Half Title

Solution Manual
Introduction to Bayesian Econometrics: A GUIded toolkit using R

Title Page

Solution Manual
Introduction to Bayesian Econometrics: A GUIded toolkit using R

by Andrés Ramírez-Hassan, PhD. Statistical Science.

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

Contents

Foreword

Preface

The main goal of this manual is to show the solution of the exercises of the book **Introduction to Bayesian Econometrics: A GUIded toolkit using R**. Trying to solve the exercises is important for those who want to improve their understanding about the Bayesian inferential framework more approachable.

To instructors

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 four ways to install our GUI:

1. Install *shiny* package, and then, type

   shiny::runGitHub("besmarter/BSTApp", launch.browser=T)

   in the **R** console or any **R** code editor and execute it. We recommend to type directly, no copy and paste.

2. Visit **https://andres-ramirez-hassan.shinyapps.io/BSTApp/**. Please note: the free Posit Cloud tier sometimes runs out of memory, which can cause the app to stop. Sorry for the inconvenience.

3. Visit **https://fly-besmarter.fly.dev/**. As with Posit Cloud, occasional memory limits on the free tier may affect performance.

4. Use a **Docker** image by typing in the **Command Prompt**

   (a) docker pull aramir21/besmartergui:latest

   (b) docker run –rm -p 3838:3838 aramir21/besmartergui

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

In addition, the `.tex` source files for the book's slides can be found in the GitHub repository:
https://github.com/aramir21/PresentationsBookBayesianEconometrics.

There is a free online version of the book at:
https://andresramirezhassan-introduction-bayesian-econometrics-

gui.share.connect.posit.cloud/, or **https://bookdown.org/aramir21/IntroductionBayesianEconometricsGuidedTour/**.

These allow users to copy and paste all the code and view every figure in full color.

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 [15, p. 182-191]. Additionally, students who wish to master the material in this book should have programming skills in **R** software.[1]

Instructors can send me an email to have access to this solution manual. They 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 [10], [5], [39], [15], [13], [24], and [22].

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* [37], 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 would like to thank Chris Parmeter for his valuable suggestions on how to present our user guide; Professors Raúl 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 their engaging discussions and fruitful collaborations in Bayesian econometrics and statistics. I am also deeply grateful 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 and has been a constant source of intellectual inspiration.

[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**.

I also wish to acknowledge my colleagues and staff at Universidad EAFIT for their continuous support.

Finally, I acknowledge the use of ChatGPT, which assisted me in improving the grammar, clarity, and flow of the text, as well as in tidying up some of the code presented in this book. Nevertheless, all concepts, mathematical developments, and underlying logic are entirely my own, based on my understanding and readings of the literature. Any remaining errors are solely my responsibility, for which I apologize in advance. I sincerely thank the reader and hope that this book proves useful.

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 wife, Estephania, for her unwavering love and support.

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		
$:=$	Is defined as	$>$	Greater than
argmax	Argument of the maximum	$<$	Less than
argmin	Argument of the minimum	$\approx$	Approximately equal to
tr	Trace operator	$\gtrsim$	Greater than or approximately equal to
vec	Vectorization operator		
lim	Limit	Δ	Difference operator
$\otimes$	Kronecker product	$\subseteq$	Subset
diag$\{\cdot\}$	Diagonal matrix	$\subset$	Proper subset
$dim\{\cdot\}$	Dimension of an object	$\xrightarrow{d}$	Convergence in distribution (law)
$\boldsymbol{I}_l$	l-dimensional identity matrix		
$\boldsymbol{i}_N$	N-dimensional vector of ones	$\xrightarrow{p}$	Convergence in probability
		$\xrightarrow{a.s.}$	Almost surely convergence

Part I

Foundations: Theory, simulation methods and programming

1

Basic formal concepts

Solutions of Exercises

1. *The court case: the blue or green cap*

   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?*

   Answer

   Set WB and WG equal to the events that the witness said the cab was blue and green, respectively. Set B and G equal to the events that the cabs are blue and green, respectively. We need to calculate $P(B|WB)$, then:

$$\begin{aligned} P(B|WB) &= \frac{P(B, WB)}{P(WB)} \\ &= \frac{P(WB|B) \times P(B)}{P(WB|B) \times P(B) + P(WB|\neg B) \times (1 - P(B))} \\ &= \frac{0.8 \times 0.15}{0.8 \times 0.15 + 0.25 \times 0.85} \\ &= 0.36 \end{aligned} \tag{1.1}$$

2. *The Monty Hall problem*

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

   Answer

   Let's name P_i the event *contestant picks door No. i*, H_i the event *host picks*

door No. i, and C_i the event *car is behind door No. i*. Let's assume that the contestant picked door number 1, and the host picked door number 3, then the contestant is interested in the probability of the event $P(C_i|H_3, P_1), i =$ 2 or 4. Then, $P(H_3|C_3, P_1) = 0$, $P(H_3|C_2, P_1) = P(H_3|C_4, P_1) = 1/2$ and $P(H_3|C_1, P_1) = 1/3$. Then,

$$\begin{aligned} P(C_i|H_3, P_1) &= \frac{P(C_i, H_3, P_1)}{P(H_3, P_1)} \\ &= \frac{P(H_3|C_i, P_1)P(C_i|P_1)P(P_1)}{P(H_3|P_1) \times P(P_1)} \\ &= \frac{P(H_3|C_i, P_1)P(C_i)}{P(H_3|P_1)} \\ &= \frac{1/2 \times 1/4}{1/3} \\ &= \frac{3}{8}, \end{aligned} \tag{1.2}$$

where the third equation uses the fact that C_i and P_i are independent events, and $P(H_3|P_1) = 1/3$ due to this depending just on P_1 (not on C_i).

Therefore, changing the initial decision increases the probability of getting the car from 1/4 to 3/8!

Let's check the case with n doors, and assume that the contestant picks the door No. 1, the car is behind the door No. n, and the host, who knows what is behind each door, opens any of the remaining $n-2$ doors, where there is a goat. The contestant is interested in the probability of the event:

$$\begin{aligned} P\left(C_n|(H_2 \cup \ldots \cup H_{n-1}) \cap P_1\right) &= \frac{P\left((H_2 \cup H_3 \cup \ldots \cup H_{n-1})|C_n \cap P_1\right) P(C_n|P_1)P(P_1)}{P\left((H_2 \cup H_3 \cup \ldots \cup H_{n-1})|P_1\right) P(P_1)} \\ &= \frac{\left[P\left(H_2|C_n \cap P_1\right) + \ldots + P\left(H_{n-1}|C_n \cap P_1\right)\right] P(C_n)}{P\left(H_2|P_1\right) + P\left(H_3|P_1\right) + \ldots + P\left(H_{n-1}|P_1\right)} \\ &= \frac{1 \times \left(\frac{1}{n}\right)}{\frac{1}{n-1} + \frac{1}{n-1} + \ldots + \frac{1}{n-1}} \\ &= \left(\frac{1}{n}\right)\left(\frac{n-1}{n-2}\right). \end{aligned} \tag{1.3}$$

In general, the probability of winning the car changing the pick is $\frac{1}{n}\frac{n-1}{n-2}$, while the probability of winning given no change is $\frac{1}{n}$. Given that $\frac{1}{n}\frac{n-1}{n-2} > \frac{1}{n}$ for all $n \geq 3$, where the difference between both probabilities is $\frac{1}{n(n-2)}$. We observe that as the number of doors increases, the difference between the two probabilities becomes zero.

Let's see a code for the general setting,

R code. The Monty Hall Problem

```
set.seed(0101) # Set simulation seed
S <- 100000 # Simulations
Game <- function(opt = 3){
# opt: number of options. opt > 2
opts <- 1:opt
car <- sample(opts, 1) # car location
guess1 <- sample(opts, 1) # Initial guess
if(opt == 3 && car != guess1) {
 host <- opts[-c(car, guess1)]
 } else {
 host <- sample(opts[-c(car, guess1)], 1)
}
win1 <- guess1 == car # Win given no change
if(opt == 3) {
 guess2 <- opts[-c(host, guess1)]
 } else {
 guess2 <- sample(opts[-c(host, guess1)], 1)
 }
win2 <- guess2 == car # Win given change
return(c(win1, win2))
}
#Win probabilities
Prob <- rowMeans(replicate(S, Game(opt = 4)))
#Winning probabilities no changing door
Prob[1]
0.25151
#Winning probabilities changing door
Prob[2]
0.37267
```

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

   Answer

   First, we get the posterior distribution,

$$\pi(\lambda|\mathbf{y}) = \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\lambda^{\alpha_0-1}e^{-\lambda\beta_0}\right)\left(\prod_{i=1}^{N}\frac{\lambda^{y_i}e^{-\lambda}}{y_i!}\right) \tag{1.4}$$

$$
\begin{aligned}
&= \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\lambda^{\alpha_0-1}e^{-\lambda\beta_0}\right)\left(\frac{\lambda^{\sum_{i=1}^N y_i}e^{-N\lambda}}{\prod_{i=1}^N y_i!}\right) \\
&= \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\frac{1}{\prod_{i=1}^N y_i!}\right)\lambda^{\sum_{i=1}^N y_i+\alpha_0-1}e^{-\lambda(\beta_0+N)} \\
&\propto \lambda^{\sum_{i=1}^N y_i+\alpha_0-1}e^{-\lambda(\beta_0+N)}. \qquad (1.5)
\end{aligned}
$$

The last expression is the kernel of a Gamma distribution with parameters $\alpha_n = \sum_{i=1}^N y_i + \alpha_0$ and $\beta_n = \beta_0 + N$.

Given that $\int_0^\infty \pi(\lambda|\mathbf{y})\,d\lambda = 1$, then the constant of proportionality in the last expression is $\Gamma(\alpha_n)/\beta_n^{\alpha_n}$. Therefore the posterior density function $\pi(\lambda|\mathbf{y})$ is $G(\alpha_n, \beta_n)$.

The posterior mean is

$$
\begin{aligned}
E[\lambda|\mathbf{y}] &= \frac{\alpha_n}{\beta_n} \qquad (1.6) \\
&= \frac{\sum_{i=1}^N y_i + \alpha_0}{\beta_0 + N} \\
&= \left(\frac{N}{\beta_0+N}\right)\bar{y} + \left(\frac{\beta_0}{\beta_0+N}\right)\frac{\alpha_0}{\beta_0} \\
&= w\bar{y} + (1-w)\,E[\lambda],
\end{aligned}
$$

where $w = \frac{N}{\beta_0+N}$, $\bar{y}$ is the sample mean, and $E[\lambda] = \frac{\alpha_0}{\beta_0}$.

The posterior predictive distribution is given by

$$
\begin{aligned}
\pi(Y_0|\mathbf{y}) &= \int_0^\infty \frac{\lambda^{y_0}e^{-\lambda}}{y_0!}\pi(\lambda|\mathbf{y})\,d\lambda \qquad (1.7) \\
&= \int_0^\infty \left(\frac{\lambda^{y_0}e^{-\lambda}}{y_0!}\right)\left(\frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)}\lambda^{\alpha_n-1}e^{-\lambda\beta_n}\right)d\lambda \\
&= \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)\,y_0!}\int_0^\infty \lambda^{y_0+\alpha_n-1}e^{-\lambda(1+\beta_n)}d\lambda \\
&= \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)\,y_0!}\frac{\Gamma(y_0+\alpha_n)}{(1+\beta_n)^{y_0+\alpha_n}} \\
&= \frac{\Gamma(y_0+\alpha_n)}{\Gamma(\alpha_n)\,y_0!}\left(\frac{1}{1+\beta_n}\right)^{y_0}\left(\frac{\beta_n}{1+\beta_n}\right)^{\alpha_n}
\end{aligned}
$$

$$= \frac{(y_0+\alpha_n-1)!}{(\alpha_n-1)!y_0!}\left(\frac{1}{1+\beta_n}\right)^{y_0}\left(\frac{\beta_n}{1+\beta_n}\right)^{\alpha_n}$$
$$= \binom{y_0+\alpha_n-1}{y_0}\left(\frac{1}{1+\beta_n}\right)^{y_0}\left(\frac{\beta_n}{1+\beta_n}\right)^{\alpha_n}.$$

Therefore $Y_0|y \sim NB(\alpha_n, p_n)$ where $p_n = \frac{1}{1+\beta_n}$.
To use empirical Bayes, we have the following setting

$$\left[\hat{\alpha_0}\hat{\beta_0}\right] = \arg\max_{\alpha_0,\beta_0} \ln(p(\mathbf{y})),$$

where

$$p(y) = \int_0^\infty \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\lambda^{\alpha_0-1}e^{-\lambda\beta_0}\right)\left(\prod_{i=1}^N \frac{\lambda^{y_0}e^{-\lambda}}{y_0!}\right)d\lambda \tag{1.8}$$
$$= \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)\prod_{i=1}^N y_i!}\right)\int_0^\infty \lambda^{\sum_{i=1}^N y_i+\alpha_0-1}e^{-\lambda(\beta_0+N)}d\lambda$$
$$= \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)\prod_{i=1}^N y_i!}\right)\left(\frac{\Gamma\left(\sum_{i=1}^N y_i+\alpha_0\right)}{(\beta_0+N)^{\sum_{i=1}^N y_i+\alpha_0}}\right)$$
$$= \frac{\Gamma\left(\sum_{i=1}^N y_i+\alpha_0\right)}{\Gamma(\alpha_0)\prod_{i=1}^N y_i!}\left(\frac{1}{\beta_0+N}\right)^{\sum_{i=1}^N y_i}\left(\frac{\beta_0}{\beta_0+N}\right)^{\alpha_0}.$$

R code. Health insurance, predictive distribution using vague hyperparameters

```
set.seed(010101)
y <- c(0, 3, 2, 1, 0) # Data
N <- length(y)

# Predictive distribution
ProbBo <- function(y, a0, b0){
  N <- length(y)
  #sample size
  aN <- a0 + sum(y)
  # Posterior shape parameter
  bN <- b0 + N
  # Posterior scale parameter
  p <- 1 / (bN + 1)
  # Probability negative binomial density
  Pr <- 1 - pnbinom(0, size = aN, prob = (1 - p))
  # Probability of visiting the Doctor
  # Observe that in R there is a slightly
  # different parametrization.
  return(Pr)
}

# Using a vague prior:
a0 <- 0.001 # Prior shape parameter
b0 <- 0.001 # Prior scale parameter
PriMeanV <- a0 / b0 # Prior mean
PriVarV <- a0 / b0^2 # Prior variance
Pp <- ProbBo(y, a0 = 0.001, b0 = 0.001)
# This setting is vague prior information.
Pp
0.67
```

R code. Health insurance, predictive distribution using empirical Bayes

```
# Using Emprirical 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) + a0*log(b0/(b0
    +N)) - lgamma(a0)}
    # log marginal likelihood
  return(-lnp)
}
theta0 <- c(0.01, 0.01)
# 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$convergence
# Checking convergence
EmpBay$value # Maximum
a0EB <- EmpBay$par[1]
# Prior shape using empirical Bayes
a0EB
128.383
b0EB <- EmpBay$par[2]
# Prior scale using empirical Bayes
b0EB
106.801
PriMeanEB <- a0EB / b0EB
# Prior mean
PriVarEB <- a0EB / b0EB^2
# Prior variance
PpEB <- ProbBo(y, a0 = a0EB, b0 = b0EB)
# This setting is using empirical Bayes.
PpEB
0.69
```

4. Suppose that you are analyzing to buy a car insurance next year. To make

a better decision you want to know *what is the probability that you have a car claim next year?* You have the records of your car claims in 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, $Y_i \sim Ber(p)$, and your probabilistic prior beliefs about p are well described by a beta distribution with parameters α_0 and β_0, $p \sim B(\alpha_0, \beta_0)$, then, you are interested in calculating the probability of a claim the next year $P(Y_0 = 1|\mathbf{y})$.

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

Answer

The posterior distribution is given by

$$\begin{aligned}\pi(p|\mathbf{y}) &= \left[\frac{\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)}p^{\alpha_0-1}(1-p)^{\beta_0-1}\right]\left[\prod_{i=1}^{N}p^{y_i}(1-p)^{1-y_i}\right] \\ &= \frac{\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)}p^{\sum_{i=1}^{N}y_i+\alpha_0-1}(1-p)^{\beta_0+N-\sum_{i=1}^{N}y_i-1} \\ &\propto p^{\sum_{i=1}^{N}y_i+\alpha_0-1}(1-p)^{\beta_0+N-\sum_{i=1}^{N}y_i-1}.\end{aligned} \tag{1.9}$$

The last expression is the kernel of a Beta distribution with parameters $\alpha_n = \sum_{i=1}^{N} y_i + \alpha_0$ and $\beta_n = \beta_0 + N - \sum_{i=1}^{N} y_i$. Thus, the posterior mean is

$$\begin{aligned}E[p|\mathbf{y}] &= \frac{\alpha_n}{\alpha_n+\beta_n} \\ &= \frac{\sum_{i=1}^{N}y_i+\alpha_0}{\alpha_0+\beta_0+N} \\ &= \frac{N\bar{y}}{\alpha_0+\beta_0+N}+\frac{\alpha_0}{\alpha_0+\beta_0+N} \\ &= \frac{N}{\alpha_0+\beta_0+N}(\bar{y})+\frac{\alpha_0+\beta_0}{\alpha_0+\beta_0+N}\left(\frac{\alpha_0}{\alpha_0+\beta_0}\right) \\ &= w(\bar{y})+(1-w)E[p],\end{aligned} \tag{1.10}$$

where $w = \frac{N}{\alpha_0+\beta_0+N}$, $\bar{y}$ is the sample mean, and $E[p] = \frac{\alpha_0}{\alpha_0+\beta_0}$ is the prior mean.

The posterior predictive distribution of claim the next year is given by

$$\begin{aligned}\pi(Y_0 = 1|\mathbf{y}) &= \int_0^1 P(Y_0 = 1|\mathbf{y}, p)\pi\left(p|\mathbf{y}\right) dp \\ &= \int_0^1 p \times \pi\left(p|\mathbf{y}\right) dp \\ &= \mathbb{E}\left[p|\mathbf{y}\right] \\ &= \frac{\alpha_n}{\alpha_n + \beta_n}. \end{aligned} \tag{1.11}$$

To use empirical Bayes, we have the following setting

$$\left[\hat{\alpha_0}\ \hat{\beta_0}\right] = \arg\max_{\alpha_0, \beta_0} \ln(p(\mathbf{y})),$$

where

$$\begin{aligned} p(\mathbf{y}) &= \int_0^1 \left[\frac{\Gamma\left(\alpha_0 + \beta_0\right)}{\Gamma\left(\alpha_0\right)\Gamma\left(\beta_0\right)} p^{\alpha_0 - 1}\left(1-p\right)^{\beta_0 - 1}\right]\left[\prod_{i=1}^{N}\left(1-p\right)^{1-y_i}\right] dp \\ &= \frac{\Gamma\left(\alpha_0 + \beta_0\right)}{\Gamma\left(\alpha_0\right)\Gamma\left(\beta_0\right)} \int_0^1 p^{\sum_{i=1}^N y_i + \alpha_0 - 1}\left(1-p\right)^{\beta_0 + N - \sum_{i=1}^N y_i - 1} dp \\ &= \frac{\Gamma\left(\alpha_0 + \beta_0\right)}{\Gamma\left(\alpha_0\right)\Gamma\left(\beta_0\right)} \frac{\Gamma\left(\sum_{i=1}^N y_i + \alpha_0\right)\Gamma\left(\beta_0 + N - \sum_{i=1}^N y_i\right)}{\Gamma\left(\alpha_0 + \beta_0 + N\right)}. \end{aligned} \tag{1.12}$$

R code. Car claim, predictive distribution using vague hyperparameters

```
set.seed(010101)
y <- c(0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0)
# Data
N <- length(y)
#require(TailRank)
# Predictive distribution
ProbBo <- function(y, a0, b0){
  N <- length(y)
  #sample size
  aN <- a0 + sum(y)
  # Posterior shape parameter
  bN <- b0 + N - sum(y)
  # Posterior scale parameter
  pr <- aN / (aN + bN)
  # Probability of a claim the next year
  return(pr)
}
# Using a vague prior:
a0 <- 1 # Prior shape parameter
b0 <- 1 # Prior scale parameter
PriMeanV <- a0 / (a0 + b0)
# Prior mean
PriVarV <- (a0*b0) / (((a0+b0)^2)*(a0+b0+1))
# Prior variance
Pp <- ProbBo(y, a0 = 1, b0 = 1)
# This setting is defining vague prior information.
# The probability of a claim
Pp
0.47
```

R code. Car claim, predictive distribution using empirical Bayes

```
# Using Emprirical 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(a0+b0) + lgamma(aN) + lgamma(b0+N-sum(
    y)) -lgamma(a0) - lgamma(b0) - lgamma(a0+b0+N)}
  # log marginal likelihood
  return(-lnp)
}
theta0 <- c(0.1, 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$convergence
# Checking convergence
EmpBay$value # Maximum
a0EB <- EmpBay$par[1]
# Prior shape using empirical Bayes
b0EB <- EmpBay$par[2]
# Prior scale using empirical Bayes
PriMeanEB <- a0EB /(a0EB + b0EB)
# Prior mean
PriVarEB <- (a0EB*b0EB)/(((a0EB+b0EB)^2)*(a0EB+b0EB+1))
# Prior variance
PpEB <- ProbBo(y, a0 = a0EB, b0 = b0EB)
# This setting is using empirical Bayes.
PpEB
0.47
```

R code. Car claim, density plots

```
# Density figures
lambda <- seq(0.001, 1, 0.001)
# Values of lambda
VaguePrior <- dbeta(lambda, shape1 = a0, shape2 = b0)
EBPrior <- dbeta(lambda, shape1 = a0EB, shape2 = b0EB)
PosteriorV <- dbeta(lambda, shape1 = a0 + sum(y), shape2 =
    b0 + N - sum(y))
PosteriorEB <- dbeta(lambda, shape1 = a0EB + sum(y), shape2
    = b0EB + N - sum(y))
# Likelihood function
Likelihood <- function(theta, y){
  LogL <- dbinom(y, 1, theta, log = TRUE)
  #  LogL <- dbern(y, theta)
  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("$p$")) + ylab("Density") + ggtitle("Prior: Vague
     Beta")
fig2 <- ggplot(data = data, aes(lambda, EBPrior)) +
  geom_line() +
  xlab(TeX("$p$")) + ylab("Density") +
  ggtitle("Prior: Empirical Bayes Beta")
fig3 <- ggplot(data = data, aes(lambda, PosteriorV)) +
  geom_line() +
  xlab(TeX("$p$")) + ylab("Density") +
  ggtitle("Posterior: Vague Beta")
fig4 <- ggplot(data = data, aes(lambda, PosteriorEB)) +
  geom_line() +
  xlab(TeX("$p$")) + ylab("Density") +
  ggtitle("Posterior: Empirical Bayes Beta")
FIG <- ggarrange(fig1, fig2, fig3, fig4,
ncol = 2, nrow = 2)
annotate_figure(FIG,
top = text_grob("Vague versus Empirical Bayes: Beta-
    Bernoulli model", color = "black", face = "bold", size =
     14))
```

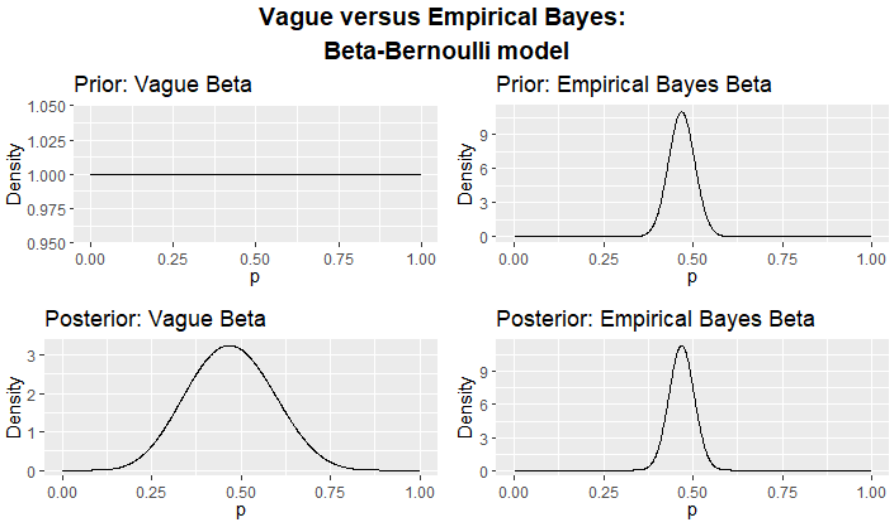

FIGURE 1.1
Vague versus Empirical Bayes: Bernoulli-Beta model.

