[359] Donald B. Rubin. Teaching statistical inference for causal effects in experiments and observational studies. *Journal of Educational and Behavioral Statistics*, 29(3):343–367, 2004.

[360] Donnald B. Rubin. The Bayesian bootstrap. *The Annals of Statistics*, 9(1):130–134, 1981.

[361] Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(2):319–392, 2009.

[362] Håvard Rue, Andrea Riebler, Sigrunn H Sørbye, Janine B Illian, Daniel P Simpson, and Finn K Lindgren. Bayesian computing with inla: a review. *Annual Review of Statistics and Its Application*, 4(1):395–421, 2017.

[363] John Denis Sargan. The estimation of economic relationships using instrumental variables. *Econometrica*, 26(3):393–415, 1958.

[364] L. J. Savage. *The foundations of statistics*. John Wiley & Sons, Inc., New York, 1954.

[365] Robert Schlaifer and Howard Raiffa. *Applied statistical decision theory*. Wiley New York, 1961.

[366] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[367] Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayes and big data: The consensus monte carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, 2016.

[368] Steven L Scott, Alexander W Blocker, Fernando V Bonassi, Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayes and big data: The consensus monte carlo algorithm. In *Big Data and Information Theory*, pages 8–18. Routledge, 2022.

[369] Thomas Sellke, MJ Bayarri, and James O Berger. Calibration of p values for testing precise null hypotheses. *The American Statistician*, 55(1):62–71, 2001.

[370] Steve Selvin. A problem in probability (letter to the editor). *The American Statistician*, 29(1):67–71, 1975.

[371] Francesco Serafini. Multinomial logit models with inla. *R-INLA Tutorial. https://inla.r-inla-download.org/r-inla.org/doc/vignettes/multinomial.pdf*, 2019.

[372] M. Serna Rodríguez, A. Ramírez Hassan, and A. Coad. Uncovering value drivers of high performance soccer players. *Journal of Sport Economics*, 20(6):819–849, 2019.

[373] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[374] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[375] Neil Shephard. Partial non-gaussian state space. *Biometrika*, 81(1):115–131, 1994.

[376] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, Cham, Switzerland, 4th edition, 2017.

[377] Susan J Simmons, Fang Fang, Qijun Fang, and Karl Ricanek. Markov chain Monte Carlo model composition search strategy for quantitative trait loci in a Bayesian hierarchical model. *World Academy of Science, Engineering and Technology*, 63:58–61, 2010.

[378] Christopher A Sims. Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, pages 1–48, 1980.

[379] Sisson, Scott A, Fan, Yanan, Beaumont, and Mark. *Handbook of Approximate Bayesian Computation*. CRC Press, 2018.

[380] Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the USA*, 104(6):1760–1765, 2007.

[381] A. F. M. Smith. A General Bayesian Linear Model. *Journal of the Royal Statistical Society. Series B (Methodological).*, 35(1):67–75, 1973.

[382] Adrian FM Smith and Alan E Gelfand. Bayesian statistics without tears: a sampling–resampling perspective. *The American Statistician*, 46(2):84–88, 1992.

[383] Ercan Solak, Roderick Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen. Derivative observations in gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems*, pages 1033–1040. MIT Press, 2003.

[384] Qifan Song, Yan Sun, Mao Ye, and Faming Liang. Extended stochastic gradient markov chain monte carlo for large-scale bayesian variable selection. *Biometrika*, 107(4):997–1004, 2020.

[385] Rodney A. Sparapani, Charles Spanbauer, and Robert McCulloch. Nonparametric machine learning and efficient computation with bayesian additive regression trees: the bart r package. *Journal of Statistical Software*, 97(1):1–66, 2021.

[386] David J Spiegelhalter, Nicola G Best, Bradley P Carlin, and Angelika Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the royal statistical society: Series b (statistical methodology)*, 64(4):583–639, 2002.

[387] Douglas Staiger and James H. Stock. Instrumental variables regression with weak instruments. *Econometrica*, 65(3):557–586, 1997.

[388] Stan Development Team. shinystan: Interactive visual and numerical diagnostics and posterior analysis for Bayesian models., 2017. R package version 2.3.0.