R code. Car claim, prior, likelihood and posterior density plots

```
# Prior, likelihood and posterior:
#Empirical Bayes Binonial-Beta model
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"))
```

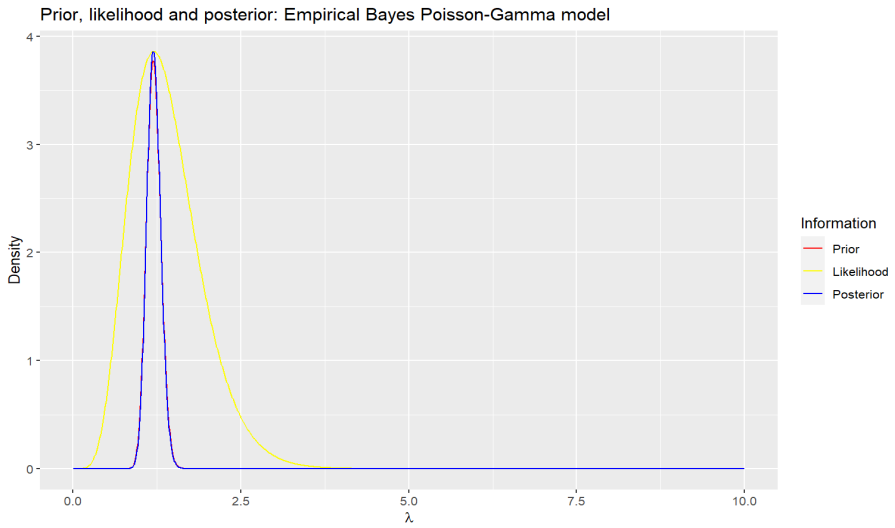

FIGURE 1.2
Prior, likelihood and posterior: Bernoulli-Beta model.

R code. Car claim, predictive probabilities plots

```
# Predictive distributions
require(TailRank)
PredDen <- function(y, y0, a0, b0){
  N <- length(y)
  aN <- a0 + sum(y) # Posterior shape parameter
  bN <- b0 + N - sum(y) # Posterior scale parameter
  Pr <- aN/(aN+bN)
  Probs <- dbinom(y0, 1, prob = Pr)
  return(Probs)
}
y0 <- 0:1
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(0,1,by=1))
```

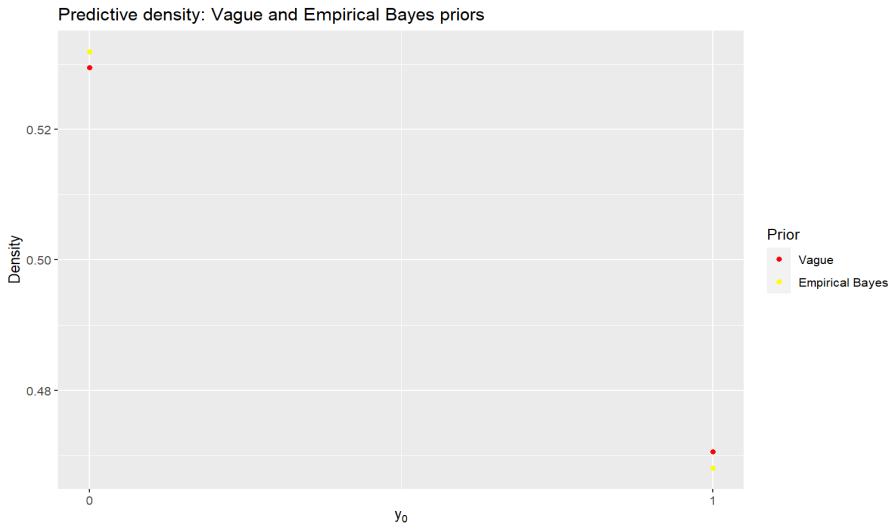

FIGURE 1.3
Predictive probabilities: Bernoulli-Beta model.

R code. Car claim, Bayesian model average

```
# Posterior odds: Vague vs Empirical Bayes
PO12 <- exp(-LogMgLik(c(a0EB, b0EB), y = y))/exp(-LogMgLik(c
    (a0, b0), y = y))
PostProMEM <- PO12/(1 + PO12)
# Posterior model probability Empirical Bayes
PostProMEM
0.757
PostProbMV <- 1 - PostProMEM
# Posterior model probability vague prior
PostProbMV
0.242
# Bayesian model average (BMA)
PostMeanEB <- (a0EB + sum(y)) / (a0EB + b0EB + N)
# Posterior mean Empirical Bayes
PostMeanV <- (a0 + sum(y)) / (a0 + b0 + N)
# Posterior mean vague priors
BMAmean <- PostProMEM * PostMeanEB + PostProbMV * PostMeanV
# BMA posterior mean
PostVarEB <- (a0EB + sum(y))*(b0EB + N - sum(y)) / ((a0EB +
    b0EB + N)^2)*(a0EB + b0EB + N -1)
# Posterior variance Empirical Bayes
PostVarV <- (a0 + sum(y))*(b0 + N - sum(y)) / ((a0 + b0 + N)
    ^2)*(a0 + b0 + N -1)
# Posterior variance vague prior
BMAVar <- PostProMEM * PostVarEB + PostProbMV * PostVarV +
    PostProMEM * (PostMeanEB - BMAmean)^2 + PostProbMV * (
    PostMeanV - BMAmean)^2
# BMA posterior variance
# 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(0,1,by=1))
```

R code. Car claim, Bayesian updating plots

```
# Bayesian updating
BayUp <- function(y, lambda, a0, b0){
 N <- length(y)
 aN <- a0 + sum(y)
 # Posterior shape parameter
 bN <- b0 + N - sum(y)
 # Posterior scale parameter
 p <- dbeta(lambda, shape1 = aN, shape2 = 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 = 1, b0 = 1)}
  else{
  PostUpi <- BayUp(y[i], lambda,
  a0 = PostUpi$a0New, b0 = PostUpi$b0New)
 }
 PostUp <- cbind(PostUp, PostUpi$Post)
}
DataUp <- data.frame(cbind(rep(lambda, 15), c(PostUp), rep
    (1:15, each = 1000)))  #Data frame
colnames(DataUp) <- c("Lambda", "Density", "Factor")
DataUp$Factor <- factor(DataUp$Factor, levels=c("1","2","3",
    "4","5","6","7","8","9","10","11","12","13","14","15"),
    labels=c("Iter_1","Iter_2","Iter_3","Iter_4","Iter_5","
    Iter_6","Iter_7","Iter_8","Iter_9","Iter_10","Iter_11","
    Iter_12","Iter_13","Iter_14","Iter_15"))
ggplot(data = DataUp, aes_string(x = "Lambda",
 y = "Density", group = "Factor")) +
 geom_line(aes(color = Factor)) +
 xlab(TeX("$p$")) + ylab("Density") +
 ggtitle("Bayesian updating:
 Beta-Binomial model with vague prior") +
 guides(color=guide_legend(title="Update"))
```

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

Answer

$\int_\Theta |\theta - a|\pi(\theta|\mathbf{y})d\theta = \int_{-\infty}^{a}(a-\theta)\pi(\theta|\mathbf{y})d\theta + \int_a^{\infty}(\theta - a)\pi(\theta|\mathbf{y})d\theta$. Differentiating with respect to a, and equating to zero,

$$\int_{-\infty}^{a}\pi(\theta|\mathbf{y})d\theta = \int_a^{\infty}\pi(\theta|\mathbf{y})d\theta, \tag{1.13}$$

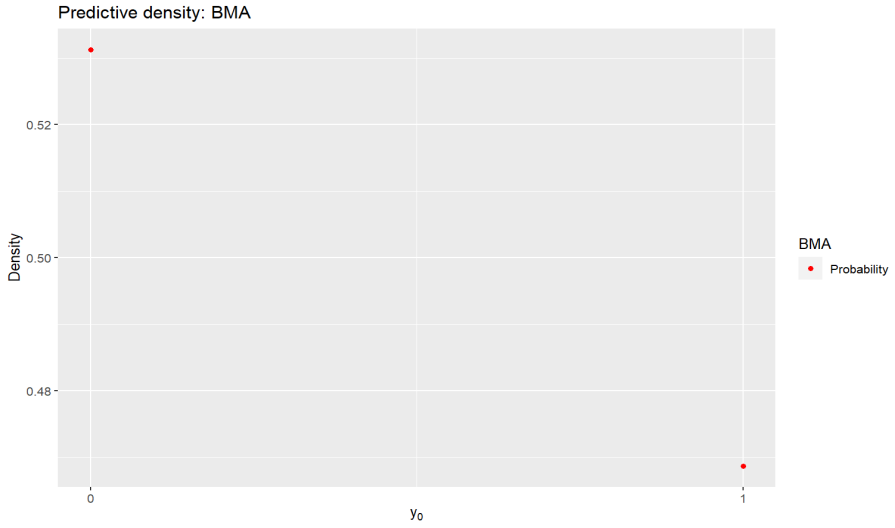

FIGURE 1.4
Predictive probabilities: Bernoulli-Beta Bayesian model average.

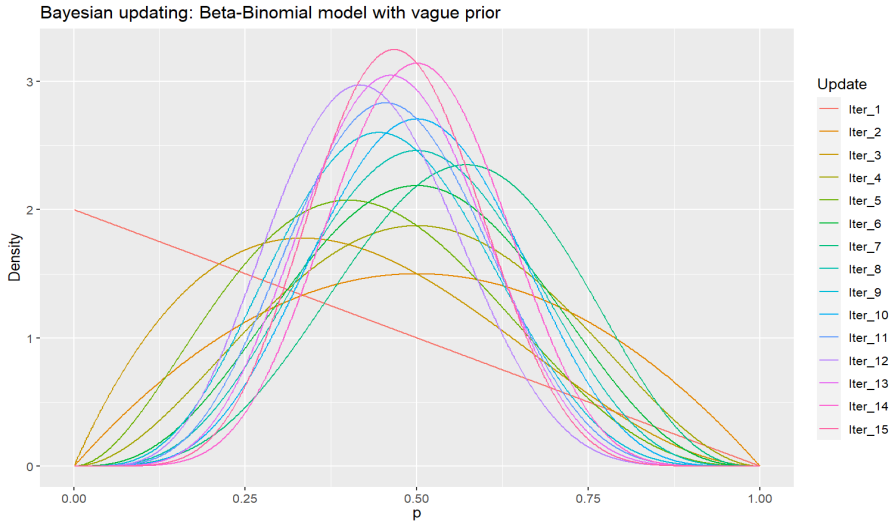

FIGURE 1.5
Predictive probabilities: Bernoulli-Beta Bayesian model updating.

then,

$$2\int_{-\infty}^{a}\pi(\theta|\mathbf{y})d\theta=\int_{-\infty}^{\infty}\pi(\theta|\mathbf{y})d\theta=1, \tag{1.14}$$

that is, $a^*(\mathbf{y})$ is the median.

2

Conceptual differences of the Bayesian and Frequentist approaches

Solutions of Exercises

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

   The **Jeffreys-Lindley's paradox** [17, 26] is an apparent disagreement between the Bayesian and Frequentist frameworks to a hypothesis testing situation.

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

   - Show that the posterior model probability for the model under the null is approximately 0.95. Assume $\pi(H_0) = \pi(H_1) = 0.5$, and $\pi(\theta)$ equal to $U(0, 1)$ under H_1.
   - Show that the p-value for this hypothesis test is equal to 0.023 using the normal approximation, $Y \sim N(N \times \theta, N \times \theta \times (1 - \theta))$.

   Answer

   - The marginal likelihood under the null hypothesis is $p(y|H_0) = \binom{N}{y}\theta^y(1-\theta)^{N-y} \approx 1.95 \times 10^{-4}$ given $\theta = 0.5$ under H_0, $N = 49,581 + 48,870$ and $y = 49,581$. On the other hand, the marginal likelihood under the alternative hypothesis is

$$\begin{aligned} p(y|H_1) &= \int_0^1 \binom{N}{y} \theta^y (1-\theta)^{N-y} d\theta \\ &= \binom{N}{y} B(y+1, N-k+1) \\ &= \frac{\Gamma(N+1)}{\Gamma(y+1)\Gamma(n-y+1)} \frac{\Gamma(y+1)\Gamma(N-y+1)}{\Gamma(N+2)} \\ &= \frac{N!}{(N+1)!} \\ &= \frac{1}{N+1} \\ &\approx 1.016 \times 10^{-5}. \end{aligned}$$

Then, $PO_{01} = \frac{1.95\times10^{-4}}{1.016\times10^{-5}} = 19.19$, this implies that the posterior model probability under the null hypothesis is $\pi(H_0|y) = \frac{19.19}{1+19.19} = 0.95$.

- Under the null hypothesis,

$$\begin{aligned} p &= 2\int_{49{,}581}^{\infty} (2\pi\sigma^2)^{-1/2} \exp\left\{-\frac{1}{2\sigma^2}(y-\mu)^2\right\} dy \\ &= 0.0235, \end{aligned}$$

where $\mu = N \times \theta = 49,225.5$, and $\sigma^2 = N \times \theta \times (1-\theta) = 24,612.75$ under the null hypothesis ($\theta = 0.5$).

Observe that the posterior model probability supports the null hypothesis, whereas the p-value implies rejection of the null hypothesis using a 5% significance level.

Observe that actually this is not a paradox, as we are answering two different questions. The Bayes factor is comparing two models ($\theta = 0.5$ versus $\theta \sim U(0,1)$), whereas the p-value is checking the compatibility between $\theta = 0.5$ and the sample information. Despite that $\theta = 0.5$ is not compatible with sample information, it is better than the models assuming $\theta \sim U(0,1)$ as most of these values of θ are far away from the sample mean. Thus, the model under the null is a bad description of the data, but it is better than the model under the alternative hypothesis.[1]

[1] Observe that there are at least another two issues in this example. First, the prior under the alternative is non-informative, this implies problems for Bayes factors, and second, the prior under the alternative is positive at $\theta = 0.5$, which is the null ([20] propose non-local prior densities in Bayesian hypothesis tests to tackle these issues).

2. We want to test H_0. $\mu = \mu_0$ vs 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 $\pi(\mu, \sigma) \propto 1/\sigma$ 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,

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

   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 and the classical test statistic for the null hypothesis.

   Answer

$$
\begin{aligned}
p(\mathbf{y}|\mathcal{M}_1) &= \int_{-\infty}^{\infty}\int_0^{\infty}(2\pi)^{-N/2}\sigma^{-N}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^N(y_i-\mu)^2\right\}\frac{1}{\sigma}d\sigma d\mu \\
&= (2\pi)^{-N/2}\int_{-\infty}^{\infty}\int_0^{\infty}\sigma^{-(N+1)}\exp\left\{-\frac{N}{2\sigma^2}\frac{\sum_{i=1}^N(y_i-\mu)^2}{N}\right\}d\sigma d\mu \\
&= (2\pi)^{-N/2}\frac{\Gamma(N/2)}{2}2^{N/2}\int_{-\infty}^{\infty}\left[\sum_{i=1}^N(y_i-\mu)^2\right]^{-N/2}d\mu \\
&= (2\pi)^{-N/2}\frac{\Gamma(N/2)}{2}2^{N/2}\int_{-\infty}^{\infty}\left[\sum_{i=1}^N[(y_i-\bar{y})-(\mu-\bar{y})]^2\right]^{-N/2}d\mu \\
&= (2\pi)^{-N/2}\frac{\Gamma(N/2)}{2}2^{N/2}\int_{-\infty}^{\infty}\left[\alpha_n\hat{\sigma}^2+N(\mu-\bar{y})^2\right]^{-N/2}d\mu \\
&= (2\pi)^{-N/2}\frac{\Gamma(N/2)}{2}2^{N/2}\left(\frac{\alpha_n\hat{\sigma}^2}{\alpha_n\hat{\sigma}^2}\right)^{-N/2}\int_{-\infty}^{\infty}\left[\alpha_n\hat{\sigma}^2+N(\mu-\bar{y})^2\right]^{-N/2}d\mu \\
&= (2\pi)^{-N/2}\frac{\Gamma(N/2)}{2}2^{N/2}\left(\alpha_n\hat{\sigma}^2\right)^{-N/2}\int_{-\infty}^{\infty}\left[1+\frac{N(\mu-\bar{y})^2}{\alpha_n\hat{\sigma}^2}\right]^{-N/2}d\mu \\
&= \frac{\pi^{-N/2}}{2}\Gamma(N/2)\left(\frac{1}{\alpha_n\hat{\sigma}^2}\right)^{N/2}\left(\frac{N}{\alpha_n\hat{\sigma}^2}\right)^{-1/2}\frac{\Gamma(1/2)\Gamma(\alpha_n/2)}{\Gamma((\alpha_N+1)/2)}.
\end{aligned}
$$

   The third line takes into account that the integral in the second line is the kernel of an inverted-gamma distribution, and the last line takes into account that the integral in the previous line is the kernel of a student's t distribution [49].

$$
\begin{aligned}
p(\mathbf{y}|\mathcal{M}_0) &= \int_0^\infty (2\pi)^{-N/2}\sigma^{-N}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-\mu_0)^2\right\}\frac{1}{\sigma}d\sigma \\
&= (2\pi)^{-N/2}\int_0^\infty \sigma^{-(N+1)}\exp\left\{-\frac{N}{2\sigma^2}\frac{\sum_{i=1}^{N}(y_i-\mu_0)^2}{N}\right\}d\sigma \\
&= (2\pi)^{-N/2}\left[\frac{2}{\Gamma(N/2)}\left(\frac{N}{2}\frac{\sum_{i=1}^{N}(y_i-\mu_0)^2}{N}\right)^{N/2}\right]^{-1}.
\end{aligned}
$$

The third line takes into account that the integral in the second line is the kernel of an inverted-gamma distribution [49].

Given these results is easy to get PO_{01}.

In addition,

$$
\begin{aligned}
PO_{01} &= \frac{\Gamma((\alpha_n+1)/2)}{\Gamma(1/2)\Gamma(\alpha_n/2)}(\alpha_n\hat{\sigma}^2/N)^{-1/2}\left[1+\frac{(\mu_0-\bar{y})^2}{\alpha_n\hat{\sigma}^2/N}\right]^{-\left(\frac{\alpha_n+1}{2}\right)} \\
&= \frac{\Gamma((\alpha_n+1)/2)}{\Gamma(1/2)\Gamma(\alpha_n/2)}(\alpha_n\hat{\sigma}^2/N)^{-1/2}\left[1+\frac{1}{\alpha_n}\left(\frac{\mu_0-\bar{y}}{\hat{\sigma}/\sqrt{N}}\right)^2\right]^{-\left(\frac{\alpha_n+1}{2}\right)} \\
&= \frac{\Gamma((\alpha_n+1)/2)}{\Gamma(1/2)\Gamma(\alpha_n/2)}(\alpha_n\hat{\sigma}^2/N)^{-1/2}\left[1+\frac{1}{\alpha_n}t^2\right]^{-\left(\frac{\alpha_n+1}{2}\right)},
\end{aligned}
$$

where $t=\frac{\bar{y}-\mu_0}{\hat{\sigma}/\sqrt{N}}$ is the classical statistical test. Then, as t increases then the PO_{01} decreases, both indicating support against the null hypothesis H_0. $\mu=\mu_0$. However, there are other terms affecting the posterior odds, then, there is no necessary agreement between the classical test statistic and the posterior odds.

3. **Math test continues**

   Using the setting of the **Example: Math test** in subsection 2.6.1 in the book, test H_0. $\mu=\mu_0$ vs 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.

   Answer

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
y.H0 <- c(100, 100.5, 101, 101.5, 102)
test <- (y.H0 - y_bar)/serror
pval <- 2*pt(test, alpha)
pval
0.0000459 0.0015431 0.0299338 0.2690040 1
# p-values
PO01 <- (gamma(N/2)*((N-1)*serror^2)^(-0.5)*(1+test^2/alpha)
    ^(-N/2))/(gamma(1/2)*gamma((N-1)/2))
PO01/(1+PO01)
0.0001705 0.0050345 0.0725330 0.3210223 0.4702050
# Posterior model probability of the null hypothesis.
```

3

Cornerstone models: Conjugate families

Solutions of Exercises

1. Write in the canonical form the distribution of the Bernoulli example, and find the mean and variance of the sufficient statistic.

   Answer

   Given $p(\mathbf{y}|\theta) = (1-\theta)^N \exp\left\{\sum_{i=1}^N y_i \log\left(\frac{\theta}{1-\theta}\right)\right\}$ where $\eta = \log\frac{\theta}{1+\theta}$ which implies $\theta = \frac{\exp(\eta)}{1-\exp(\eta)}$, then $p(\mathbf{y}|\theta) = \exp\left\{\sum_{i=1}^N y_i\eta - N\log(1+\exp(\eta))\right\}$. Thus $B(\eta) = N\log(1+\exp(\eta))$, $\nabla(B(\eta)) = N\frac{\exp(\eta)}{1+\exp(\eta)} = N\theta$ and $\nabla^2(B(\eta)) = N\left\{\frac{\exp(\eta)(1+\exp(\eta))}{(1+\exp(\eta))^2} - \frac{\exp(\eta)\exp(\eta)}{(1+\exp(\eta))^2}\right\} = N\theta(1-\theta)$.

2. Given a random sample $\mathbf{y} = [y_1, y_2, \ldots, y_N]^\top$ from N *binomial experiments* each having known size n_i and same unknown probability θ. Show that $p(\mathbf{y}|\theta)$ is in the exponential family, and find the posterior distribution, the marginal likelihood and the predictive distribution of the binomial-beta model assuming the number of trials is known.

   Answer

   The density function is

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

   Observe that $\sum_{i=1}^N n_i$ is the total sample size of Bernoulli experiments.

Using Theorem 1 in Chapter 4, the prior distribution is

$$\begin{aligned}\pi(\theta) &\propto (1-\theta)^{B_0} \exp\left\{a_0 \log\left(\frac{\theta}{1-\theta}\right)\right\} \\ &= \theta^{a_0}(1-\theta)^{B_0-a_0} \\ &= \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1},\end{aligned}$$

where $\alpha_0 = a_0 + 1$ and $\beta_0 = B_0 - a_0 + 1$. This is the kernel of a beta distribution. Thus, the posterior distribution is

$$\begin{aligned}\pi(\theta|\mathbf{y}) &\propto \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1} \times \theta^{\sum_{i=1}^N y_i}(1-\theta)^{\sum_{i=1}^N n_i - \sum_{i=1}^N y_i} \\ &= \theta^{\alpha_0+\sum_{i=1}^N y_i-1}(1-\theta)^{\beta_0+\sum_{i=1}^N n_i-\sum_{i=1}^N y_i-1} \\ &= \theta^{\alpha_n-1}(1-\theta)^{\beta_n-1},\end{aligned}$$

where $\alpha_n = \alpha_0 + \sum_{i=1}^N y_i$ and $\beta_n = \beta_0 + \sum_{i=1}^N n_i - \sum_{i=1}^N y_i$.
The marginal likelihood is

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

The third line due to having the kernel of a Beta distribution.
Finally, the predictive distribution is

$$\begin{aligned}p(Y_0|\mathbf{y}) &= \int_0^1 \binom{n_{y_0}}{y_0} \theta^{y_0}(1-\theta)^{n_{y_0}-y_0} \frac{\theta^{\alpha_n-1}(1-\theta)^{\beta_n-1}}{B(\alpha_n,\beta_n)} d\theta \\ &= \frac{\binom{n_{y_0}}{y_0}}{B(\alpha_n,\beta_n)} \int_0^1 \theta^{\alpha_n+y_0-1}(1-\theta)^{\beta_n+n_{y_0}-y_0-1} d\theta \\ &= \binom{n_{y_0}}{y_0} \frac{B(\alpha_n+y_0, \beta_n+n_{y_0}-y_0)}{B(\alpha_n,\beta_n)},\end{aligned}$$

where n_{y_0} is the known size associated with y_0, and the last line due to having the kernel of a beta distribution. The predictive is a *beta-binomial distribution.*

3. Given a random sample $\mathbf{y} = [y_1, y_2, \ldots, y_N]^\top$ from a *exponential distribution*. Show that $p(\mathbf{y}|\lambda)$ is in the exponential family, and find the posterior distribution, marginal likelihood and predictive distribution of the exponential-gamma model.

   Answer

   We see that the exponential distribution belongs to the exponential family as $p(\mathbf{y}|\lambda) = \prod_{i=1}^N \lambda \exp(-\lambda y_i) = \lambda^N \exp(-\lambda \sum_{i=1}^N y_i)$.

   Using the gamma distribution in the rate parametrization, we see that $\pi(\lambda|\mathbf{y}) \propto \lambda^{\alpha_0 - 1} \exp(-\lambda\beta_0) \times \lambda^N \exp(-\lambda \sum_{i=1}^N y_i) = \lambda^{\alpha_0 + N - 1} \exp(-\lambda(\beta_0 + \sum_{i=1}^N y_i))$. This is the kernel of a gamma distribution, that is, $\lambda|\mathbf{y} \sim G(\alpha_n, \beta_n)$ where $\alpha_n = \alpha_0 + N$ and $\beta_n = \beta_0 + \sum_{i=1}^N y_i$.

   The marginal likelihood is

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

   Finally, the predictive distribution is

$$\begin{aligned} p(Y_0|\mathbf{y}) &= \int_0^\infty \lambda \exp\left\{-\lambda y_0\right\} \lambda^{\alpha_n - 1} \exp\left\{-\beta_n \lambda\right\} \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)} d\lambda \\ &= \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)} \int_0^\infty \lambda^{\alpha_n + 1 - 1} \exp\left\{-\lambda(\beta_n + y_0)\right\} d\lambda \\ &= \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)} \times \frac{\Gamma(\alpha_n + 1)}{(\beta_n + y_0)^{\alpha_n + 1}} \\ &= \frac{\alpha_n \beta_n^{\alpha_n}}{(\beta_n + y_0)^{\alpha_n + 1}}. \end{aligned}$$

   This is a *Lomax distribution*.

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

   Answer

$$
\begin{aligned}
p(\mathbf{y}|\boldsymbol{\mu},\boldsymbol{\Sigma}) &= (2\pi)^{-N/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left\{-\frac{1}{2}\left(\mathbf{y}-\boldsymbol{\mu}\right)^{\top}\boldsymbol{\Sigma}^{-1}\left(\mathbf{y}-\boldsymbol{\mu}\right)\right\} \\
&= (2\pi)^{-N/2}\exp\left\{-\frac{1}{2}\left(\mathbf{y}^{\top}\boldsymbol{\Sigma}^{-1}\mathbf{y}-2\mathbf{y}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\boldsymbol{\mu}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\log(|\boldsymbol{\Sigma}|)\right)\right\} \\
&= (2\pi)^{-N/2}\exp\left\{-\frac{1}{2}\left(tr\left\{\mathbf{y}^{\top}\boldsymbol{\Sigma}^{-1}\mathbf{y}\right\}-2\mathbf{y}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\boldsymbol{\mu}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\log(|\boldsymbol{\Sigma}|)\right)\right\} \\
&= (2\pi)^{-N/2}\exp\left\{-\frac{1}{2}\left(vec\left(\mathbf{y}\mathbf{y}^{\top}\right)^{\top}vec\left(\boldsymbol{\Sigma}^{-1}\right)-2\mathbf{y}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\boldsymbol{\mu}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\log(|\boldsymbol{\Sigma}|)\right)\right\},
\end{aligned}
$$

where tr and vec are the trace and vectorization operators, respectively.

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

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

Answer

$$
\begin{aligned}
p(\mathbf{Y}) = &\int_{\mathcal{R}^p}\int_{\mathcal{S}}(2\pi)^{-pN/2}|\boldsymbol{\Sigma}|^{-N/2}\exp\left\{-\frac{1}{2}tr[(\mathbf{S}+N(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}})(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}})^{\top})\boldsymbol{\Sigma}^{-1}]\right\} \\
&\times(2\pi)^{-p/2}\beta_0^{p/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left\{-\frac{\beta_0}{2}tr[(\boldsymbol{\mu}-\boldsymbol{\mu}_0)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^{\top}\boldsymbol{\Sigma}^{-1}]\right\} \\
&\times|\boldsymbol{\Sigma}|^{-(\alpha_0+p+1)/2}\frac{2^{-\alpha_0 p/2}|\boldsymbol{\Psi}_0|^{\alpha_0/2}}{\Gamma_p(\alpha_0/2)}\exp\left\{-\frac{1}{2}tr(\boldsymbol{\Psi}_0\boldsymbol{\Sigma}^{-1})\right\}d\boldsymbol{\Sigma}d\boldsymbol{\mu} \\
= &\frac{(2\pi)^{-\frac{1}{2}(pN+p)}|\boldsymbol{\Psi}_0|^{\alpha_0/2}\beta_0^{p/2}2^{-\alpha_0 p/2}}{\Gamma_p(\alpha_0/2)}\int_{\mathcal{R}^p}\int_{\mathcal{S}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}(N+1+\alpha_0+p+1)} \\
&\times\exp\left\{-\frac{1}{2}tr[(\mathbf{S}+N(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}})(\boldsymbol{\mu}-\hat{\boldsymbol{\mu}})^{\top}+\beta_0(\boldsymbol{\mu}-\boldsymbol{\mu}_0)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^{\top}+\boldsymbol{\Psi}_0)\boldsymbol{\Sigma}^{-1}]\right\}d\boldsymbol{\Sigma}d\boldsymbol{\mu}.
\end{aligned}
$$

We have in the integral the kernel of an Inverse-Wishart distribution, then

$$
\begin{aligned}
p(\mathbf{Y}) &= \frac{\Gamma_p\left(\frac{N+1+\alpha_0}{2}\right)|\mathbf{\Psi}_0|^{\alpha_0/2}\beta_0^{p/2}}{\Gamma_p(\alpha_0/2)\pi^{p(N+1)/2}} \\
&\times \int_{\mathcal{R}^p} |\mathbf{S}+\mathbf{\Psi}_0+(N+\beta_0)(\boldsymbol{\mu}-\boldsymbol{\mu}_n)(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top \\
&+ N\beta_0/(N+\beta_0)(\hat{\boldsymbol{\mu}}-\boldsymbol{\mu}_0)(\hat{\boldsymbol{\mu}}-\boldsymbol{\mu}_0)^\top|^{-(\alpha_n+1)/2}d\boldsymbol{\mu} \\
&= \frac{\Gamma_p\left(\frac{N+1+\alpha_0}{2}\right)|\mathbf{\Psi}_0|^{\alpha_0/2}\beta_0^{p/2}}{\Gamma_p(\alpha_0/2)\pi^{p(N+1)/2}} \\
&\times \int_{\mathcal{R}^p} (|\mathbf{\Psi}_n||1+\beta_n(\boldsymbol{\mu}-\boldsymbol{\mu}_n)\mathbf{\Psi}_n^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top|)^{-\frac{1}{2}(\alpha_n+1)}d\boldsymbol{\mu} \\
&= \frac{\Gamma_p\left(\frac{\alpha_n+1}{2}\right)|\mathbf{\Psi}_0|^{\alpha_0/2}\beta_0^{p/2}}{\Gamma_p(\alpha_0/2)\pi^{p(N+1)/2}}|\mathbf{\Psi}_n|^{-\frac{1}{2}(\alpha_n+1)} \\
&\times \int_{\mathcal{R}^p} [1+\beta_n(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top\mathbf{\Psi}_n^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_n)]^{-\frac{1}{2}(\alpha_n+1)}d\boldsymbol{\mu}.
\end{aligned}
$$

The last equality uses the definition of $\mathbf{\Psi}_n$, β_n and α_n, and the Sylvester's determinant theorem. Observe that we have the kernel of a multivariate t distribution. Then,

$$
\begin{aligned}
p(\mathbf{Y}) &= \frac{\Gamma_p\left(\frac{\alpha_n+1}{2}\right)|\mathbf{\Psi}_0|^{\alpha_0/2}\beta_0^{p/2}}{\Gamma_p(\alpha_0/2)\pi^{p(N+1)/2}}|\mathbf{\Psi}_n|^{-\frac{1}{2}(\alpha_n+1)} \\
&\times \int_{\mathcal{R}^p} \left[1+\frac{1}{\alpha_n+1-p}(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top\left(\frac{\mathbf{\Psi}_n}{\beta_n(\alpha_n+1-p)}\right)^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_n)\right]^{-\frac{1}{2}(\alpha_n+1-p+p)} d\boldsymbol{\mu} \\
&= \frac{\Gamma_p\left(\frac{\alpha_n+1}{2}\right)\Gamma_p\left(\frac{\alpha_n+1-p}{2}\right)|\mathbf{\Psi}_0|^{\alpha_0/2}\beta_0^{p/2}(\alpha_n+1-p)^{p/2}\pi^{p/2}|\mathbf{\Psi}_n|^{-\frac{1}{2}(\alpha_n+1)}}{\Gamma_p(\alpha_0/2)\pi^{p(N+1)/2}\Gamma_p\left(\frac{\alpha_n+1-p+p}{2}\right)\left|\frac{\mathbf{\Psi}_n}{\beta_n(\alpha_n+1-p)}\right|^{-1/2}} \\
&= \frac{\Gamma_p\left(\frac{v_n}{2}\right)}{\Gamma_p\left(\frac{\alpha_0}{2}\right)}\frac{|\mathbf{\Psi}_0|^{\alpha_0/2}}{|\mathbf{\Psi}_n|^{\alpha_n/2}}\left(\frac{\beta_0}{\beta_n}\right)^{p/2}(\pi)^{-Np/2},
\end{aligned}
$$

where $v_n = \alpha_n + 1 - p$.

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

   Answer

$$
\begin{aligned}
p(\mathbf{Y}_0|\mathbf{Y}) &\propto \int_{\mathcal{R}^p}\int_{\mathcal{S}} |\boldsymbol{\Sigma}|^{-1/2}\exp\left\{-\frac{1}{2}tr[(\mathbf{y}_0-\boldsymbol{\mu})(\mathbf{y}_0-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}]\right\}\\
&\quad\times|\boldsymbol{\Sigma}|^{-1/2}\exp\left\{-\frac{\beta_n}{2}tr[(\boldsymbol{\mu}-\boldsymbol{\mu}_n)(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top\boldsymbol{\Sigma}^{-1}]\right\}\\
&\quad\times|\boldsymbol{\Sigma}|^{-(\alpha_n+p+1)/2}\exp\left\{-\frac{1}{2}tr(\boldsymbol{\Psi}_n\boldsymbol{\Sigma}^{-1})\right\}d\boldsymbol{\Sigma}d\boldsymbol{\mu}\\
&\propto \int_{\mathcal{R}^p}|(\mathbf{y}_0-\boldsymbol{\mu})(\mathbf{y}_0-\boldsymbol{\mu})^\top+\beta_n(\boldsymbol{\mu}-\boldsymbol{\mu}_n)(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top+\boldsymbol{\Psi}_n|^{-(\alpha_n+2)/2}d\boldsymbol{\mu}.
\end{aligned}
$$

The last equality uses that there is the kernel of an Inverse Wishart distribution.

Taking into account that

$$
\begin{aligned}
(\mathbf{y}_0-\boldsymbol{\mu})(\mathbf{y}_0-\boldsymbol{\mu})^\top+\beta_n(\boldsymbol{\mu}-\boldsymbol{\mu}_n)(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top &= (1+\beta_n)\left(\boldsymbol{\mu}-\frac{(\mathbf{y}_0+\beta_n\boldsymbol{\mu}_n)}{1+\beta_n}\right)\left(\boldsymbol{\mu}-\frac{(\mathbf{y}_0+\beta_n\boldsymbol{\mu}_n)}{1+\beta_n}\right)^\top\\
&\quad+\frac{\beta_n}{1+\beta_n}(\mathbf{y}_0-\boldsymbol{\mu}_n)(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top.
\end{aligned}
$$

Then,

$$
\begin{aligned}
p(\mathbf{Y}_0|\mathbf{Y}) &\propto \int_{\mathcal{R}^p}|(\mathbf{y}_0-\boldsymbol{\mu})(\mathbf{y}_0-\boldsymbol{\mu})^\top+\beta_n(\boldsymbol{\mu}-\boldsymbol{\mu}_n)(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^\top+\boldsymbol{\Psi}_n|^{-(\alpha_n+2)/2}d\boldsymbol{\mu}\\
&= \int_{\mathcal{R}^p}\left|(1+\beta_n)\left(\boldsymbol{\mu}-\frac{(\mathbf{y}_0+\beta_n\boldsymbol{\mu}_n)}{1+\beta_n}\right)\left(\boldsymbol{\mu}-\frac{(\mathbf{y}_0+\beta_n\boldsymbol{\mu}_n)}{1+\beta_n}\right)^\top\right.\\
&\quad\left.+\frac{\beta_n}{1+\beta_n}(\mathbf{y}_0-\boldsymbol{\mu}_n)(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top+\boldsymbol{\Psi}_n\right|^{-(\alpha_n+2)/2}d\boldsymbol{\mu}\\
&= \int_{\mathcal{R}^p}\left[\left|\underbrace{\boldsymbol{\Psi}_n+\frac{\beta_n}{1+\beta_n}(\mathbf{y}_0-\boldsymbol{\mu}_n)(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top}_{\boldsymbol{\Lambda}_n}\right|\right.\\
&\quad\left.\left|1+(1+\beta_n)\left(\boldsymbol{\mu}-\frac{(\mathbf{y}_0+\beta_n\boldsymbol{\mu}_n)}{1+\beta_n}\right)^\top\frac{1}{\alpha_n+2-p}\left(\frac{\boldsymbol{\Lambda}_n}{\alpha_n+2-p}\right)^{-1}\left(\boldsymbol{\mu}-\frac{(\mathbf{y}_0+\beta_n\boldsymbol{\mu}_n)}{1+\beta_n}\right)\right|\right]^{-(\alpha_n+2-p+p)/2}d\boldsymbol{\mu}\\
&\propto \left|\boldsymbol{\Psi}_n+\frac{\beta_n}{1+\beta_n}(\mathbf{y}_0-\boldsymbol{\mu}_n)(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top\right|^{-(\alpha_n+2)/2}\\
&\quad\times\left|\boldsymbol{\Psi}_n+\frac{\beta_n}{1+\beta_n}(\mathbf{y}_0-\boldsymbol{\mu}_n)(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top\right|^{1/2}\\
&= \left|\boldsymbol{\Psi}_n+\frac{\beta_n}{1+\beta_n}(\mathbf{y}_0-\boldsymbol{\mu}_n)(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top\right|^{-(\alpha_n+1)/2}\\
&\propto \left[1+(\mathbf{y}_0-\boldsymbol{\mu}_n)^\top\frac{1}{\alpha_n+1-p}\left(\frac{\boldsymbol{\Psi}_n(1+\beta_n)}{(\alpha_n+1-p)\beta_n}\right)^{-1}(\mathbf{y}_0-\boldsymbol{\mu}_n)\right]^{-(\alpha_n+1-p+p)}.
\end{aligned}
$$

The second equality uses the Sylvester's determinant theorem, and that there is the kernel of a multivariate t distribution.

Then, we have that the predictive distribution is a multivariate t distribution centered at $\boldsymbol{\mu}_n$, $\alpha_n + 1 - p$ degrees of freedom, and scale matrix $\frac{\boldsymbol{\Psi}_n(1+\beta_n)}{(\alpha_n+1-p)\beta_n}$.

To show the second statement, let's start by the definition of the predictive density to show that $\mathbf{Y}_0|\mathbf{Y} \sim T_{N_0,M}(\alpha_n - M + 1, \mathbf{X}_0\mathbf{B}_n, \mathbf{I}_{N_0} + \mathbf{X}_0\mathbf{V}_n\mathbf{X}_0^\top, \boldsymbol{\Psi}_n)$.

$$\begin{aligned}
\pi(\mathbf{Y}_0|\mathbf{Y}) \propto \int_{\mathcal{S}}\int_{\mathcal{B}} \Bigg\{ & |\boldsymbol{\Sigma}|^{-N_0/2} \exp\left\{-\frac{1}{2} tr[(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B})^\top(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B})\boldsymbol{\Sigma}^{-\mathbf{1}}]\right\} \\
& \times |\boldsymbol{\Sigma}|^{-K/2} \exp\left\{-\frac{1}{2} tr[(\mathbf{B} - \mathbf{B}_n)^\top\mathbf{V}_n^{-1}(\mathbf{B} - \mathbf{B}_n)\boldsymbol{\Sigma}^{-\mathbf{1}}]\right\} \\
& \times |\boldsymbol{\Sigma}|^{-(\alpha_n+M+1)/2} \exp\left\{-\frac{1}{2} tr[\boldsymbol{\Psi}_n\boldsymbol{\Sigma}^{-\mathbf{1}}]\right\}\Bigg\} d\mathbf{B}d\boldsymbol{\Sigma} \\
= \int_{\mathcal{S}}\int_{\mathcal{B}} \Bigg\{ & |\boldsymbol{\Sigma}|^{-(N_0+K+\alpha_n+M+1)/2} \exp\Bigg\{-\frac{1}{2} tr\Big[\big((\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B})^\top(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}) \\
& + (\mathbf{B} - \mathbf{B}_n)^\top\mathbf{V}_n^{-1}(\mathbf{B} - \mathbf{B}_n) + \boldsymbol{\Psi}_n\big)\boldsymbol{\Sigma}^{-\mathbf{1}}\Big]\Bigg\}\Bigg\} d\mathbf{B}d\boldsymbol{\Sigma}.
\end{aligned}$$

Setting $\mathbf{M} = (\mathbf{X}_0^\top\mathbf{X}_0 + \mathbf{V}_n^{-1})$, and $\mathbf{B}_* = \mathbf{M}^{-1}(\mathbf{V}_n\mathbf{B}_n + \mathbf{X}_0^\top\mathbf{Y}_0)$, we have that $(\mathbf{B} - \mathbf{B}_*)^\top\mathbf{M}(\mathbf{B} - \mathbf{B}_*) + \mathbf{B}_n^\top\mathbf{V}_n^{-1}\mathbf{B}_n + \mathbf{Y}_0^\top\mathbf{Y}_0 - \mathbf{B}_*^\top\mathbf{M}\mathbf{B}_* = (\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B})^\top(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}) + (\mathbf{B} - \mathbf{B}_n)^\top\mathbf{V}_n^{-1}(\mathbf{B} - \mathbf{B}_n)$. Then,

$$\begin{aligned}
\pi(\mathbf{Y}_0|\mathbf{Y}) \propto \int_{\mathcal{S}} & |\boldsymbol{\Sigma}|^{-(N_0+K+\alpha_n+M+1)/2} \\
& \times \exp\left\{-\frac{1}{2} tr[(\boldsymbol{\Psi}_n + \mathbf{B}_n^\top\mathbf{V}_n^{-1}\mathbf{B}_n + \mathbf{Y}_0^\top\mathbf{Y}_0 - \mathbf{B}_*^\top\mathbf{M}\mathbf{B}_*)\boldsymbol{\Sigma}^{-\mathbf{1}}]\right\} \\
& \times \int_{\mathcal{B}} \exp\left\{-\frac{1}{2} tr[(\mathbf{B} - \mathbf{B}_*)^\top\mathbf{M}(\mathbf{B} - \mathbf{B}_*)\boldsymbol{\Sigma}^{-\mathbf{1}}]\right\} d\mathbf{B}d\boldsymbol{\Sigma}.
\end{aligned}$$

The latter is the kernel of a matrix normal distribution, thus

$$\begin{aligned}
\pi(\mathbf{Y}_0|\mathbf{Y}) \propto \int_{\mathcal{S}} & |\boldsymbol{\Sigma}|^{-(N_0+\alpha_n+M+1)/2} \\
& \times \exp\left\{-\frac{1}{2} tr[(\boldsymbol{\Psi}_n + \mathbf{B}_n^\top\mathbf{V}_n^{-1}\mathbf{B}_n + \mathbf{Y}_0^\top\mathbf{Y}_0 - \mathbf{B}_*^\top\mathbf{M}\mathbf{B}_*)\boldsymbol{\Sigma}^{-\mathbf{1}}]\right\} d\boldsymbol{\Sigma}
\end{aligned}$$

This is the kernel of an inverse-Wishart distribution, then

$$\pi(\mathbf{Y}_0|\mathbf{Y}) \propto \left|\boldsymbol{\Psi}_n + \mathbf{B}_n^\top\mathbf{V}_n^{-1}\mathbf{B}_n + \mathbf{Y}_0^\top\mathbf{Y}_0 - \mathbf{B}_*^\top\mathbf{M}\mathbf{B}_*\right|^{-(N_0+\alpha_n)/2}.$$

Setting $\mathbf{C}^{-1} = \mathbf{I}_{N_0} + \mathbf{X}_0\mathbf{V}_n\mathbf{X}_0^\top$ such that $\mathbf{C} = \mathbf{I}_{N_0} - \mathbf{X}_0(\mathbf{X}_0^\top\mathbf{X}_0 + \mathbf{V}_n^{-1})^{-1}\mathbf{X}_0^\top$ (see footnote 4 in Chapter 4), then $\mathbf{B}_n^\top\mathbf{V}_n^{-1}\mathbf{B}_n + \mathbf{Y}_0^\top\mathbf{Y}_0 - \mathbf{B}_*^\top\mathbf{M}\mathbf{B}_* = (\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}_n)^\top\mathbf{C}(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}_n)$. This is done following exactly same procedure as deducing the predictive distribution in the linear regression model in the book. Thus,

$$\begin{aligned}\pi(\mathbf{Y}_0|\mathbf{Y}) &\propto \left|\boldsymbol{\Psi}_n + (\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}_n)^\top\mathbf{C}(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}_n)\right|^{-(N_0+\alpha_n)/2} \\ &\propto \left|\mathbf{I}_{N_0} + \mathbf{C}(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}_n)\boldsymbol{\Psi}^{-1}(\mathbf{Y}_0 - \mathbf{X}_0\mathbf{B}_n)^\top\right|^{-(\alpha_n+1-M+N_0+M-1)/2}.\end{aligned}$$

The second proportionality follows from the Sylvester's theorem. Observe that this is the kernel of a matrix t distribution with $\alpha_n + 1 - M$ degrees of freedom, location $\mathbf{X}_0\mathbf{B}_n$ and scale matrices $\boldsymbol{\Psi}_n$ and $\mathbf{C}^{-1} = \mathbf{I}_{N_0} + \mathbf{X}_0\mathbf{V}_n\mathbf{X}_0^\top$.

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

Answer

Taking into account that

$$\begin{aligned}\delta^* &= \delta_0 + \mathbf{y}^\top\mathbf{y} + \boldsymbol{\beta}_0^\top\mathbf{B}_0^{-1}\boldsymbol{\beta}_0 - \boldsymbol{\beta}_n^\top\mathbf{B}_n^{-1}\boldsymbol{\beta}_n \\ &= \delta_0 + \mathbf{y}^\top\mathbf{y} + \boldsymbol{\beta}_0^\top\mathbf{B}_0^{-1}\boldsymbol{\beta}_0 - (\mathbf{B}_0^{-1}\boldsymbol{\beta}_0 + \mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}})^\top\mathbf{B}_n(\mathbf{B}_0^{-1}\boldsymbol{\beta}_0 + \mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}) \\ &= \delta_0 + \mathbf{y}^\top\mathbf{y} - \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\mathbf{B}_n\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} - 2\hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\mathbf{B}_n\mathbf{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{\beta}_0^\top(\mathbf{B}_0^{-1} - \mathbf{B}_0^{-1}\mathbf{B}_n\mathbf{B}_0^{-1})\boldsymbol{\beta}_0 \\ &\quad - \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} \\ &= \delta_0 + \mathbf{y}^\top\mathbf{y} - \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}^\top(\mathbf{X}^\top\mathbf{X} - \mathbf{X}^\top\mathbf{X}\mathbf{B}_n\mathbf{X}^\top\mathbf{X})\hat{\boldsymbol{\beta}} \\ &\quad - 2\hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\mathbf{B}_n\mathbf{B}_0^{-1}\boldsymbol{\beta}_0 + \boldsymbol{\beta}_0^\top(\mathbf{B}_0^{-1} - \mathbf{B}_0^{-1}\mathbf{B}_n\mathbf{B}_0^{-1})\boldsymbol{\beta}_0.\end{aligned}$$

Observe that

$$\begin{aligned}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) &= \mathbf{y}^\top\mathbf{y} - 2\hat{\boldsymbol{\beta}}\mathbf{X}^\top\mathbf{y} + \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} \\ &= \mathbf{y}^\top\mathbf{y} - 2\hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top(\mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\mu}}) + \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} \\ &= \mathbf{y}^\top\mathbf{y} - \hat{\boldsymbol{\beta}}^\top\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}},\end{aligned}$$

where $\mathbf{y} = \mathbf{X}\hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\mu}}$, and $\mathbf{X}^\top\hat{\boldsymbol{\mu}} = 0$.

The following matrix identities are useful [43]:

$$(\mathbf{D} + \mathbf{E})^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}(\mathbf{D}^{-1} + \mathbf{E}^{-1})^{-1}\mathbf{D}^{-1},$$

and

$$(\mathbf{D} + \mathbf{E})^{-1} = \mathbf{D}^{-1}(\mathbf{E}^{-1} + \mathbf{D}^{-1})\mathbf{E}^{-1}.$$

Using these identities,

$$\begin{aligned}[(\mathbf{X}^\top\mathbf{X})^{-1}+\mathbf{B}_0]^{-1} &= \mathbf{X}^\top\mathbf{X}-\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X}+\mathbf{B}_0^{-1})^{-1}\mathbf{X}^\top\mathbf{X}\\ &= \mathbf{B}_0^{-1}-\mathbf{B}_0^{-1}(\mathbf{X}^\top\mathbf{X}+\mathbf{B}_0^{-1})^{-1}\mathbf{B}_0^{-1}\\ &= \mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X}+\mathbf{B}_0^{-1})^{-1}\mathbf{B}_0^{-1}.\end{aligned}$$

Then,

$$\begin{aligned}\delta^* &= \delta_0+(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}})^\top(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}})+\hat{\boldsymbol{\beta}}^\top[(\mathbf{X}^\top\mathbf{X})^{-1}+\mathbf{B}_0]^{-1}\hat{\boldsymbol{\beta}}\\ &- 2\hat{\boldsymbol{\beta}}[(\mathbf{X}^\top\mathbf{X})^{-1}+\mathbf{B}_0]^{-1}\boldsymbol{\beta}_0+\boldsymbol{\beta}_0^\top[(\mathbf{X}^\top\mathbf{X})^{-1}+\mathbf{B}_0]^{-1}\boldsymbol{\beta}_0\\ &= \delta_0+(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}})^\top(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}})\\ &+ (\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}_0)^\top[(\mathbf{X}^\top\mathbf{X})^{-1}+\mathbf{B}_0]^{-1}(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}_0).\end{aligned}$$

In a similar way for the second part,

$$\begin{aligned}(\mathbf{V}_0+(\mathbf{X}^\top\mathbf{X})^{-1})^{-1} &= \mathbf{V}_0^{-1}-\mathbf{V}_0^{-1}(\mathbf{V}_0^{-1}+\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{V}_0^{-1}\\ &= \mathbf{X}^\top\mathbf{X}-\mathbf{X}^\top\mathbf{X}(\mathbf{V}_0^{-1}+\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}\\ &= \mathbf{X}^\top\mathbf{X}((\mathbf{X}^\top\mathbf{X})^{-1}+\mathbf{V}_0)^{-1}\mathbf{V}_0^{-1},\end{aligned}$$

we use these results and some algebra to show that $\mathbf{B}_0^\top\mathbf{V}_0^{-1}\mathbf{B}_0+\widehat{\mathbf{B}}^\top\mathbf{X}^\top\mathbf{X}\widehat{\mathbf{B}}-\mathbf{B}_n^\top\mathbf{V}_n^{-1}\mathbf{B}_n=(\hat{\mathbf{B}}-\mathbf{B}_0)^\top\mathbf{V}_n(\hat{\mathbf{B}}-\mathbf{B}_0)$ taking into account that $\mathbf{V}_n=(\mathbf{V}_0^{-1}+\mathbf{X}^\top\mathbf{X})^{-1}$ and $\widehat{\mathbf{B}}=(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$.

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

Answer

Taking into account that $(\mathbf{A}+\mathbf{B})^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}(\mathbf{A}^{-1}+\mathbf{B}^{-1})^{-1}\mathbf{A}^{-1}$ [43], then we observe that $(\mathbf{B}_n^{-1}-\mathbf{B}_n^{-1}\mathbf{M}^{-1}\mathbf{B}_n^{-1})=(\mathbf{B}_n+(\mathbf{X}_0^\top\mathbf{X}_0)^{-1})^{-1}$, where $(\mathbf{B}_n+(\mathbf{X}_0^\top\mathbf{X}_0)^{-1})^{-1}=\mathbf{X}_0^\top\mathbf{X}_0-\mathbf{X}_0^\top\mathbf{X}_0(\mathbf{B}_n^{-1}+\mathbf{X}_0^\top\mathbf{X}_0)^{-1}\mathbf{X}_0^\top\mathbf{X}_0=\mathbf{X}_0^\top\mathbf{X}_0-\mathbf{X}_0^\top\mathbf{X}_0\mathbf{M}^{-1}\mathbf{X}_0^\top\mathbf{X}_0$, thus

$$\begin{aligned}\boldsymbol{\beta}_n^\top(\mathbf{B}_n^{-1}-\mathbf{B}_n^{-1}\mathbf{M}^{-1}\mathbf{B}_n^{-1})\boldsymbol{\beta}_n &= \boldsymbol{\beta}_n^\top(\mathbf{X}_0^\top\mathbf{X}_0-\mathbf{X}_0^\top\mathbf{X}_0\mathbf{M}^{-1}\mathbf{X}_0^\top\mathbf{X}_0)\boldsymbol{\beta}_n\\ &= \boldsymbol{\beta}_n^\top\mathbf{X}_0^\top(\mathbf{I}_{N_0}-\mathbf{X}_0\mathbf{M}^{-1}\mathbf{X}_0^\top)\mathbf{X}_0\boldsymbol{\beta}_n\\ &= \boldsymbol{\beta}_n^\top\mathbf{X}_0^\top\mathbf{C}\mathbf{X}_0\boldsymbol{\beta}_n\\ &= \boldsymbol{\beta}_{**}^\top\mathbf{C}\boldsymbol{\beta}_{**}.\end{aligned}$$

Let's show that $\boldsymbol{\beta}_{**}=\mathbf{X}_0\boldsymbol{\beta}_n$,

$$
\begin{aligned}
\boldsymbol{\beta}_{**} &= \mathbf{C}^{-1}\mathbf{X}_0\mathbf{M}^{-1}\mathbf{B}_n^{-1}\boldsymbol{\beta}_n \\
&= (\mathbf{I}_{N_0} + \mathbf{X}_0\mathbf{B}_n\mathbf{X}_0^\top)\mathbf{X}_0\mathbf{M}^{-1}\mathbf{B}_n^{-1}\boldsymbol{\beta}_n \\
&= (\mathbf{I}_{N_0} + \mathbf{X}_0\mathbf{B}_n\mathbf{X}_0^\top)\mathbf{X}_0(\mathbf{B}_n - \mathbf{B}_n((\mathbf{X}_0^\top\mathbf{X}_0)^{-1} + \mathbf{B}_n)^{-1}\mathbf{B}_n)\mathbf{B}_n^{-1}\boldsymbol{\beta}_n \\
&= (\mathbf{I}_{N_0} + \mathbf{X}_0\mathbf{B}_n\mathbf{X}_0^\top)(\mathbf{X}_0\boldsymbol{\beta}_n - \mathbf{X}_0\mathbf{B}_n((\mathbf{X}_0^\top\mathbf{X}_0)^{-1} + \mathbf{B}_n)^{-1}\boldsymbol{\beta}_n) \\
&= \mathbf{X}_0\boldsymbol{\beta}_n - \mathbf{X}_0\mathbf{B}_n((\mathbf{X}_0^\top\mathbf{X}_0)^{-1} + \mathbf{B}_n)^{-1}\boldsymbol{\beta}_n + \mathbf{X}_0\mathbf{B}_n\mathbf{X}_0^\top\mathbf{X}_0\boldsymbol{\beta}_n \\
&- \mathbf{X}_0\mathbf{B}_n\mathbf{X}_0^\top\mathbf{X}_0\mathbf{B}_n((\mathbf{X}_0^\top\mathbf{X}_0)^{-1} + \mathbf{B}_n)^{-1}\boldsymbol{\beta}_n \\
&= \mathbf{X}_0\boldsymbol{\beta}_n - \mathbf{X}_0\mathbf{B}_n[((\mathbf{X}_0^\top\mathbf{X}_0)^{-1} + \mathbf{B}_n)^{-1} - \mathbf{X}_0^\top\mathbf{X}_0 + \mathbf{X}_0^\top\mathbf{X}_0\mathbf{B}_n((\mathbf{X}_0^\top\mathbf{X}_0)^{-1} + \mathbf{B}_n)^{-1}]\boldsymbol{\beta}_n.
\end{aligned}
$$

Using that $(\mathbf{A}+\mathbf{B})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{A}+\mathbf{B})^{-1}$, we observe that the expression in brackets is equal to $\mathbf{0}$, then we have the result.

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

Answer

$$
\begin{aligned}
(\mathbf{Y}-\mathbf{X}\mathbf{B})^\top(\mathbf{Y}-\mathbf{X}\mathbf{B}) &= (\mathbf{Y}-\mathbf{X}\hat{\mathbf{B}}+\mathbf{X}\hat{\mathbf{B}}-\mathbf{X}\mathbf{B})^\top(\mathbf{Y}-\mathbf{X}\hat{\mathbf{B}}+\mathbf{X}\hat{\mathbf{B}}-\mathbf{X}\mathbf{B}) \\
&= (\mathbf{Y}-\mathbf{X}\hat{\mathbf{B}})^\top(\mathbf{Y}-\mathbf{X}\hat{\mathbf{B}}) + 2(\mathbf{Y}-\mathbf{X}\hat{\mathbf{B}})^\top(\mathbf{X}\hat{\mathbf{B}}-\mathbf{X}\mathbf{B}) \\
&+ (\mathbf{X}\mathbf{B}-\mathbf{X}\hat{\mathbf{B}})^\top(\mathbf{X}\mathbf{B}-\mathbf{X}\hat{\mathbf{B}}) \\
&= \mathbf{S} + (\mathbf{B}-\hat{\mathbf{B}})^\top\mathbf{X}^\top\mathbf{X}(\mathbf{B}-\hat{\mathbf{B}}),
\end{aligned}
$$

given that $(\mathbf{Y}-\mathbf{X}\hat{\mathbf{B}})^\top(\mathbf{X}\hat{\mathbf{B}}-\mathbf{X}\mathbf{B}) = \hat{\mathbf{U}}^\top\mathbf{X}(\hat{\mathbf{B}}-\mathbf{B})$, using that $\hat{\mathbf{B}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$ which implies $\mathbf{X}^\top\mathbf{X}\hat{\mathbf{B}} = \mathbf{X}^\top\mathbf{Y} = \mathbf{X}^\top\mathbf{X}\hat{\mathbf{B}} + \mathbf{X}^\top\hat{\mathbf{U}}$, then $\mathbf{X}^\top\hat{\mathbf{U}} = \mathbf{0}$.

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

This is the most famous Richard Price's example developed in the Appendix of the Bayes' theorem paper [3]. Here, we implicitly use *Laplace's Rule of Succession* to solve this question. In particular, if we were a priori uncertain about the probability the Sun will rise on a specified day, we can assume a prior uniform distribution over (0,1), that is, a beta (1,1) distribution. Then, what is the probability that the Sun will rise tomorrow?

Answer

This exercise is an application of the Bernoulli-beta model. Thus, the likelihood is given by a binomial distribution where the probability of success is θ, $p(\mathbf{y}|\theta) \propto \theta^{\sum_{i=1}^N y_i}(1-\theta)^{N-\sum_{i=1}^N y_i}$. In addition, the prior distribution is beta, that is, $\pi(\theta) \propto \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}$, where $\alpha_0 = \beta_0 = 1$. Then, the predictive distribution that the sun will rise tomorrow is $p(Y_0 = 1|\mathbf{y}) = \frac{1+S}{2+N}$, where $S = \sum_{i=1}^N y_i$ is the number of successes (the Sun rise). $\frac{1+S}{2+N}$ is known

as the *Laplace's Rule of Succession* that was introduced by Laplace in the 18^{th} century in the course of treating the sunrise problem.

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 supporters of Hillary vs Trump?
    - What is the probability that sampling one hundred independent individuals 44, 40 and 16 support Hillary, Trump and other, respectively?

    Answer

R code. Multinomial-Dirichlet model: Polling 2016 USA presidential race