[389] Stan Development Team. Stan modeling language users guide and reference manual, 2024., 2024.

[390] Stephen Stigler. Richard price, the first bayesian. *Statistical Science*, 33(1):117–125, 2018.

[391] James H. Stock and Motohiro Yogo. Asymptotic distributions of instrumental variables statistics with many instruments. In Donald W. K. Andrews and James H. Stock, editors, *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, pages 109–120. Cambridge University Press, 2005.

[392] Andrew M. Stuart and Aretha L. Teckentrup. Posterior consistency for gaussian process approximations of bayesian posterior distributions. *Mathematics of Computation*, 87(310):721–753, 2018.

[393] M. A. Tanner and W. H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540, 1987.

[394] Simon Tavaré, David J Balding, Robert C Griffiths, and Peter Donnelly. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2):505–518, 1997.

[395] Yee Whye Teh, Alexandre H. Thiery, and Sebastian J. Vollmer. Consistency and fluctuations for stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17(1):193–225, 2016.

[396] Samuel Thomas and Wanzhu Tu. Learning hamiltonian monte carlo in r. *The American Statistician*, 75(4):403–413, 2021.

[397] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[398] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, pages 1701–1728, 1994.

[399] Luke Tierney and Joseph B Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

[400] Luke Tierney, Robert E Kass, and Joseph B Kadane. Fully exponential laplace approximations to expectations and variances of nonpositive functions. *Journal of the american statistical association*, 84(407):710–716, 1989.

[401] Minh-Ngoc Tran, David J. Nott, and Robert Kohn. Variational bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 26(4):873–882, 2017.

[402] Regina Tüchler. Bayesian variable selection for logistic models using auxiliary mixture sampling. *Journal of Computational and Graphical Statistics*, 17(1):76–94, 2008.

[403] A. Tversky and D. Kahneman. Judgement under uncertainty: heuristics and biases. *Science*, 185:1124–1131, 1974.

[404] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

[405] Martijn Van Hasselt. Bayesian inference in a sample selection model. *Journal of Econometrics*, 165(2):221–232, 2011.

[406] Isabella Verdinelli and Larry Wasserman. Computing bayes factors using a generalization of the savage-dickey density ratio. *Journal of the American Statistical Association*, 90(430):614–618, 1995.

[407] Sebastian J. Vollmer, Konstantinos C. Zygalakis, and Yee Whye Teh. Exploration of the (non-)asymptotic bias and variance of stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17(159):1–48, 2016.

[408] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[409] Stephen G Walker. Modern bayesian asymptotics. *Statistical Science*, pages 111–117, 2004.

[410] Stephen G. Walker. New approaches to bayesian consistency. *Annals of Statistics*, 32(5):2028–2043, 2004.

[411] Xiangyu Wang and David B Dunson. Parallelizing mcmc via weierstrass sampler. *arXiv preprint arXiv:1312.4605*, 2013.

[412] Ronald L. Wasserstein and Nicole A. Lazar. The ASA's statement on p–values: context, process and purpose. *The American Statistician*, 2016.

[413] Daniel Wegmann, Christoph Leuenberger, and Laurent Excoffier. Efficient approximate Bayesian computation coupled with Markov chain Monte Carlo without likelihood. *Genetics*, 182(4):1207–1218, 2009.

[414] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[415] Mike West and Jeff Harrison. *Bayesian forecasting and dynamic models*. Springer Science & Business Media, 2006.

[416] Andrew G. Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pages 1775–1784. PMLR, 2015.

[417] R. Winkelmann. Health care reform and the number of doctor visits - An econometric analysis. *Journal of Applied Econometrics*, 19(4):455–472, 2004.

[418] Simon N Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.

[419] P. Woodward. BugsXLA: Bayes for the common man. *Journal of Statistical Software*, 14(5):1–18, 2005.

[420] Jeffrey M Wooldridge. *Econometric analysis of cross section and panel data*. MIT press, 2010.

[421] Jeffrey M. Wooldridge. *Introductory Econometrics: A Modern Approach*. Cengage Learning, Mason, Ohio: South-Western, fifth edition, 2012.

[422] Jeffrey M. Wooldridge. *Introductory Econometrics: A Modern Approach*. Cengage Learning, Boston, MA, 6th edition, 2016.

[423] Tomasz Woźniak. Bayesian vector autoregressions. *Australian Economic Review*, 49(3):365–380, 2016.

[424] Tomasz Woźniak. *bsvars: Bayesian Estimation of Structural Vector Autoregressive Models*, 2024. R package version 3.1.

[425] Changye Wu and Christian P Robert. Average of recentered parallel mcmc for big data. *arXiv preprint arXiv:1706.04780*, 2017.

[426] Wang Xiaolei and Tomasz Woźniak. *bsvarSIGNs: Bayesian SVARs with Sign, Zero, and Narrative Restrictions*, 2024. R package version 1.0.1.

[427] Yuling Yao. Bayesian aggregation. In *Wiley StatsRef: Statistics Reference Online*, pages 1–13. John Wiley & Sons, Ltd, 2021.

[428] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Using stacking to average bayesian predictive distributions. *Bayesian Analysis*, 13(3):917–1003, 2018.

[429] A. Zellner. *Introduction to Bayesian inference in econometrics*. John Wiley & Sons Inc., 1996.

[430] Arnold Zellner. On assessing prior distributions and bayesian regression analysis with g-prior distributions. *Bayesian inference and decision techniques*, 1986.

[431] Fengshuo Zhang and Chao Gao. Convergence rates of variational posterior distributions. *The Annals of Statistics*, 48(4):2180–2207, 2020.

[432] S. Ziliak. Guinnessometrics; the Economic Foundation of student's t. *Journal of Economic Perspectives*, 22(4):199–216, 2008.

[433] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix

TABLE 14.1
Libraries and commands in BEsmarter GUI.

Univariate models			
Model	**Library**	**Command**	**Reference**
Normal	MCMCpack	MCMCregress	[272]
Logit	MCMCpack	MCMClogit	[272]
Probit	bayesm	rbprobitGibbs	[348]
Multinomial(Mixed) Probit	bayesm	rmnpGibbs	[348]
Multinomial(Mixed) Logit	bayesm	rmnlIndepMetrop	[348]
Ordered Probit	bayesm	rordprobitGibbs	[348]
Negative Binomial(Poisson)	bayesm	rnegbinRw	[348]
Tobit	MCMCpack	MCMCtobit	[272]
Quantile	MCMCpack	MCMCquantreg	[272]
Bayesian bootstrap	bayesboot	bayesboot	[20]
Multivariate models			
Model	**Library**	**Command**	**Reference**
Multivariate	bayesm	rmultireg	[348]
Seemingly Unrelated Regression	bayesm	rsurGibbs	[348]
Instrumental Variable	bayesm	rivGibbs	[348]
Bivariate Probit	bayesm	rmvpGibbs	[348]
Hierarchical longitudinal models			
Model	**Library**	**Command**	**Reference**
Normal	MCMCpack	MCMChregress	[272]
Logit	MCMCpack	MCMChlogit	[272]
Poisson	MCMCpack	MCMChpoisson	[272]
Bayesian model averaging			
Model	**Library**	**Command**	**Reference**
Normal (BIC)	BMA	bicreg	[321]
Normal (MC^3)	BMA	MC3.REG	[321]
Normal (instrumental variables)	ivbma	ivbma	[245]
Logit (BIC)	BMA	bic.glm	[321]
Gamma (BIC)	BMA	bic.glm	[321]
Poisson (BIC)	BMA	bic.glm	[321]
Diagnostics			
Diagnostic	**Library**	**Command**	**Reference**
Trace plot	coda	traceplot	[312]
Autocorrelation plot	coda	autocorr.plot	[312]
Geweke test	coda	geweke.diag	[312]
Raftery & Lewis test	coda	raftery.diag	[312]
Heidelberger & Welch test	coda	heidel.diag	[312]

TABLE 14.2
Datasets templates in folder *DataSim*.