```
set.seed(010101)
# Multinomial-Dirichlet example:
# Polling 2016 USA presidential race
y <- c(411, 373, 149)
# Clinton, Trump, Other
# Public Policy Polling September 27-28,
# 2016 five-way race
alpha0 <- rep(1, 3)
# Hyperparameters: non-informative distribution
alphan <- alpha0 + y
S <- 100000
# Sample draws of posterior
thetas <- MCMCpack::rdirichlet(S, alphan)
colnames(thetas) <- c("Clinton", "Trump", "Other")
head(thetas)
       Clinton     Trump     Other
[1,] 0.4211346 0.4188607 0.1600046
[2,] 0.4244207 0.4224523 0.1531270
[3,] 0.4349268 0.3843953 0.1806779
[4,] 0.4533499 0.4005530 0.1460972
[5,] 0.4381799 0.3968502 0.1649699
[6,] 0.4436852 0.3971321 0.1591827
dif <- thetas[,1] - thetas[,2]
# Difference of shares Hillary vs Trump
data <- data.frame(dif)
names(data) <- c("Difference")
library(ggplot2)
p <- ggplot(data) +
  geom_histogram(aes(x = Difference), binwidth = 0.01) +
  geom_vline(xintercept=0.0, lwd=1, colour="red") +
  ggtitle("Percentage difference Clinton vs Trump 2016
    presidential race") + xlab("Percentage Difference") +
    ylab("")
difmcmc <- coda::mcmc(dif)
# Declaring a MCMC object
summary(difmcmc)

Iterations = 1:1e+05
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1e+05

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

Mean             SD       Naive SE Time-series SE
4.062e-02      2.996e-02      9.474e-05      9.474e-05
2. Quantiles for each variable:

2.5%       25%       50%       75%     97.5%
-0.01817  0.02033  0.04058  0.06089  0.09923
CW <- mean(difmcmc>0)
CW
0.91339
```

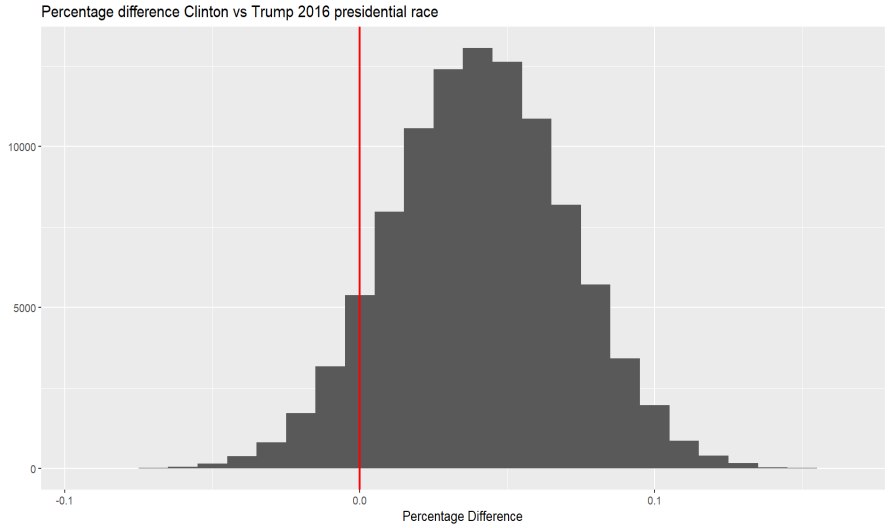

FIGURE 3.1
Percentage difference: Hillary Clinton vs Donald Trump, five-way race.

There is a 95% probability that the percentage difference between Hillary and Trump according to this poll is (-1.8%, 9.9%). The probability of Hillary having more supporters is 91.3%

R code. Multinomial-Dirichlet model: Polling 2016 USA presidential race

```
# Predictive distribution by simulation
y0 <- c(44, 40, 16)
Pred <- apply(thetas, 1, function(p) {rmultinom(1, size =
    sum(y0), prob = p)})
sum(sapply(1:S, function(s) {sum(Pred[,s] == y0) == 3}))/S
0.00825
# Predictive distribution by 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)
0.00850
```

The probability that from one hundred random selected people 44 support Hillary, 40 support Trump and 16 support other candidate is 0.85%.

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

Answer

R code. Math test example continues

```
set.seed(010101)
N <- 50
# Sample size
muhat <- 102
# Sample mean
sig2hat <- 10
# Sample variance
# Hyperparameters
mu0 <- 100
beta0 <- 1
delta0 <- 0.001
alpha0 <- 0.001
S <- 100000
# Posterior draws
alphan <- alpha0 + N
deltan <- sig2hat*(N - 1) + delta0 + beta0*N/(beta0 + N)*(
    muhat - mu0)^2
sig2Post <- invgamma::rinvgamma(S, shape = alphan, rate =
    deltan)
summary(sig2Post)
betan <- beta0 + N
mun <- (beta0*mu0 + N*muhat)/betan
muPost_t <- ((deltan/(alphan*betan))^0.5)*rt(S, alphan) +
    mun
c1 <- rgb(173,216,230,max = 255, alpha = 50, names = "lt.
    blue")
c2 <- rgb(255,192,203, max = 255, alpha = 50, names = "lt.
    pink")
hist(muPost_t, main = "Histogram: Posterior mean", xlab = "
    Posterior mean", col = c1)
muPost_tq <- quantile(muPost_t, c(0.025, 0.5, 0.975))
muPost_tq
2.5%      50%    97.5%
101.0837 101.9608 102.8435
PmuPost_tcutoff <- mean(muPost_t > cutoff)
PmuPost_tcutoff
0.01087
```

Figure 3.2 shows the posterior distribution of the mean.

We have that the 95% credible interval is (101.08, 102.84), and the probability of having a value greater than 103 is 1.09%.

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

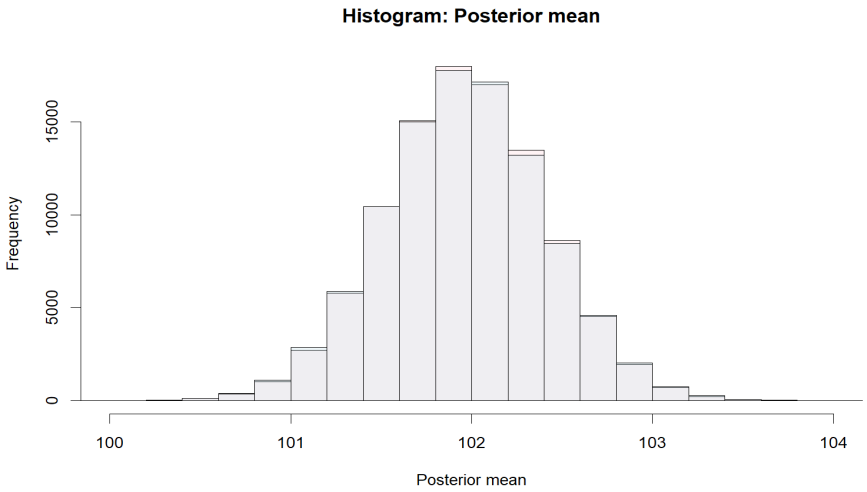

FIGURE 3.2
Histogram of the posterior distribution of the mean

(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.

Answer

R code. Demand of electricity

```
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 = "")
DataUtEst <- DataUt %>%
filter(Electricity != 0)
attach(DataUtEst)
# Dependent variable: Monthly consumption (kWh) in log
Y <- log(Electricity)
N <- length(Y)
# Regressors quantity including intercept
X <- cbind(LnPriceElect, IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem, Children, Lnincome, 1)
# Regressor without price
Xnew <- cbind(IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem, Children, Lnincome, 1)
# 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)
}
# Hyperparameters
d0 <- 0.001
a0 <- 0.001
# Empirical Bayes: Obtain c0 maximizing the log marginal likelihood
c0 <- 1000
EB <- optim(c0, fn = LogMarLikLM, method = "Brent", lower = 0.0001, upper = 10^6, X = X)
EBnew <- optim(c0, fn = LogMarLikLM, method = "Brent", lower = 0.0001, upper = 10^6, X = Xnew)
# Change of order to take into account the -1 in the LogMarLikLM function
BFEM <- exp(EBnew$value - EB$value)
BFEM
71897025
```

The Bayes factor based on the empirical Bayes of the model with electricity price versus the model without electricity price is equal to 71897938, this gives very strong evidence to include the price in the specification.

14. **Utility demand**

Use the file *Utilities.csv* to estimate a multivariate linear regression model where $\mathbf{Y}_i = [\log(\text{electricity}_i)\ \log(\text{water}_i)\ \log(\text{gas}_i)]$ 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 4.3. Omit households that do not consume any of the utilities in this exercise.

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

(a) Find the posterior mean estimates and the highest posterior density intervals at 95% of $\mathbf{B}$ and $\mathbf{\Sigma}$.

(b) 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?

(c) 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.

Answer

We find that the Bayes factor of the baseline model (log(Income)) versus the two alternative models using income in dollars and thousand dollars are 108925764 and 0.1089261. The former gives strong evidence in favor of the baseline model, whereas the latter gives positive evidence for the model using the income in thousand dollars. This result despite that the location coefficients are the same in the two alternative specifications, except for the change in scale of the coefficients associated with income. This example shows that Bayes factors are sensitive to units of measure, and consequently, it is relevant to think carefully about the priors when performing hypothesis testing using a Bayesian framework. Observe that a nice feature in Bayesian inference is that we followed the same conceptual framework (Bayes factor) in the previous exercise and this exercise. In one

hand, the previous exercise is an example of nested models, that is, one model is a restricted version of a more general model. On the other hand, this exercise is an example of non-nested models. This is not the case in the Frequentist approach. The statistical framework is not the same when testing nested and non-nested models.

R code. Utilities demand: Multivariate regression, posterior inference

```
rm(list = ls())
set.seed(010101)
library(dplyr)
# Electricity demand
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)
Y <- cbind(log(Electricity), log(Water), log(Gas))
X <- cbind(LnPriceElect, LnPriceWater, LnPriceGas, IndSocio1
    , IndSocio2, Altitude, Nrooms, HouseholdMem, Children,
    Lnincome, 1)
M <- dim(Y)[2]
K <- dim(X)[2]
N <- dim(Y)[1]
# Hyperparameters
B0 <- matrix(0, K, M)
c0 <- 1000
V0 <- c0*diag(K)
Psi0 <- c0*diag(M)
a0 <- M
# 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))
Bs <- sapply(1:s, function(s) {MixMatrix::rmatrixt(n = 1,
    mean=Bn, U = Vn,V = Psin, df = an + 1 - M)})
summary(coda::mcmc(t(Bs)))
SIGMs <- t(sapply(1:s, function(l) {gdata::lowerTriangle(
    SIGs[,,l], diag=TRUE, byrow=FALSE)}))
summary(coda::mcmc(SIGMs))
hdiBs <- HDInterval::hdi(t(Bs), credMass = 0.95) # Highest
    posterior density credible interval
hdiBs
hdiSIG <- HDInterval::hdi(SIGMs, credMass = 0.95) # Highest
    posterior density credible interval
hdiSIG

```

R code. Utilities demand: Multivariate regression, Bayes factors

```
# Log marginal function (multiply by -1 due to minimization)
LogMarLikLM <- function(X, c0){
  c10 <- c0[1]; c20 <- c0[2]
  k <- dim(X)[2]
  N <- dim(X)[1]
  # Hyperparameters
  V0 <- c10*diag(K)
  Psi0 <- c20*diag(M)
  # 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
  # Log marginal likelihood
  logpy <- (N*M/2)*log(1/pi)+(a0/2)*log(det(Psi0)) - (an/2)*
    log(det(Psin)) + (M/2)*(log(det(Vn)) - log(det(V0))) +
    lgamma(an/2)-lgamma(a0/2)
  return(-logpy)
}
c0 <- rep(1000, 2)
LogML <- LogMarLikLM(X=X, c0 = c0)
# Using income in dollars as regressor
Xnew <- cbind(LnPriceElect, LnPriceWater, LnPriceGas,
    IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem,
    Children, exp(Lnincome), 1)
LogMLnew <- LogMarLikLM(X=Xnew, c0 = c0)
# Bayes factor
BF12 <- exp(LogMLnew - LogML)
BF12
# Using income in thousand dollars as regressor
XnewT <- cbind(LnPriceElect, LnPriceWater, LnPriceGas,
    IndSocio1, IndSocio2, Altitude, Nrooms, HouseholdMem,
    Children, exp(Lnincome)/1000, 1)
LogMLnewT <- LogMarLikLM(X=XnewT, c0 = c0)
# Bayes factor
BF13 <- exp(LogMLnewT - LogML)
BF13

```

R code. Utilities demand: Multivariate regression, predictive distribution

```
# Predictive distribution
Xpred <- c(log(0.15), log(0.70), log(0.75), 1, 0, 0, 2, 3,
    1, log(500), 1)
Mean <- Xpred%*%Bn
Hn <- 1+t(Xpred)%*%Vn%*%Xpred
UtilDemand <- exp(replicate(s, MixMatrix::rmatrixt(n = 1,
    mean=Mean, U = Hn, V = Psin, df = an + 1 - M)))
ElePred <- UtilDemand[1,1,]
WatPred <- UtilDemand[1,2,]
GasPred <- UtilDemand[1,3,]
data <- data.frame(cbind(ElePred, WatPred, GasPred)) #Data
    frame
annotations1 <- data.frame(
x = round(quantile(data$ElePred, c(0.025, 0.5, 0.975)),1),
y = c(600, 1000, 600),
label = c("2.5%:", "50%:", "97.5%:")
)
annotations2 <- data.frame(
x = round(quantile(data$WatPred, c(0.025, 0.5, 0.975)),1),
y = c(600, 1000, 600),
label = c("2.5%:", "50%:", "97.5%:")
)
annotations3 <- data.frame(
x = round(quantile(data$GasPred, c(0.025, 0.5, 0.975)),1),
y = c(600, 1000, 600),
label = c("2.5%:", "50%:", "97.5%:")
)
require(ggplot2) # Cool figures
require(ggpubr) # Multiple figures in one page
require(latex2exp) # LaTeX equations in figures
fig1 <- ggplot(data = data, aes(ElePred)) + geom_histogram(
    bins = 40, color = "#000000", fill = "#0099F8") +  xlab(
    "kWh") + ylab("Frequency") + ggtitle("Electricity") +
    xlim(0, 1050) + geom_text(data = annotations1, aes(x = x
    , y = y, label = paste(label, x)), size = 3, fontface =
    "bold")
fig2 <- ggplot(data = data, aes(WatPred)) + geom_histogram(
    bins = 40, color = "#000000", fill = "#0099F8") +  xlab(
    TeX("$M^3$")) + ylab("Frequency") +  ggtitle("Water") +
    xlim(0, 100) + geom_text(data = annotations2, aes(x = x,
     y = y, label = paste(label, x)), size = 3, fontface = "
    bold")
fig3 <- ggplot(data = data, aes(GasPred)) + geom_histogram(
    bins = 40, color = "#000000", fill = "#0099F8") +  xlab(
    TeX("$M^3$")) + ylab("Frequency") +  ggtitle("Gas") +
    xlim(0, 80) + geom_text(data = annotations3, aes(x = x,
    y = y, label = paste(label, x)), size = 3, fontface = "
    bold")

```

Figures 3.3, 3.4 and 3.5 show the marginal predictive distributions of electricity, water and gas for the reference household. The median predictive values are kWh 168.8, M^3 12.3 and M^3 10.1, respectively. In addition, the 95% credible intervals are (27.7, 1028.9), (1.5, 98.7) and (1.5, 67.5) for electricity, water and gas.

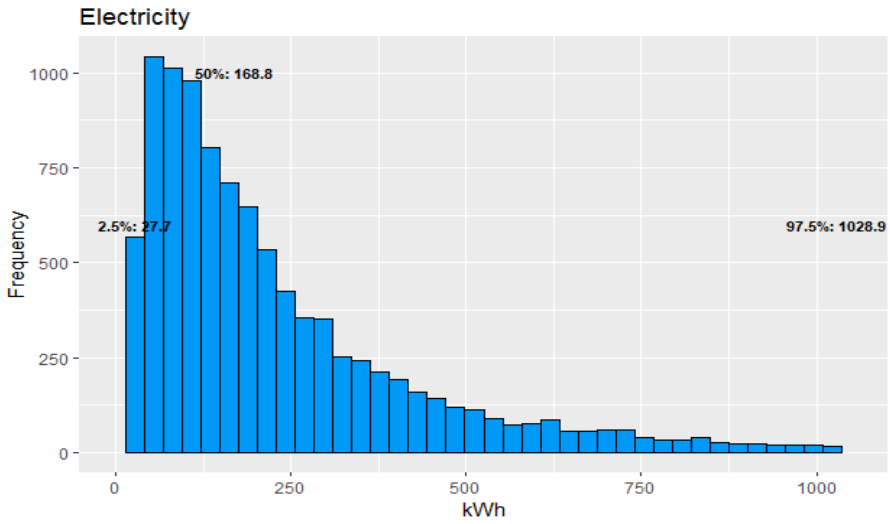

FIGURE 3.3
Histogram using the posterior predictive distribution of electricity demand

15. **Ph.D. students sleeping hours [1, 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. 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

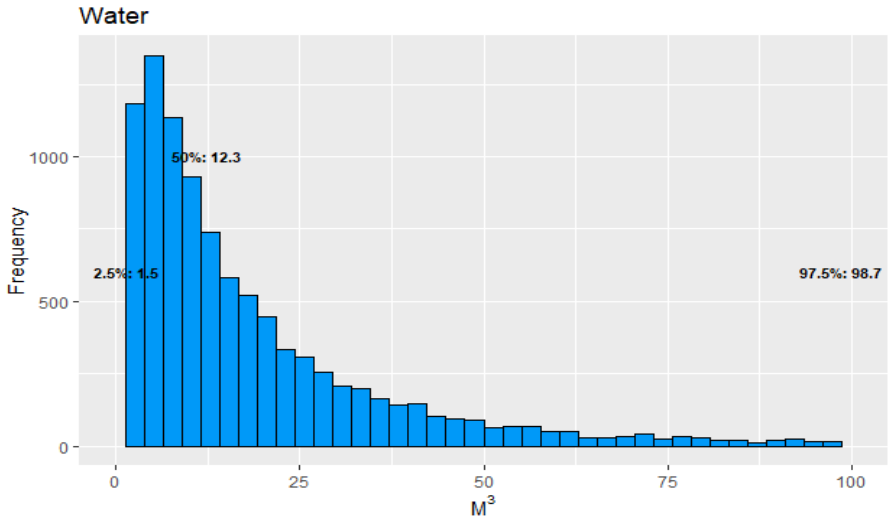

FIGURE 3.4
Histogram using the posterior predictive distribution of water demand

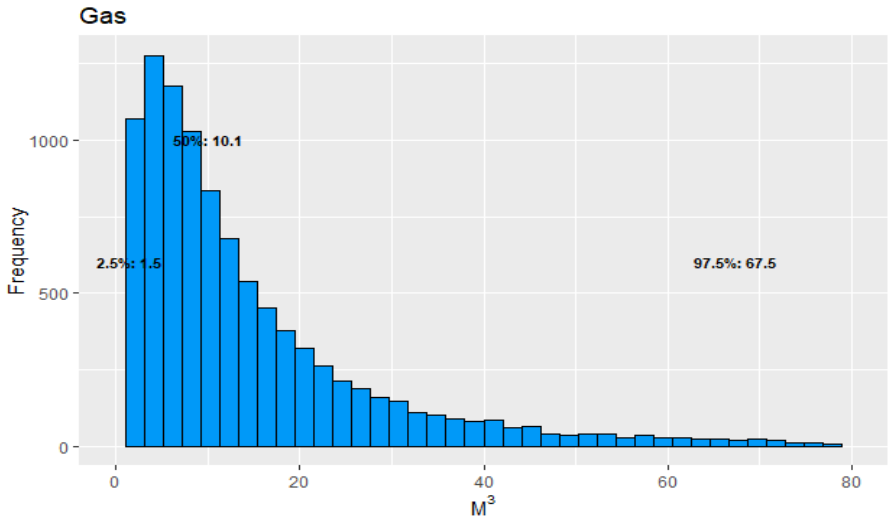

FIGURE 3.5
Histogram using the posterior predictive distribution of gas demand

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

Answer

We know from the Bernoulli-beta model that the posterior distribution of the proportion of Ph.D. students that sleep at least 6 hours is beta, $\theta|\boldsymbol{y} \sim B(\alpha_n, \beta_n)$, $\alpha_n = \alpha_0 + 15$ and $\beta_n = \beta_0 + 37$, where $\alpha_0 = 1.44$ and $\beta_0 = 2.57$.

The following code shows how to get the posterior distribution in this exercise.

R code. Ph.D. students sleeping hours: Posterior distribution

```
rm(list = ls()); set.seed(010101)
# Function to calibrate the hyperparameters
elec <- function(theta, perc, q){
  E1 <- perc[1] - pbeta(q[1], theta[1], theta[2])
  E2 <- perc[2] - pbeta(q[2], theta[1], theta[2])
  loss <- E1^2 + E2^2
  return(loss)
}
q1 <- 0.4; q2 <- 0.75; q <- c(q1, q2) # Quantiles
p1e <- 0.6; p2e <- 0.95; perc <- c(p1e, p2e) # Cumulate
    probability from experts
theta0 <- c(1, 1) # Parameters
elec(theta0, perc, q)
Res <- optim(theta0, elec, perc = perc, q = q)
p <- seq(0, 1, length = 500)
a <- Res$par[1]; b <- Res$par[2]
pbeta(q1, a, b) # P(p <= q1) = p1e
pbeta(q2, a, b) # P(p <= q2) = p2e
s <- 15 # Success; f <- 37 # Failures
prior <- dbeta(p,a,b)
likelihood <- dbeta(p, s + 1, f + 1) #Seeing the binomial
    distribution as a likelihood is a beta distribution!!!
post <- dbeta(p, a+s, b+f)
plot(p,post,type="l",ylab="Density", main = "Proportion of
    students who sleep at least 6 hours per day",lty=2,lwd
    =3,col=gray.colors(1,start=0.1))
lines(p,likelihood,lty=1,lwd=3,col=gray.colors(1,start=0.4))
lines(p,prior,lty=3,lwd=3,col=gray.colors(1,start=0.6))
legend(.7,4,c("Prior","Likelihood","Posterior"), lty=c
    (3,1,2),lwd=c(3,3,3),col=c(col=gray.colors(1,start=0.6),
    col=gray.colors(1,start=0.4),col=gray.colors(1,start
    =0.1)))
S <- 10000; postdraws1 <- rbeta(S, a+s, b+f)
mean(postdraws1); quantile(postdraws1, c(0.025, 0.975))
# Experts
p <- c(seq(0.05, 0.55, by = 0.05)) # Probabilities
prior <- c(0.05, 0.07, 0.1, 0.12, 0.15, 0.17, 0.15, 0.11,
    0.06, 0.01, 0.01) # Expert assessment
llikelihood <- s * log(p) + f * log(1 - p) # Log likelihood
likelihood2 <- exp(llikelihood -max(llikelihood)) # Scale
    the likelihood for plotting
product <- likelihood2 * prior # Proportional to posterior
posterior  <- product/sum(product) # Posterior
# Figure
plot(p, prior, type = "l", ylab="Probability", lwd=2, lty=2,
     ylim=c(0,0.99), xlim = c(0, 0.6), col=gray.colors(1,
    start=0.1), main = "Posterior distribution: Proportion
    sleep hours")
lines(p, posterior,type = "l",lwd=2,col=gray.colors(1,start
    =0.6))
lines(p,likelihood2,type = "l",lwd=4,col=gray.colors(1,start
    =0.3))
legend(0.45,0.85,c('Prior','Likelihood','Posterior'),lty=c
    (2,1,1), col=c(gray.colors(1,start=0.1),gray.colors(1,
    start=0.3),gray.colors(1,start=0.6)),lwd=2)
postdraws2 <- sample(p, S, replace = TRUE, prob = posterior)
mean(postdraws2); quantile(postdraws2, c(0.025, 0.975))
```

Figures 3.6 and 3.7 show the prior, likelihood, and posterior distributions of the proportion of Ph.D. students who sleep at least 6 hours per day based on the binomial-beta and binomial-discrete models. The mean under both models is 29%, and the 95% credible intervals are (18%,42%) and (20%,40%), respectively.

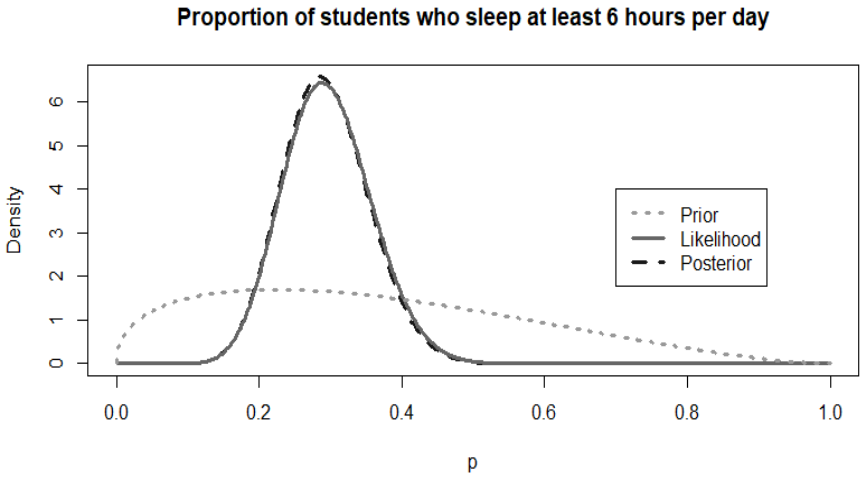

FIGURE 3.6
Posterior distribution binomial-beta model: Proportion Ph.D. students that sleep at least 6 hours per day

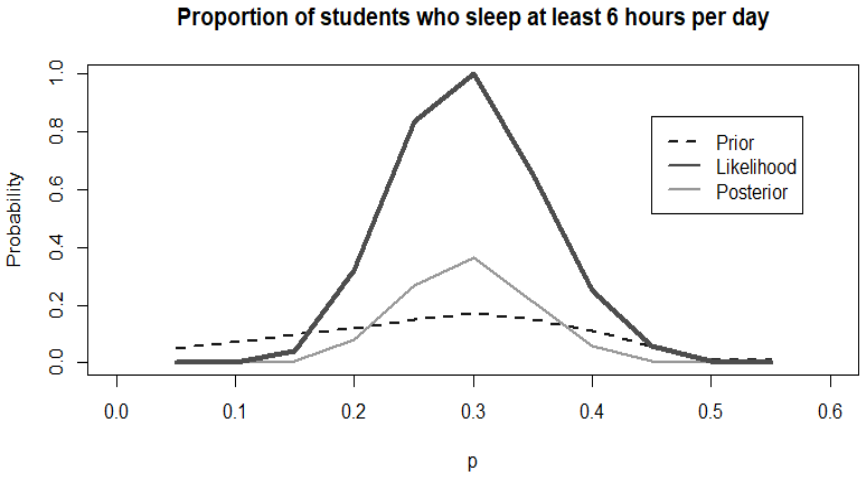

FIGURE 3.7
Posterior distribution binomial-discrete model: Proportion Ph.D. students that sleep at least 6 hours per day

4

Simulation methods

Solutions of 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

   Answer

   The posterior distribution is

$$\pi(\mu, \sigma^2|\boldsymbol{y}) \propto (\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \mu)^2\right\} \times \exp\left\{-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right\} \times \left(\frac{1}{\sigma^2}\right)^{\alpha_0/2+1} \exp\left\{-\frac{\delta_0}{2\sigma^2}\right\}.$$

   Thus, the conditional posterior distribution of μ is

$$\begin{aligned}\pi(\mu, \sigma^2|\boldsymbol{y}) &\propto \exp\left\{-\frac{1}{2}\left[\frac{1}{\sigma^2}\sum_{i=1}^{N}(y_i - \mu)^2 + \frac{1}{\sigma_0^2}(\mu - \mu_0)^2\right]\right\} \\ &\propto \exp\left\{-\frac{1}{2}\left[\mu^2\left(\frac{1}{\sigma^2/N} + \frac{1}{\sigma_0^2}\right) - 2\mu\left(\frac{\bar{y}}{\sigma^2/N} + \frac{\mu_0}{\sigma_0^2}\right)\right]\right\}.\end{aligned}$$

We set $\mu_n = \sigma_n^2 \left(\frac{\bar{y}}{\sigma^2/N} + \frac{\mu_0}{\sigma_0^2} \right)$ and $\sigma_n^2 = \left(\frac{1}{\sigma^2/N} + \frac{1}{\sigma_0^2} \right)^{-1}$. Thus,

$$\begin{aligned}\pi(\mu, \sigma^2|\boldsymbol{y}) &\propto \exp\left\{ -\frac{1}{2\sigma_n^2} \left[\mu^2 - 2\mu\mu_n + \mu_n^2 - \mu_n^2 \right] \right\} \\ &\propto \exp\left\{ -\frac{1}{2\sigma_n^2} (\mu - \mu_n)^2 \right\}.\end{aligned}$$

This is the kernel of a normal distribution, that is, $\mu|\sigma^2, \boldsymbol{y} \sim N(\mu_n, \sigma_n^2)$.

The conditional posterior distribution of σ^2 is given by

$$\begin{aligned}\pi(\sigma^2|\mu, \boldsymbol{y}) &\propto (\sigma^2)^{-N/2} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \mu)^2 \right\} \\ &\quad \times \left(\frac{1}{\sigma^2} \right)^{\alpha_0/2+1} \exp\left\{ -\frac{\delta_0}{2\sigma^2} \right\} \\ &\propto (\sigma^2)^{-N/2-\alpha_0/2-1} \exp\left\{ -\frac{1}{2\sigma^2} \left[\sum_{i=1}^{N} (y_i - \mu)^2 + \delta_0 \right] \right\}.\end{aligned}$$

Thus, $\sigma^2|\mu, \boldsymbol{y} \sim IG(\alpha_n/2, \delta_n/2)$, where $\alpha_n = N + \alpha_0$ and $\delta_n = \sum_{i=1}^{N} (y_i - \mu)^2 + \delta_0 = N\hat{\sigma}^2 + N(\bar{y} - \mu)^2 + \delta_0$ given that $\sum_{i=1}^{N} (y_i - \bar{y}) = 0$, where $\bar{y}$ and $\hat{\sigma}$ are the mean and standard deviation estimates.

As we have the conditional posterior distributions, we can use the Gibbs sampling algorithm to perform inference in this model. The following code shows how to do it.

R code. Gibbs sampler: The math example

```
rm(list = ls())
set.seed(010101)
N <- 50
# Sample size
muhat <- 102
# Sample mean
sig2hat <- 100
# Sample variance