Univariate models		
Model	**Data set file**	**Data set simulation**
Normal	11SimNormalmodel.csv	11SimNormal.R
Logit	12SimLogitmodel.csv	12SimLogit
Probit	13SimProbitmodel.csv	13SimProbit.R
Multinomial(Mixed) Probit	14SimMultProbmodel.csv	14SimMultinomialProbit.R
Multinomial(Mixed) Logit	15SimMultLogitmodel.csv	15SimMultinomialLogit.R
Ordered Probit	16SimOrderedProbitmodel.csv	16SimOrderedProbit.R
Negative Binomial(Poisson)	17SimNegBinmodel.csv	17SimNegBin.R
Tobit	18SimTobitmodel.csv	18SimTobit.R
Quantile	19SimQuantilemodel.csv	19SimQuantile.R
Bayesian bootstrap	41SimBootstrapmodel.csv	41SimBootstrapmodel.R
Multivariate models		
Model	**Data set file**	**Data set simulation**
Multivariate	21SimMultivariate.csv	21SimMultReg.R
Seemingly Unrelated Regression	22SimSUR.csv	22SimSUR.R
Instrumental Variable	23SimIV.csv	23SimIV.R
Bivariate Probit	24SimMultProbit.csv	24SimMultProbit.R
Hierarchical longitudinal models		
Model	**Data set file**	**Data set simulation**
Normal	31SimLogitudinalNormal.csv	31SimLogitudinalNormal.R
Logit	32SimLogitudinalLogit.csv	32SimLogitudinalLogit.R
Poisson	33SimLogitudinalPoisson.csv	33SimLogitudinalPoisson.R
Bayesian model averaging		
Model	**Data set file**	**Data set simulation**
Normal (BIC)	511SimNormalBMA.csv	511SimNormalBMA.R
Normal (MC^3)	512SimNormalBMA.csv	512SimNormalBMA.R
Normal (instrumental variables)	513SimNormalBMAivYXW.csv 513SimNormalBMAivZ.csv	513SimNormalBMAiv.R
Logit (BIC)	52SimLogitBMA.csv	52SimLogitBMA.R
Gamma (BIC)	53SimGammaBMA.csv	53SimGammaBMA.R
Poisson (BIC)	53SimPoissonBMA.csv	53SimPoissonBMA.R

TABLE 14.3
Real datasets in folder *DataApp*.

Univariate models		
Model	**Data set file**	**Dependent variable**
Normal	1ValueFootballPlayers.csv	log(Value)
Logit	2HealthMed.csv	Hosp
Probit	2HealthMed.csv	Hosp
Multinomial(Mixed) Probit	Fishing.csv	mode
Multinomial(Mixed) Logit	Fishing.csv	mode
Ordered Probit	2HealthMed.csv	MedVisPrevOr
Negative Binomial(Poisson)	2HealthMed.csv	MedVisPrev
Tobit	1ValueFootballPlayers.csv	log(ValueCens)
Quantile	1ValueFootballPlayers.csv	log(Value)
Bayesian bootstrap	1ValueFootballPlayers.csv	log(Value)
Multivariate models		
Model	**Data set file**	**Dependent variable**
Multivariate	4Institutions.csv	logpcGDP95 and PAER
Seemingly Unrelated Regression	5Institutions.csv	logpcGDP95 and PAER
Instrumental Variable	6Institutions.csv	logpcGDP95 and PAER
Bivariate Probit	7HealthMed.csv	$y = [\text{Hosp SHI}]'$
Hierarchical longitudinal models		
Model	**Data set file**	**Dependent variable**
Normal	8PublicCap.csv	log(gsp)
Logit	9VisitDoc.csv	DocVis
Poisson	9VisitDoc.csv	DocNum
Bayesian model averaging		
Model	**Data set file**	**Dependent variable**
Normal (BIC)	10ExportDiversificationHHI.csv	avghhi
Normal (MC^3)	10ExportDiversificationHHI.csv	avghhi
Normal (instrumental variables)	11ExportDiversificationHHI.csv 12ExportDiversificationHHIInstr.csv	avghhi and avglgdpcap
Logit (BIC)	13InternetMed.csv	internet
Gamma (BIC)	14ValueFootballPlayers.csv	log market value
Poisson (BIC)	15Fertile2.csv	ceb