# Hyperparameters
mu0 <- 100
sig20 <- 100
delta0 <- 0.001
alpha0 <- 0.001

MCMC <- 10000; burnin <- 1000; S <- MCMC + burnin
keep <- (burnin+1):S
# Posterior draws
alphan <- alpha0 + N
sig2Post <- rep(NA, S)
muPost <- rep(NA, S)
sig2 <- sig20
for(s in 1:S){
  sig2n <- (1/(sig2/N)+1/sig20)^(-1)
  mun <- sig2n*(muhat/(sig2/N)+mu0/sig20)
  mu <- rnorm(1, mun, sig2n^0.5)
  muPost[s] <- mu
  deltan <- N*(sig2hat + (muhat - mu)^2)
  sig2 <- invgamma::rinvgamma(1, shape = alphan, rate =
    deltan)
  sig2Post[s] <- sig2
}
sig2s <- coda::mcmc(sig2Post[keep])
mus <- coda::mcmc(muPost[keep])
summary(sig2s)
summary(mus)
hist(mus, main = "Histogram: Posterior mean", xlab = "
    Posterior mean", col = "blue", breaks = 50)
muPost_tq <- quantile(mus, c(0.025, 0.5, 0.975))
muPost_tq
cutoff <- 103
PmuPost_tcutoff <- mean(mus > cutoff)
PmuPost_tcutoff
```

Figure 4.1 shows the histogram of the draws of the posterior mean of the math test results. The posterior mean and median are 101.94 and 101.95,

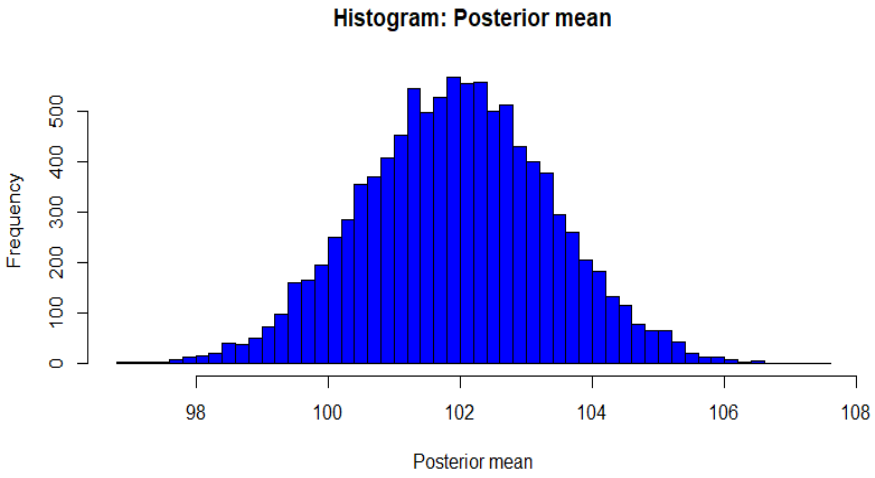

FIGURE 4.1
Histogram of posterior draws of mean: Math test

and the 95% credible interval is (99.14, 104.79).

2. Show that the Gibbs sampler is a particular case of the Metropolis-Hastings where the acceptance probability is equal to 1.

   Answer

   Without loss of generality let's assume that there are two blocks, $\boldsymbol{\theta} = [\boldsymbol{\theta}_1 \ \boldsymbol{\theta}_2]^\top$ such that the Metropolis-Hastings (M-H) algorithm generates a candidate sample $(\boldsymbol{\theta}_1^c)$ using a proposal distribution $q(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}^{(s-1)})$. The candidate is accepted with probability:

$$\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) = \min\left\{1, \frac{q(\boldsymbol{\theta}_1^{(s-1)}|\boldsymbol{\theta}^c)\pi(\boldsymbol{\theta}^c|\boldsymbol{y})}{q(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}^{(s-1)})\pi(\boldsymbol{\theta}^{(s-1)}|\boldsymbol{y})}\right\}.$$

   For Gibbs sampling, the candidate $\boldsymbol{\theta}_1^c$ is drawn directly from the *full conditional distribution*, so $q(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}^{(s-1)}) = \pi(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})$. Since Gibbs sampling uses the full conditional distributions as the proposal, the key terms simplify. In particular, $q(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}^{(s-1)}) = \pi(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}_2^c, \boldsymbol{y})$, and similarly $q(\boldsymbol{\theta}_1^{(s-1)}|\boldsymbol{\theta}^c) = \pi(\boldsymbol{\theta}_1^{(s-1)}|\boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})$. Thus, the acceptance probability is

given by

$$\begin{aligned}\alpha(\boldsymbol{\theta}^{(s-1)}, \boldsymbol{\theta}^c) &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}_1^{(s-1)}|\boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}^c|\boldsymbol{y})}{\pi(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}_2^c, \boldsymbol{y})\pi(\boldsymbol{\theta}^{(s-1)}|\boldsymbol{y})}\right\} \\ &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}_1^{(s-1)}|\boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}_2^c, \boldsymbol{y})\pi(\boldsymbol{\theta}_2^c|\boldsymbol{y})}{\pi(\boldsymbol{\theta}_1^c|\boldsymbol{\theta}_2^c, \boldsymbol{y})\pi(\boldsymbol{\theta}_1^{(s-1)}|\boldsymbol{\theta}_2^{(s-1)}, \boldsymbol{y})\pi(\boldsymbol{\theta}_2^{(s-1)}|\boldsymbol{y})}\right\} \\ &= 1,\end{aligned}$$

due to $\boldsymbol{\theta}_2^{(s-1)} = \boldsymbol{\theta}_2^c$. Thus, the Gibbs sampling algorithm is implicitly a M-H algorithm where the acceptance probability is 1.

3. Implement a Metropolis-Hastings (M-H) 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.

   Answer

   The following code shows to program the M-H to get draws from the Cauchy distribution, $C(0, 1)$, using as proposal a standard normal distribution and a Student's t distribution with 5 degree of freedom.

R code. Metropolis-Hastings algorithm: Cauchy distribution

```
rm(list = ls()); set.seed(010101)
S <- 100000; df <- 5
theta1 <- runif(S); acept1 <- rep(0, S)
theta2 <- runif(S); acept2 <- rep(0, S)
for (s in 2:S){
  thetac1 <- rnorm(1) # Candidate normal standard
  thetac2 <- rt(1, df = df) # Candidate Students' t
  #Acceptance rate
  a1 <- (dcauchy(thetac1)*dnorm(theta1[s-1]))/(dcauchy(
    theta1[s-1])*dnorm(thetac1))
  a2 <- (dcauchy(thetac2)*dt(theta2[s-1], df = df))/(dcauchy
    (theta2[s-1])*dt(thetac2, df = df))
  U1 <- runif(1)
  if(U1 <= a1){
    theta1[s] <- thetac1; acept1[s] <- 1
  }else{
    theta1[s] <- theta1[s-1]; acept1[s] <- 0
  }
  if(U1 <= a2){
    theta2[s] <- thetac2; acept2[s] <- 1
  }else{
    theta2[s] <- theta2[s-1]; acept2[s] <- 0
  }
}
mean(acept1); mean(acept2)
plot(coda::mcmc(theta1)); plot(coda::mcmc(theta2))
h <- hist(theta1, breaks=50, col="blue", xlab="x", main="
    Cauchy draws from a Metropolis-Hastings algorithm:
    Normal standard proposal")
pfit <- seq(min(theta1),max(theta1),length=50)
yfit<-dcauchy(pfit)
yfit <- yfit*diff(h$mids[1:2])*length(theta1)
lines(pfit, yfit, col="red", lwd=2)
h <- hist(theta2, breaks=50, col="blue", xlab="x", main="
    Cauchy draws from a Metropolis-Hastings algorithm:
    Student's t proposal")
pfit <- seq(min(theta2),max(theta2),length=50)
yfit<-dcauchy(pfit)
yfit <- yfit*diff(h$mids[1:2])*length(theta2)
lines(pfit, yfit, col="red", lwd=2)
```

Figures 4.2 and 4.3 show the histograms of the posterior draws using the normal and Student's t distributions, along with the density of the Cauchy distribution. The spike in the posterior draws from the standard normal proposal is due to the algorithm being stuck for many iterations at a par-

ticular value, as the normal distribution has lighter tails compared to the Cauchy distribution. The convergence using the normal proposal is slower than when using the Student's t proposal, owing to the lighter tails of the former.

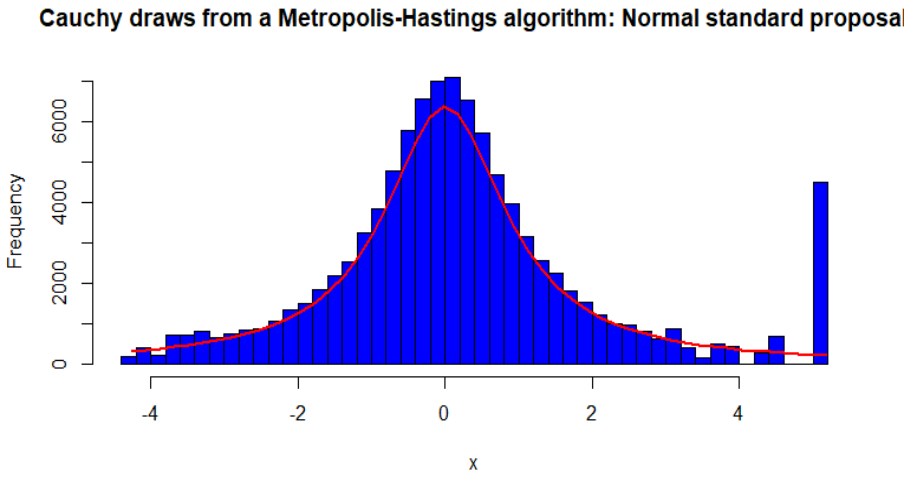

FIGURE 4.2
Histogram of posterior draws of beta distribution and the density of the beta distribution.

4. This exercise was proposed by Professor Hedibert Freitas Lopes, who cites [45] 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

$$\pi(\theta_1, \theta_2|\boldsymbol{y}) \propto \exp\left\{-\frac{1}{2}(\theta_1^2\theta_2^2 + \theta_1^2 + \theta_2^2 - 8\theta_1 - 8\theta_2)\right\}.$$

Answer

The following code demonstrates the implementation of the M-H and HMC algorithms for this exercise. For the M-H algorithm, the number of MCMC iterations, burn-in period, and thinning parameters are set to 20,000, 1,000, and 20, respectively. The variance of the proposal distribution is tuned to achieve an acceptance rate close to 25%.

For the HMC algorithm, the number of iterations and burn-in period are set to 5,000 and 500, respectively. The step size and number of leapfrog iterations are set to 0.001 and 100, and the mass matrix is defined as $\boldsymbol{M} = \boldsymbol{I}_2$. These parameters yield an acceptance rate near 65%.

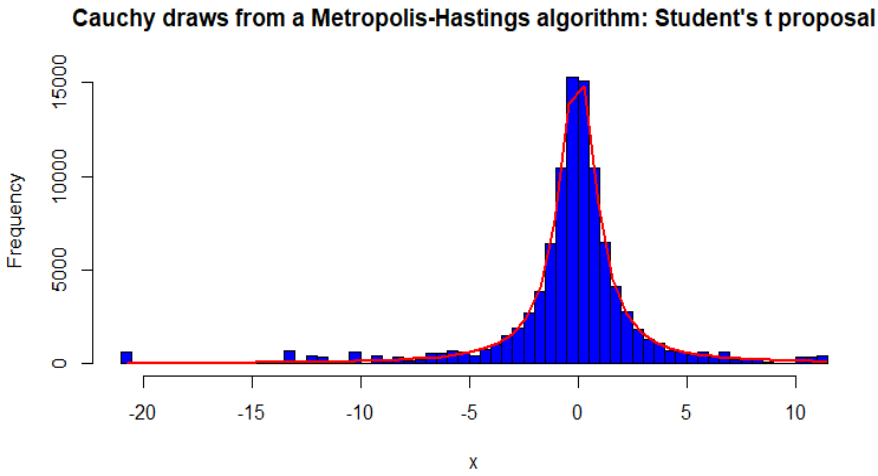

FIGURE 4.3
Histogram of posterior draws of beta distribution and the density of the beta distribution.

Figures 4.4 and 4.6 present the contour plots and posterior draws from the M-H and HMC algorithms for the joint posterior distribution. It can be observed that this distribution is bimodal, and both algorithms successfully capture this feature.

R code. Metropolis-Hastings and Hamiltonian Monte Carlo algorithms

```
rm(list = ls()); set.seed(010101)
# Posterior distribution
PostDist <- function(theta){
  theta1 <- theta[1]; theta2 <- theta[2]
  post <- exp(-0.5*(theta1^2*theta2^2+theta1^2+theta2^2-8*
    theta1-8*theta2))
  return(post)
}
# Metropolis-Hastings
MH <- function(theta, c2){
  thetac <- MASS::mvrnorm(1, mu = theta, Sigma = c2*diag(2))
  a <- PostDist(thetac)/PostDist(theta)
  U <- runif(1)
  if(U <= a){
    theta <- thetac
    accept <- 1
  }else{
    theta <- theta
    accept <- 0
  }
  return(list(theta = theta, accept = accept))
}
# Posterior draws M-H
S <- 20000; burnin <- 1000; thin <- 20; tot <- S + burnin
K <- 2; c2 <- 1.5
thetaPostMH <- matrix(NA, tot, K)
AcceptMH <- rep(NA, tot)
thetaMH <- c(0, 0)
for(s in 1:tot){
  ResMH <- MH(thetaMH, c2 = c2)
  thetaMH <- ResMH$theta
  thetaPostMH[s,] <- thetaMH
  AcceptMH[s] <- ResMH$accept
}
keep <- seq((burnin), tot, thin)
mean(AcceptMH[keep])
thetaPostMHMCMC <- coda::mcmc(thetaPostMH[keep,])
plot(thetaPostMHMCMC)
coda::autocorr.plot(thetaPostMHMCMC)
# Contour plot
ngrid <- 400
theta1 <- seq(-1, 6, length = ngrid)
theta2 <- seq(-1, 6, length = ngrid)
f <- matrix(0, ngrid, ngrid)
for (i in 1:ngrid){
  for (j in 1:ngrid){
    f[i,j] = PostDist(c(theta1[i],theta2[j]))
  }
}
plot(thetaPostMH[keep,], xlim=range(theta1), ylim=range(
    theta2), pch=16, col=grey(0.8), xlab=expression(theta
    [1]), ylab=expression(theta[2]))
contour(theta1,theta2,f, drawlabels=FALSE, add=TRUE, col = "
    blue", lwd = 1.2)
title("Random walk Metropolis-Hastings")
```

R code. Metropolis-Hastings and Hamiltonian Monte Carlo algorithms

```
HMC <- function(theta, epsilon, L, M){
  Minv <- solve(M); thetat <- theta
  K <- length(thetat)
  mom <- t(mvtnorm::rmvnorm(1, rep(0, K), M))
  logPost_Mom_t <- log(PostDist(thetat)) +  mvtnorm::dmvnorm
    (t(mom), rep(0, K), M, log = TRUE)
  theta1 <- theta[1]; theta2 <- theta[2]
  for(l in 1:L){
    if(l == 1 | l == L){
      mom <- mom + 0.5*epsilon*(-0.5*c(2*theta1*theta2^2+2*
    theta1-8, 2*theta2*theta1^2+2*theta2-8))
      theta <- theta + epsilon*Minv%*%mom
    }else{
      mom <- mom + epsilon*(-0.5*c(2*theta1*theta2^2+2*
    theta1-8, 2*theta2*theta1^2+2*theta2-8))
      theta <- theta + epsilon*Minv%*%mom
    }
  }
  logPost_Mom_star <- log(PostDist(theta)) +  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 HMC
S <- 5000; burnin <- 500; tot <- S + burnin
epsilon <- 0.0025;  L <- 100; M <- diag(2)
thetaPostHMC <- matrix(NA, tot, K)
ProbAcceptHMC  <- rep(NA, tot)
thetaHMC <- c(0, 0)
for(s in 1:tot){
  ResHMC <- HMC(theta = thetaHMC, epsilon = epsilon, L = L,
    M = M)
  thetaHMC <- ResHMC$theta
  thetaPostHMC[s,] <- thetaHMC
  ProbAcceptHMC[s] <- ResHMC$Prob
}
keep <- burnin:S
summary(ProbAcceptHMC[keep])
thetaPostHMCMCMC <- coda::mcmc(thetaPostHMC[keep,])
plot(thetaPostHMCMCMC); coda::autocorr.plot(thetaPostHMCMCMC
    )
plot(thetaPostHMC[keep,], xlim=range(theta1), ylim=range(
    theta2), pch=16, col=grey(0.8),
xlab=expression(theta[1]), ylab=expression(theta[2]))
contour(theta1,theta2,f, drawlabels=FALSE, add=TRUE, col = "
    blue", lwd = 1.2)
title("Hamiltonian Monte Carlo")
```

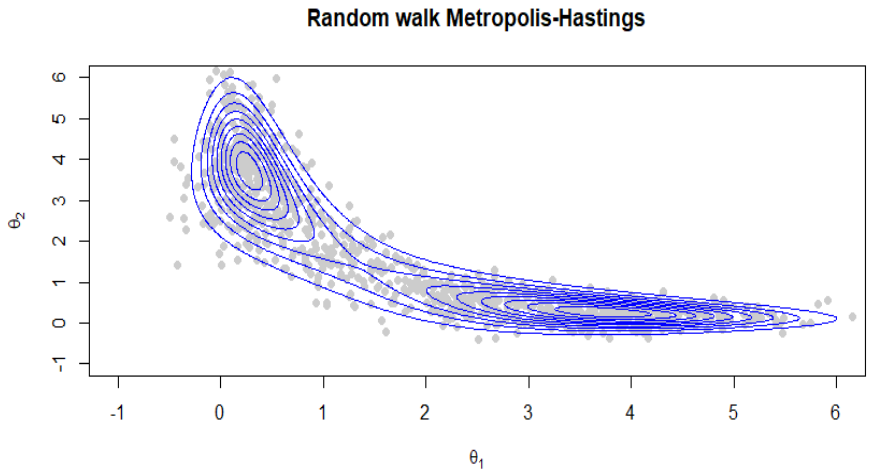

FIGURE 4.4
Contour plot: Metropolis-Hastings posterior draws.

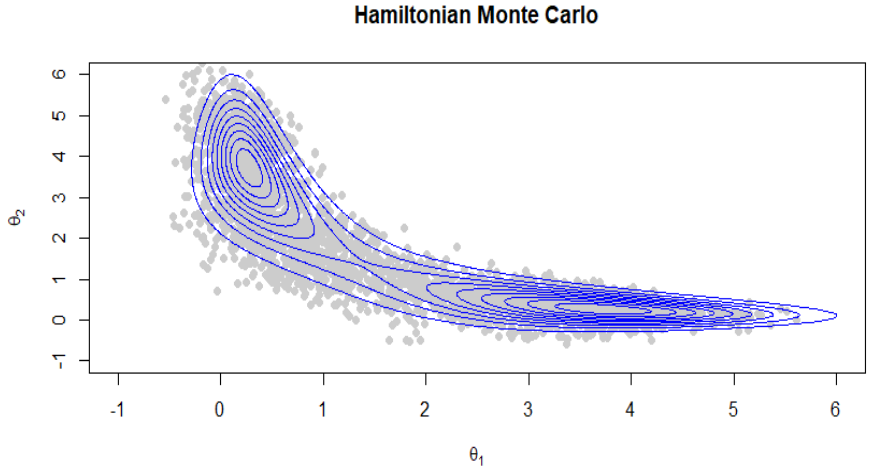

FIGURE 4.5
Contour plot: Hamiltonian Monte Carlo posterior drwas.

5. **Ph.D. students sleeping hours continues**

   (a) Use importance sampling based on a $U(0,1)$ proposal to obtain draws

of $\boldsymbol{\theta}|\boldsymbol{y} \sim B(16.55, 39.57)$ 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.

Answer

The posterior distribution of the Bernoulli-beta model is

$$p(\theta|\boldsymbol{y}) = \frac{\prod_{i=1}^{N} \theta^{y_i}(1-\theta)^{1-y_i}\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)p(\boldsymbol{y})},$$

where

$$\begin{aligned} p(\boldsymbol{y}) &= \int_{\boldsymbol{\Theta}} \frac{\prod_{i=1}^{N} \theta^{y_i}(1-\theta)^{1-y_i}\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)} d\theta \\ &= \int_{\boldsymbol{\Theta}} \frac{\theta^{\sum_{i=1}^{N} y_i+\alpha_0-1}(1-\theta)^{N-\sum_{i=1}^{N} y_i+\beta_0-1}\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)} d\theta \\ &= \frac{B(\alpha_n,\beta_n)}{B(\alpha_0,\beta_0)} \int_{\boldsymbol{\Theta}} \frac{\theta^{\sum_{i=1}^{N} y_i+\alpha_0-1}(1-\theta)^{N-\sum_{i=1}^{N} y_i+\beta_0-1}}{B(\alpha_n,\beta_n)} d\theta \\ &= \frac{B(\alpha_n,\beta_n)}{B(\alpha_0,\beta_0)}. \end{aligned}$$

The integral is equal to 1 due to being over the support of $B(\alpha_n, \beta_n)$ distribution.

R code. Importance sampling: Bernoulli-Beta model

```
rm(list = ls()); set.seed(010101); S <- 50000
# Importance sampling from standard normal proposal
an <- 16.55; bn <- 39.57 # Posterior parameters
theta <- runif(S) # Proposal
ws <- dbeta(theta, an, bn) # Weights
wstars <- ws/sum(ws) # Standardized weights
L <- 20000 # Size of posterior sample
thetaBeta <- sample(theta, L, replace = TRUE, prob = wstars)
     # Posterior draws
# Figure
h <- hist(thetaBeta, breaks=50, col="blue", xlab="x", main="
    Beta draws from importance sampling: Uniform (0,1)
    proposal")
pfit <- seq(min(thetaBeta),max(thetaBeta),length=50)
yfit<-dbeta(pfit, an, bn)
yfit <- yfit*diff(h$mids[1:2])*length(thetaBeta)
lines(pfit, yfit, col="red", lwd=2)
a0 <- 1.44; b0 <- 2.57 # Hyperparameters
s <- 15; n <- 52 # Data
LogMarLik <- lbeta(an, bn)-lbeta(a0, b0) # Marginal
    likelihood
LogMarLik
# Gelfand-Day method
LikPrior <- function(theta){
  Liki <- theta^s*(1-theta)^(n-s)
  Priori <- dbeta(theta, a0, b0)
  LikPriori <- 1 / Liki * Priori
  return(LikPriori)
}
LogMarLikGD <- log(1/mean(sapply(thetaBeta, LikPrior)))
LogMarLikGD
```

Figure 4.6 shows the histogram of the posterior draws obtained using the importance sampling algorithm with a uniform proposal. The algorithm performs well. Additionally, the analytic solution for the log marginal likelihood is -32.75, while the Gelfand-Dey method yields -32.76.

6. Example 4.1 in [14] 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 $\mu_t \sim N(0, \sqrt{1})$.

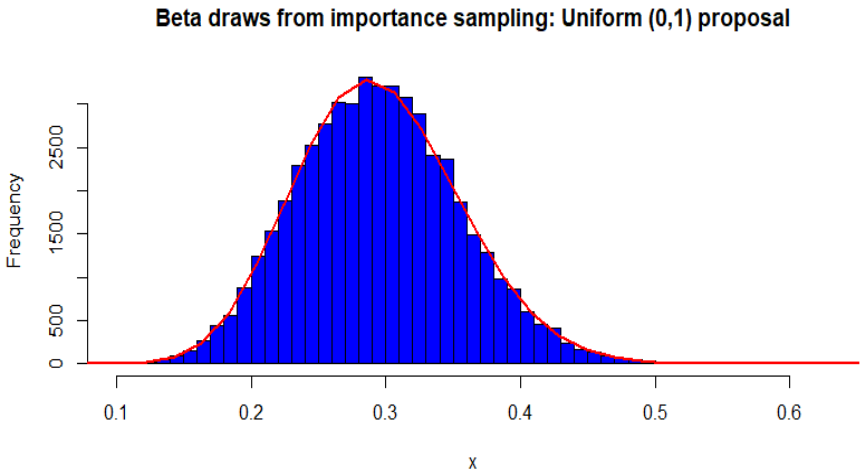

FIGURE 4.6
Importance sampling: Beta distribution.

- Perform sequential importance sampling in this example
- Perform particle (Bootstrap) filtering in this example
- Estimate the marginal likelihood in this example

Answer

The following code shows to perform sequential importance sampling and particle (Bootstrap) filtering in the Example 1 in [14]. Take into account that we use same strategy as in the dynamic linear model example in the book to calculate the mean and standard deviation of the posterior distribution.

R code. Sequential importance sampling and particle (Bootstrap) filtering: Gordon (1993) Example 1.

```
rm(list = ls()); set.seed(010101)
S <- 50000 # Number of particles
sigma_w <- 10^0.5  # State noise
sigma_mu <- 1  # Observation noise
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)  # Initial
    state
y_obs[1] <- rnorm(1, mean = theta_true[1]^2/20, sd = sigma_
    mu)
for (t in 2:T) {
  meant <- 0.5*theta_true[t-1] + 25*(theta_true[t-1]/(1+
    theta_true[t-1]^2)) + 8*cos(1.2*t)
  theta_true[t] <-  rnorm(1, mean = meant, sd = sigma_w)
  y_obs[t] <- rnorm(1, mean = theta_true[t]^2/20, sd = sigma
    _mu)
}
plot(theta_true, type = "l"); plot(y_obs, type = "l")
# 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)
weights[, 1] <- dnorm(y_obs[1], mean = particles[, 1]^2/20,
    sd = sigma_mu)
weightsSt[, 1] <- weights[, 1] / sum(weights[, 1])
# Sequential updating
for (t in 2:T) {
  particles[, t] <- rnorm(S, mean = 0.5*particles[, t-1] +
    25*(particles[, t-1]/(1+particles[, t-1]^2)) + 8*cos(1.2
    *t), sd = sigma_w)  # Compute weights
  weights[, t] <- weightsSt[, t-1] * dnorm(y_obs[t], mean =
    particles[, t]^2/20, sd = sigma_mu)
  weightsSt[, t] <- weights[, t] / sum(weights[, t])
}
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)
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)
```

R code. Sequential importance sampling and particle (Bootstrap) filtering: Gordon (1993) Example 1.

```
# Particle filtering
particles <- matrix(0, nrow = S, ncol = T)
particlesT <- matrix(0, nrow = S, ncol = T)
weights <- matrix(0, nrow = S, ncol = T)
weightsSt <- matrix(0, nrow = S, ncol = T)
weightsSTT <- matrix(1/S, nrow = S, ncol = T)
logalphas <- 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 = particles[, 1]^2/20,
    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 <- txtProgressBar(min = 0, max = T, style = 3)
for (t in 2:T) {
  # Propagate particles
  particles[, t] <- rnorm(S, mean = 0.5*particles[, t-1] +
    25*(particles[, t-1]/(1+particles[, t-1]^2)) + 8*cos(1.2
    *t), sd = sigma_w)
  # Compute weights
  logalphas[, t] <- dnorm(y_obs[t], mean = particles[, t]^2/
    20, 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]
  }
  setTxtProgressBar(pb, t)
}
close(pb)
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
```

R code. Sequential importance sampling and particle (Bootstrap) filtering: Gordon (1993) Example 1.

```
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,
    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)
MargLik <- colMeans(weights)
plot(MargLik, type = "l", xlab = "Time", ylab = "Marginal
    likelihood", main = "Marginal likelihood")
```

Figure 4.7 shows the true trajectory of the state vector (black line), the posterior mean (blue line), and the $\pm 2\hat{\sigma}_\theta$ posterior estimate (light blue shaded area) from the sequential importance sampling algorithm. We observe that there is no light blue shaded area after time period 40, and the algorithm exhibits poor performance after $t = 75$. This is due to weight degeneracy.

Figure 4.8 shows the performance of particle filtering (see Figure 3 in [14]). 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. This is due to restarting the system every stage by resampling, which in turn mitigates the weight degeneracy issue.

Finally, Figure 4.9 shows the posterior estimate of marginal likelihood based on the particle filtering algorithm.

7. **Ph.D. students sleeping hours continues**

   - Perform the diagnostics of Section 4.4 in this example.

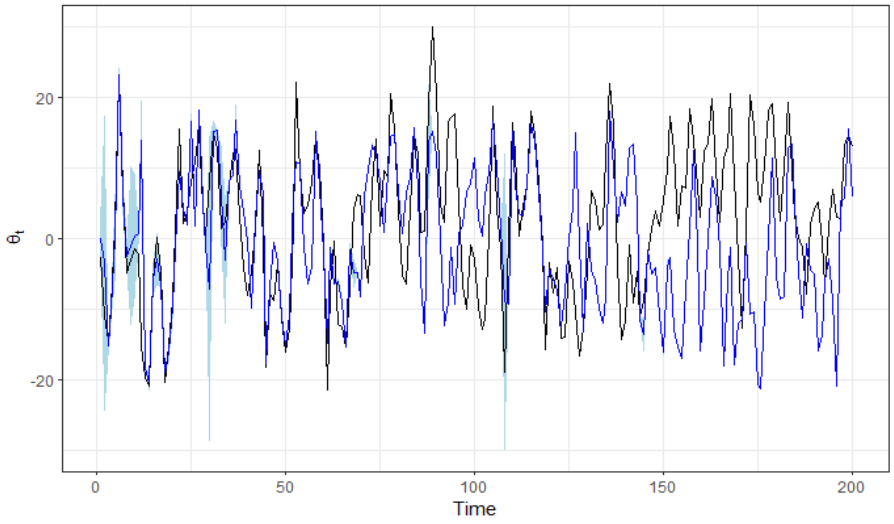

FIGURE 4.7
Sequential importance sampling: Example 1, [14].

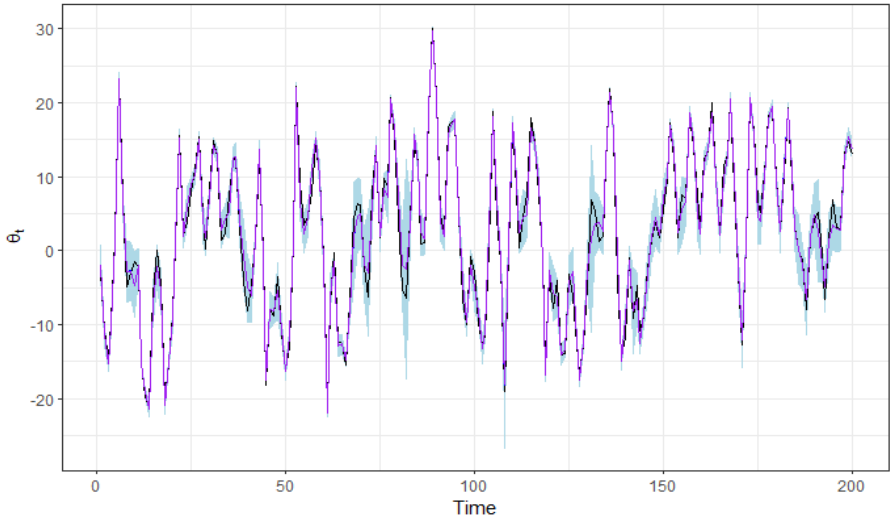

FIGURE 4.8
Particle (Bootstrap) filtering: Example 1, [14].

- 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 Ex-

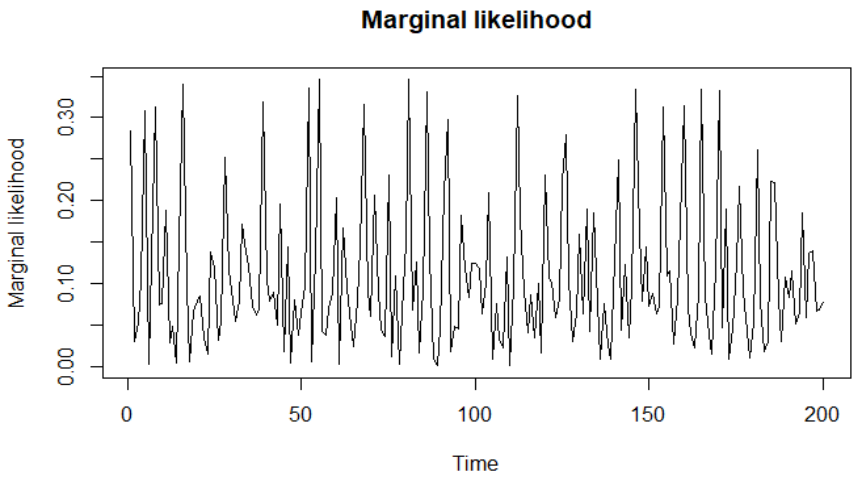

FIGURE 4.9
Marginal likelihood: Example 1, [14].

ercise 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.

Answer

We initially run the Metropolis-Hastings (M-H) algorithm for 100,000 iterations and obtain an effective sample size of 13,606. The naive and time-series standard errors are 0.00019 and 0.00052, respectively, indicating correlation—a feature also observed in the autocorrelation plot (not shown). However, the trace plot appears satisfactory. The Geweke test statistic is 1.97, marginally rejecting the null hypothesis that the means of the first 10% and the last 50% of the posterior draws are equal. Additionally, the dependence factor is 6.1, which exceeds the rule-of-thumb threshold of 5. The posterior draws pass both stages of Heidelberger and Welch's convergence diagnostic.

Given the autocorrelation issues, we set a burn-in period of 20,000 iterations and apply a thinning parameter of 10. Figures 4.10 and 4.11 display the trace and autocorrelation plots of the posterior draws. We observe that the draws appear stationary around their mean, and the autocorrelation decreases rapidly to zero.

The naive and time-series standard errors are 0.00068 and 0.00072, respectively. The similarity between these values indicates a low level of auto-

correlation, consistent with the results in Figure 4.11. The effective sample size of the posterior draws is 7,066, while the total number of posterior draws is 8,000.

The Geweke test statistic is 0.23, providing 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 close to 1, indicating minimal dependence. The Heidelberger and Welch test does not reject the null hypothesis of stationarity for the posterior draws and confirms that the mean is both accurate and stable.

Then, we implement the proposal by [12] to assess the reliability of the posterior simulator. We use as test functions the first moments of p and p^2. We fail to reject the null hypothesis of equal means across the two 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 $\alpha_0 = 1.22$ and $\beta_0 = 2.57$. In contrast, for the *successive-conditional simulator*, we use prior parameters $\alpha_0 = 2.57$ and $\beta_0 = 1.22$. In this case, we reject the null hypothesis in the two test functions, suggesting that the test performs well in this example.

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)
  a <- dbeta(pc, an, bn)/dbeta(p[s-1], an, bn)
  U <- runif(1)
  if(U <= a){ p[s] <- pc; accept[s] <- 1
  }else{p[s] <- p[s-1]; accept[s] <- 0}
}
library(coda); library(latex2exp)
thetaPost <- mcmc(p)
plot(thetaPost, density = FALSE, main = "Trace plot", ylab =
     "Proportion")
autocorr.plot(thetaPost, main = "Autocorrelation plot")
summary(thetaPost); effectiveSize(thetaPost)
geweke.diag(thetaPost, frac1 = 0.1, frac2 = 0.5)
raftery.diag(thetaPost, q = 0.025, r = 0.005, s = 0.95)
heidel.diag(thetaPost, eps = 0.1, pvalue = 0.05)
burnin <- 20000; thin <- 10; keep <- seq(burnin, S, thin)
thetaPostNew <- mcmc(p[keep])
plot(thetaPostNew, density = FALSE, main = "Trace plot",
    ylab = "Proportion")
autocorr.plot(thetaPostNew, main = "Autocorrelation plot")
summary(thetaPostNew); effectiveSize(thetaPostNew)
geweke.diag(thetaPostNew, frac1 = 0.1, frac2 = 0.5)
raftery.diag(thetaPostNew, q = 0.025, r = 0.005, s = 0.95)
heidel.diag(thetaPostNew, eps = 0.1, pvalue = 0.05)
# Marginal-conditional simulator
a0 <- 1.22; b0 <- 2.57; ThetaPrior <- rbeta(S,a0,b0)
yPrior <- rbinom(S, 1, prob = ThetaPrior)
parsmcmc1 <- coda::mcmc(cbind(ThetaPrior, ThetaPrior^2))
Summ1 <- summary(parsmcmc1)
# Successive-conditional simulator
N <- 52
SucConSim <- function(a0, b0, par){
  y <- rbinom(N, 1, prob = par)
  an <- a0 + sum(y); bn <- b0 + N - sum(y); pc <- runif(1)
  a <- dbeta(pc, an, bn)/dbeta(par, an, bn); U <- runif(1)
  if(U <= a){par <- pc}else{par <- par}
  return(par)
}
a0 <- 1.22; b0 <- 2.57 # a0 <- 2.57; b0 <- 1.22
par2 <- rbeta(1, a0, b0); pars2 <- rep(NA, S); pars2[1] <-
    par2
for(s in 2:S){
  Res <- SucConSim(a0 = a0, b0 = b0, par = pars2[s-1])
  pars2[s] <- Res
}
parsmcmc2 <- coda::mcmc(cbind(pars2, pars2^2))
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)
```

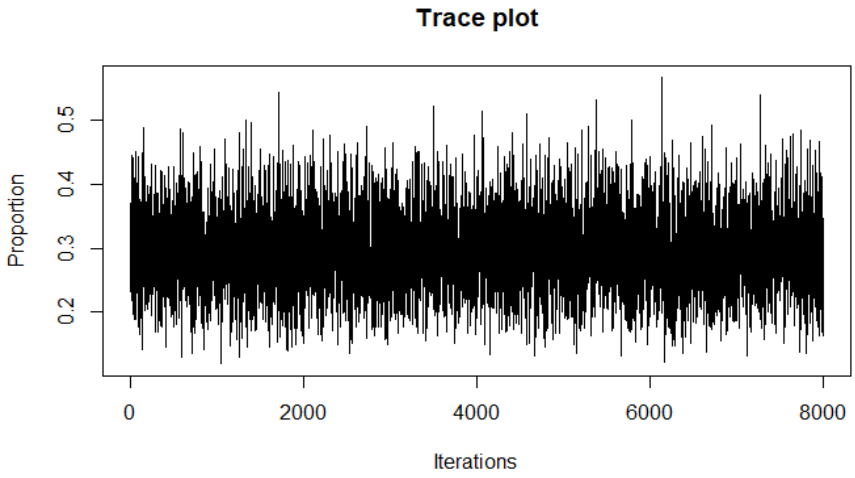

FIGURE 4.10
Trace plot of posterior draws: Proportion of students that sleep at least 6 hours.

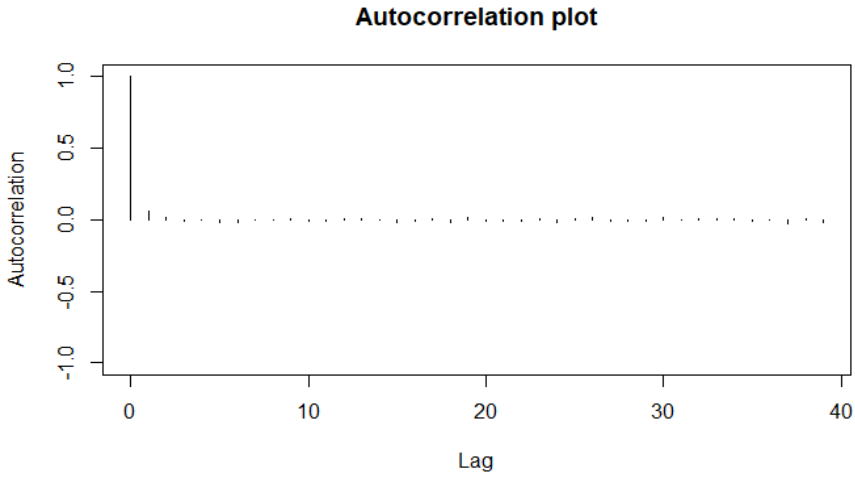

FIGURE 4.11
Autocorrelation plot of posterior draws: Proportion of students that sleep at least 6 hours.

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 that requires no programming skills, using a simple drag-and-drop design. The GUI is implemented as an interactive web application built with *shiny* [6] and integrates packages such as *MCMCpack* [27] and *bayesm* [38] within the **R** software environment [31]. It is primarily designed for teaching and applied purposes at the introductory level. In the subsequent chapters of the second part of this book, we present several applications that illustrate 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 lack of such skills is often a significant impediment to the widespread adoption of the Bayesian framework [47, 21].

Several other graphical user interfaces are available for Bayesian regression analysis. *ShinyStan* [44] is a highly flexible and open-source program; however, it requires users to have some programming skills, as it is based on the *Stan* software for Bayesian data analysis [4]. *BugsXLA* [47] is also open source but less flexible, though it does not require any programming experience. *Bayesian Regression: Nonparametric and Parametric Models* [21] 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* [6] and various libraries in **R** [30]. The specific libraries and commands used in our GUI are listed in Table A.1. It includes ten univariate models, four multivariate models, four time-series models, three hierarchical longitudinal models, and seven Bayesian model averaging frameworks. In addition, it provides basic

summaries and diagnostics of posterior chains, along with visualizations such as trace plots, autocorrelation plots, and density plots.

In terms of flexibility and functionality, our GUI lies between *ShinyStan* and *BugsXLA*: it does not require programming skills but is not as advanced as the software described in [21]. However, unlike the latter, 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 A.2 and A.3 for details). The **DataSim** folder contains files used to simulate various processes, providing access to population parameters. These files serve as valuable pedagogical tools for illustrating the statistical properties of the inferential frameworks available in our GUI. The **DataApp** folder includes the datasets used throughout this book, which users can adopt as templates when structuring their own data.

There are four ways to install our GUI. The simplest method, requiring only the installation of **R** and, optionally, an **R** code editor, is to install *shiny*, and then 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 manually rather than copying and pasting it, as quotation marks can sometimes cause issues.

The second option is to visit

https://andres-ramirez-hassan.shinyapps.io/BSTApp/

Please note: the free Posit Cloud tier sometimes runs out of memory, which can cause the app to stop. Sorry for the inconvenience.

The third option is to visit **https://fly-besmarter.fly.dev/**. As with Posit Cloud, occasional memory limits on the free tier may affect performance.

The fourth approach, and our recommendation, is using a **Docker** image by running:

1. docker pull aramir21/besmartergui:latest
2. docker run –rm -p 3838:3838 aramir21/besmartergui

[1] Bayesian Econometrics: Simulations, Models, and Applications to Research, Teaching, and Encoding with Responsibility.

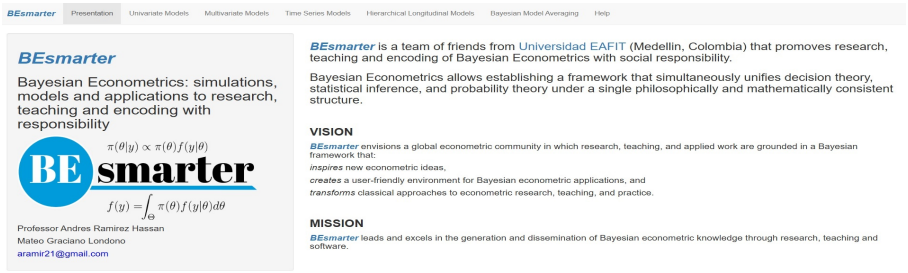

FIGURE 5.1
Display of graphical user interface.

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/*.

After using any of the four 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.

The right-hand panel then 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 type of separator used in the input file —comma, semicolon, or tab- (see 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 part of the book for technical details).

Next, users must specify the model equation. This can be done using the formula builder, where they select the dependent and independent variables

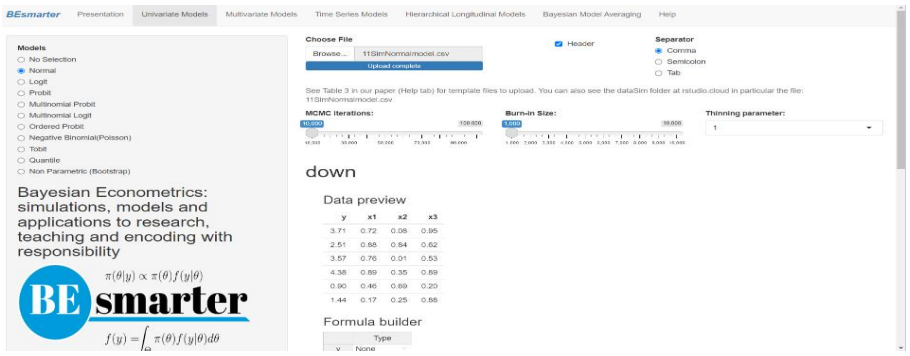

FIGURE 5.2
Univariate models: Specification.

and then click on the *Build Formula* tab. The equation is displayed in the *Main Equation* window, formatted according to **R** syntax, for example, $y \sim x1 + x2 + x3$. Users can modify this expression as needed to include higher-order terms, interaction effects, or other transformations. These modifications must follow 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 *noninformative* 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 estimation (see Figure 5.3). Once the process is complete, the GUI displays summary statistics and convergence diagnostics. In addition, widgets allow users to download the posterior chains (as *csv* files) and graphs (as *pdf* or *eps* files). Note that in the results —summary tables, posterior chains, and graphs— the coefficients are ordered such that location parameters appear 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' price), and then non-alternative-specific regressors (e.g., income). The formula builder allows users to specify the dependent variable as well as both types of independent

[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.

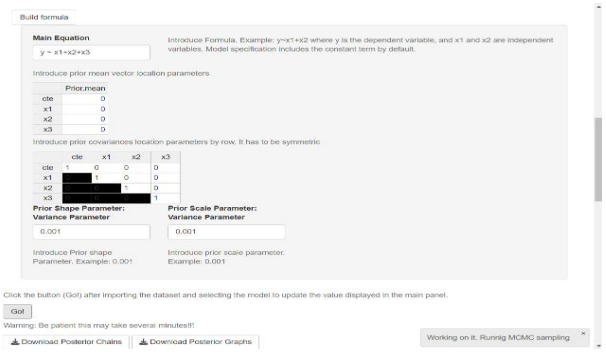

FIGURE 5.3
Univariate models: Priors.

variables (see technical details in the next chapter). Users must also define the base category, the number of alternatives (which is also required for ordered probit models), the number of alternative-specific regressors, and the number of non-alternative-specific regressors (see Figure 5.4).

For multinomial logit models, users can additionally specify a tuning parameter corresponding to the degrees of freedom of the Metropolis–Hastings algorithm (see technical details in the next chapter). This tuning option is available in our GUI given that estimation relies on the Metropolis–Hastings algorithm.

In the results of these models, the 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, burn-in 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.

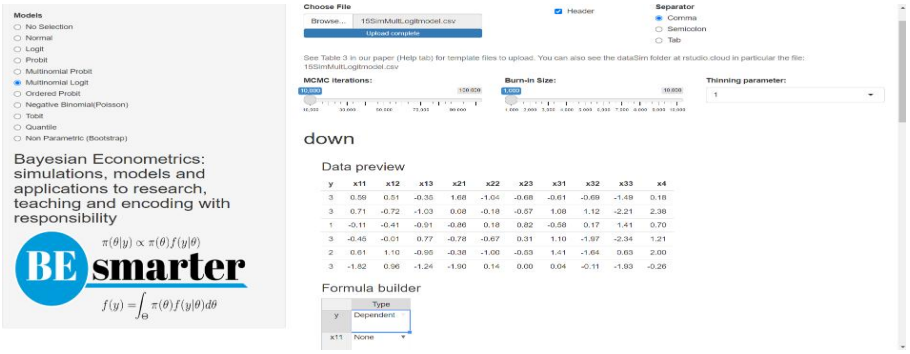

FIGURE 5.4
Univariate models: Multinomial.

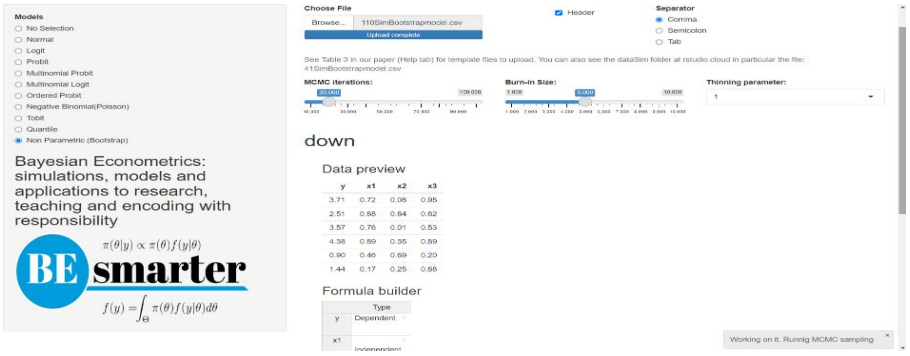

FIGURE 5.5
Univariate models: Bootstrap.

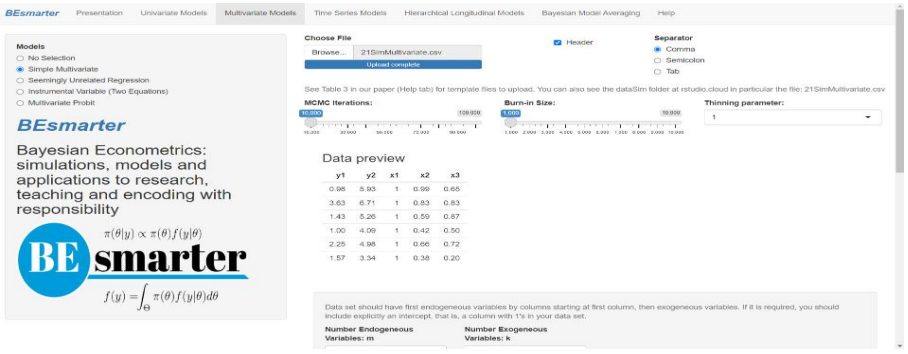

FIGURE 5.6
Multivariate models: Simple multivariate.

5.3 Multivariate models

After deploying our GUI (see Figure 5.1), the user should select *Multivariate Models* from the top panel. Figure 5.6 will then appear, displaying a radio button on the left-hand side that lists the specific models within this category.

Figure 5.6 illustrates the setup for multivariate regression models. The input file should first include the dependent variables, followed by the regressors. If each equation includes an intercept, a column of ones must be added after the dependent variables in the input file. Once the file is uploaded, users can preview the data.

The user must then 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 include the dependent variables, followed by the regressors for each equation, including the intercept (a column of ones) if applicable. Users must specify the number of dependent variables (equations), the total number of regressors (i.e., the sum of all regressors across equations), and the number of regressors per equation (including the intercept if relevant). Users may also define the values of the hyperparameters if prior information is available.

For both simple multivariate and seemingly unrelated regression models, the results first display the posterior estimates of the location parameters by equation, followed by the posterior covariance matrix.

In the instrumental variable setting, users must specify both the main equation and the instrumental equation. Intercepts are included by default. The first variable on the right-hand side of the main equation must correspond to the endogenous regressor. In the instrumental equation, this endogenous variable serves as the dependent variable and is modeled as a function of

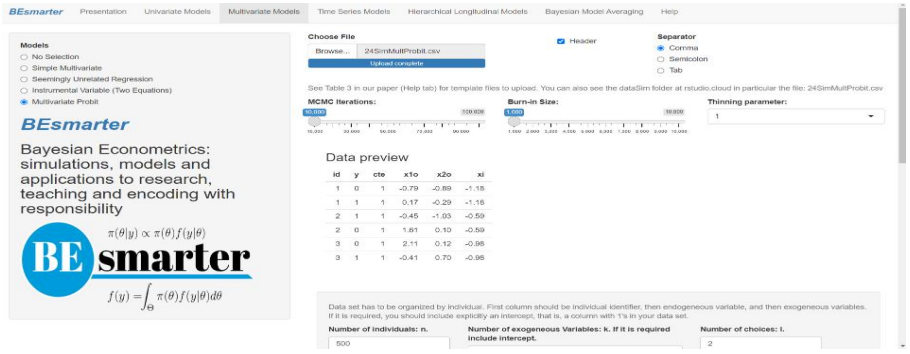

FIGURE 5.7
Multivariate models: Multivariate probit.

the instruments. Users may also specify hyperparameter values if prior information is available. The input file should contain the dependent variable, the endogenous regressor, the instruments, and the exogenous regressors. The results first report the posterior estimates of the endogenous regressor, followed by the location parameters of the auxiliary (instrumental) regression, 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, if there are three choices, each unit should appear three times in the dataset. The first column must contain a unique identifier for each unit, using ordered integers. The next column should contain the dependent variable, represented as a single vector of 0s and 1s, followed by the regressors, which must include a column of ones for the intercepts. Users must specify the number of units, the number of regressors, and the number of choices (see Figure 5.7). The results first display the posterior estimates of the location parameters by equation, followed by the posterior covariance matrix.

5.4 Time series models

After our GUI is deployed (see Figure 5.1), 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

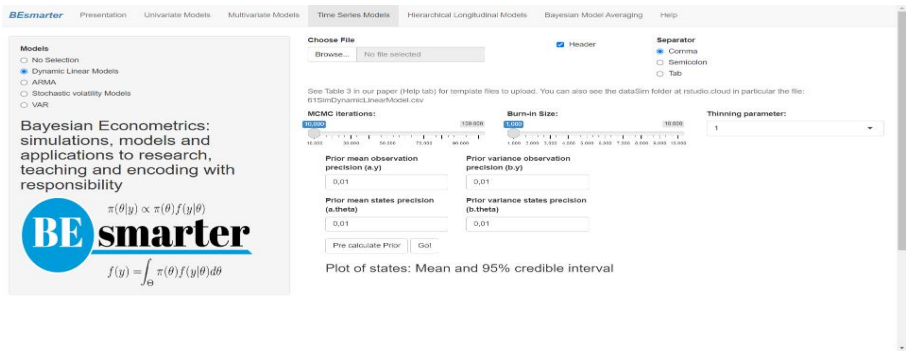

FIGURE 5.8
Time series models.

must be a *csv* file with headers in the first row. The files for DLMs and SVMs 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 A.2 for details) and **DataApp** (see Table A.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

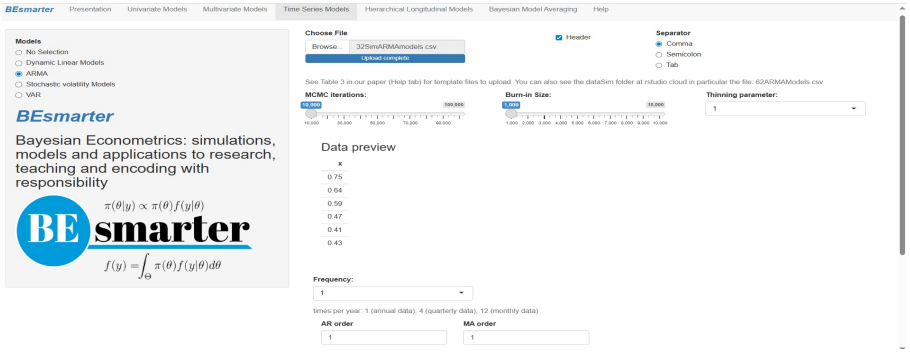

FIGURE 5.9
Time series models: ARMA specification

chains for the observation and state variances, and convergence diagnostics. 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.

FIGURE 5.10
Time series models: Stochastic volatility results.

To estimate VAR models, users should set the number of lags, the impulse 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 **??**.

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 A.2 for details) and

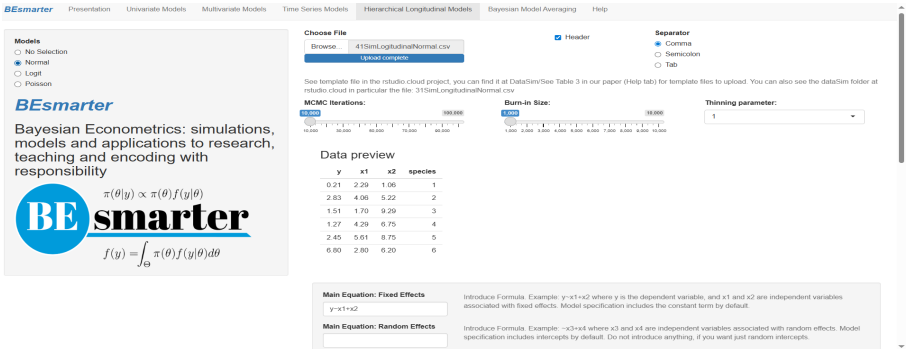

FIGURE 5.11
Hierarchical longitudinal models: Specification.

DataApp (see Table A.3 for details) in our *GitHub* repository. When the dataset is uploaded, users will have a preview of it.

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 (*species* in Figure 5.11). 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, and dynamic BMA (see Figure 5.12). 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 associ-

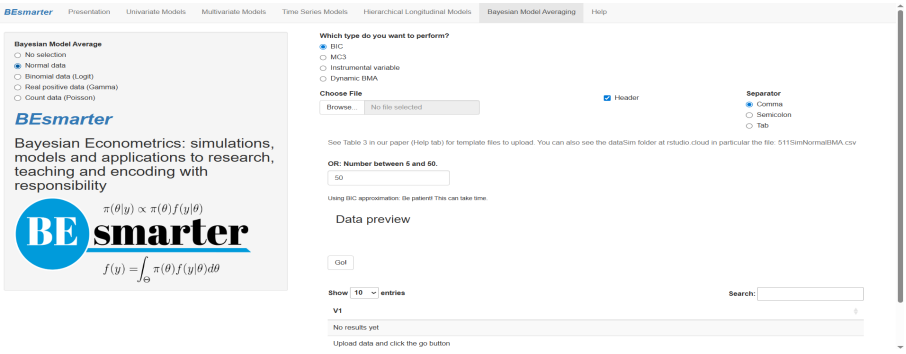

FIGURE 5.12
Bayesian model averaging: Specification and results.

ated with each regressor. The BIC framework also displays the most relevant models with their posterior model probabilities (PMP). Users can download 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 values 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, $\beta_l, l = 1, 2, \ldots, dim\{\boldsymbol{\beta}\}$, 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

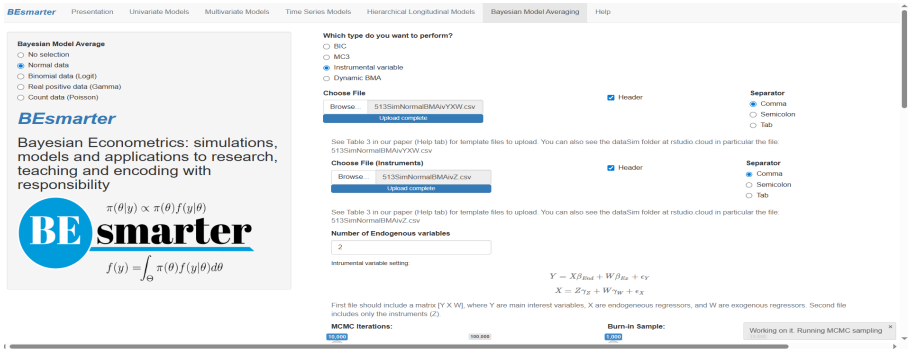

FIGURE 5.13
Bayesian model averaging: Instrumental variable specification.

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 A.2 for details) and **DataApp** (see Table A.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, users can find links to both the online *HTML* version of the book,[4] and the *PDF* version.[5] Users are also welcome to contact me at *aramir21@gmail.com* with 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. 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.

[4] https://bookdown.org/aramir21/IntroductionBayesianEconometricsGuidedTour/
[5] https://drive.google.com/file/d/1_IBKe7vS5a2XLnvg74T_UNESRoKNDUUt/edit

6

Univariate models

Solutions of Exercises

1. Derive 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 $\sigma^2 \sim IG(\alpha_0/2, \delta_0/2)$.

   Answer

   The joint posterior distribution of the parameters is

$$
\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, \boldsymbol{X}) &\propto p(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{X})\pi(\boldsymbol{\beta})\pi(\sigma^2) \\
&\propto (\sigma^2)^{-\frac{N}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right\} \\
&\times \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)\right\} \\
&\times \frac{(\delta_0/2)^{(\alpha_0/2)}}{\Gamma(\alpha_0/2)} \frac{1}{(\sigma^2)^{(\alpha_0/2+1)}} \exp\left\{-\frac{\delta_0}{2\sigma^2}\right\} \\
&\propto \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)\right\} \\
&\times \frac{1}{(\sigma^2)^{(\alpha_0+N)/2+1}} \exp\left\{-\frac{\delta_0 + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})}{2\sigma^2}\right\} \\
&= \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)\right\} \times \underbrace{\frac{1}{(\sigma^2)^{(\frac{\alpha_n}{2}+1)}} \exp\left\{-\frac{\delta_n}{2\sigma^2}\right\}}_{1}.
\end{aligned}
$$

   Observe that (1) is the kernel of an inverse-gamma density function. Thus, $\sigma^2|\boldsymbol{\beta}, \boldsymbol{y}, \boldsymbol{X} \sim IG(\alpha_n/2, \delta_n/2)$, where $\alpha_n = \alpha_0 + N$ and $\delta_n = \delta_0 + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$.

Let's see the posterior distribution of $\boldsymbol{\beta}$,

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

where $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$.
Adding and subtracting $\boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$ where

$$\begin{aligned}
\boldsymbol{B}_n &= (\boldsymbol{B}_0^{-1} + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{X})^{-1} \\
\boldsymbol{\beta}_n &= \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}}) = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{y}),
\end{aligned}$$

and completing the square

$$\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, \boldsymbol{X}) &\propto \exp\left\{-\frac{1}{2}[\boldsymbol{\beta}^\top(\boldsymbol{B}_0^{-1} + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{X})\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\boldsymbol{B}_n^{-1}\boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top\boldsymbol{X}\hat{\boldsymbol{\beta}})\right. \\
&\left. + \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n - \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n]\right\} \times \frac{1}{(\sigma^2)^{(\alpha_0+N)/2+1}} \exp\left\{-\frac{\delta_0 + \boldsymbol{y}^\top\boldsymbol{y}}{2\sigma^2}\right\} \\
&= \exp\left\{-\frac{1}{2}[\boldsymbol{\beta}^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n]\right\} \\
&\times \frac{1}{(\sigma^2)^{(\alpha_0+N)/2+1}} \exp\left\{-\frac{\delta_0 + \boldsymbol{y}^\top\boldsymbol{y} - \sigma^2\boldsymbol{\beta}_n^\top\boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n}{2\sigma^2}\right\} \\
&= \underbrace{\exp\left\{-\frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)^\top\boldsymbol{B}_n^{-1}(\boldsymbol{\beta}-\boldsymbol{\beta}_n)\right\}}_{1} \\
&\times (\sigma^2)^{-(\frac{\alpha_n}{2}+1)} \exp\left\{-\frac{\delta^*}{2\sigma^2}\right\},
\end{aligned}$$

where $\delta^* = \delta_0 + \boldsymbol{y}^\top \boldsymbol{y} + \sigma^2 \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1} \boldsymbol{\beta}_n$ does not dependent on $\boldsymbol{\beta}$.

We can see that (1) is the kernel of a multivariate normal distribution with mean equal to $\boldsymbol{\beta}_n$ and covariance matrix $\boldsymbol{B}_n$, that is, $\boldsymbol{\beta}|\sigma^2, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n)$.

We see that the posterior distributions are from the same family as the prior distributions.

2. Consider the model $y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2/\tau_i)$ (Gaussian linear model with heteroskedasticity) and 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 $\boldsymbol{\tau} = [\tau_1 \ \dots \ \tau_N]^\top$, $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sigma^{-2} \boldsymbol{X}^\top \Psi \boldsymbol{X})^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}\boldsymbol{X}^\top \Psi \boldsymbol{y})$, $\alpha_n = \alpha_0 + N$, $\delta_n = \delta_0 + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top \Psi (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$, $v_{1n} = v + 1$, $v_{2in} = v + \sigma^{-2}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2$, and $\Psi = \operatorname{diag}(\tau_i)$.

   Answer

   The joint posterior distribution of the parameters is

$$
\begin{aligned}
\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\tau} | \boldsymbol{y}, \boldsymbol{X}) &\propto p(\boldsymbol{y} | \boldsymbol{\beta}, \sigma^2, \boldsymbol{\tau}, \boldsymbol{X}) \pi(\boldsymbol{\beta}) \pi(\sigma^2) \prod_{i=1}^N \pi(\tau_i) \\
&\propto \left(\prod_{i=1}^N \tau_i^{1/2} \right) (\sigma^2)^{-\frac{N}{2}} \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top \Psi (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \right\} \\
&\times \exp\left\{ -\frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_0) \right\} \\
&\times \frac{1}{(\sigma^2)^{(\alpha_0/2+1)}} \exp\left\{ -\frac{\delta_0}{2\sigma^2} \right\} \prod_{i=1}^N \tau_i^{v/2-1} \exp\left\{ -v\tau_i/2 \right\}.
\end{aligned}
$$

   Thus, the conditional posterior distribution of $\sigma^2 | \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}$ is given by

$$
\pi(\sigma^2 | \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}) \propto (\sigma^2)^{-\frac{N+\alpha_0}{2}} \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top \Psi (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \delta_0 \right\}.
$$

   This is the kernel of an inverse-gamma distribution with shape parameter $\alpha_n/2$ and rate parameter $\delta_n/2$.

   The conditional posterior distribution of $\tau_i | \boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X}$ is

$$
\pi(\tau_i | \boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X}) \propto \tau_i^{(v+1)/2-1} \exp\left\{ -\frac{\tau_i}{2} [\sigma^{-2}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + v] \right\}.
$$

This is the kernel of a gamma distribution with parameter $(v+1)/2$ and $(\sigma^{-2}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + v)/2$.

The conditional posterior distribution of $\boldsymbol{\beta}|\sigma^2, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}$ is given by

$$\pi(\boldsymbol{\beta}|\sigma^2, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}) \propto \exp\left\{-\frac{1}{2}[\sigma^{-2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top \Psi(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)]\right\}.$$

Following same steps as in the previous exercise we get $\boldsymbol{\beta}|\sigma^2, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n)$.

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 for the market value of soccer players in Europe, setting $v = 5$ and using the same hyperparameters as in the homoscedastic case. Is there any meaningful difference in the coefficient associated with the national team compared to the homoscedastic case?

   Answer

R. code. The value of soccer players, programming our Gibbs sampler (heteroskedastic case)

```
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/1
    ValueFootballPlayers.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
a0 <- 0.001
b0 <- rep(0, k)
c0 <- 1000
B0 <- c0*diag(k)
B0i <- solve(B0)
v <- 5
# MCMC parameters
mcmc <- 5000
burnin <- 5000
tot <- mcmc + burnin
thin <- 1
# Posterior distributions programming the Gibbs sampling
# Auxiliary parameters
an <- a0 + N
v1n <- v + 1
# Gibbs sampling functions
PostSig2 <- function(Beta, tau){
  dn <- d0 + t(y - X%*%Beta)%*%diag(tau)%*%(y - X%*%Beta)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostBeta <- function(sig2, tau){
  Bn <- solve(B0i + sig2^(-1)*t(X)%*%diag(tau)%*%X)
  bn <- Bn%*%(B0i%*%b0 + sig2^(-1)*t(X)%*%diag(tau)%*%y)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostTau <- function(sig2, Beta, i){
  v2n <- v + sig2^(-1)*(y[i]-X[i,]%*%Beta)^2
  taui <- rgamma(1, v1n/2, v2n/2)
  return(taui)
}
```

R. code. The value of soccer players, programming our Gibbs sampler (heteroskedastic case)

```
PostBetas <- matrix(0, mcmc+burnin, k)
PostSigma2 <- rep(0, mcmc+burnin)
Beta <- rep(0, k)
tau <- rep(1, N)
# create progress bar in case that you want to see
    iterations progress
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  sig2 <- PostSig2(Beta = Beta, tau = tau)
  PostSigma2[s] <- sig2
  Beta <- PostBeta(sig2 = sig2, tau = tau)
  PostBetas[s,] <- Beta
  tau <- sapply(1:N, function(i){PostTau(sig2 = sig2, Beta =
     Beta, i)})
  setTxtProgressBar(pb, s)
}
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))
PosteriorSigma2 <- PostSigma2[keep]
summary(coda::mcmc(PosteriorSigma2))
summary(coda::mcmc(exp(PosteriorBetas[,5])-1))
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
1.256091       0.254974       0.003606       0.004579
2. Quantiles for each variable:
2.5%    25%    50%    75%  97.5%
0.8078 1.0746 1.2367 1.4189 1.7959
```

We see in this application that the value of a top soccer player in Europe increases 124% ($\exp(0.80) - 1$)) on average when he has played in the national team, the credible interval at 95% is (81%, 180%). These values are not very different from the application assuming homoscedasticity in the book.

4. **Example: Determinants of hospitalization continues.**

Program a Gibbs sampling algorithm for the application on the determinants of hospitalization

Answer

R. code. Determinants of hospitalization, programming our Gibbs sampler

```
set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/2HealthMed.csv", sep = ",", header = TRUE, quote = "")
attach(Data)
str(Data)
y <- Hosp # Dependent variables
X <- cbind(1, SHI, Female, Age, Age2, Est2, Est3, Fair, Good, Excellent) # Regressors
K <- dim(X)[2]
N <- dim(X)[1]
# Hyperparameters
b0 <- rep(0, K) # Prio mean
B0 <- diag(K) # Prior covariance
B0i <- solve(B0)
mcmc <- 1000; burnin <- 500; thin <- 2; tot <- mcmc + burnin; keep <- seq(burnin, tot, thin)
# Posterior distributions programming the Gibbs sampling
# Auxiliary parameters
XtX <- t(X)%*%X
# Gibbs sampling functions
PostBeta <- function(Yl){
  Bn <- solve(B0i + XtX)
  bn <- Bn%*%(B0i%*%b0 + t(X)%*%Yl)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostYl <- function(Beta, i){
  Ylmean <- X[i,]%*%Beta
  if(y[i] == 1){
    Yli <- truncnorm::rtruncnorm(1, a = 0, b = Inf, mean = Ylmean, sd = 1)
  }else{
    Yli <- truncnorm::rtruncnorm(1, a = -Inf, b = 0, mean = Ylmean, sd = 1)
  }
  return(Yli)
}
PostBetas <- matrix(0, mcmc+burnin, K)
Beta <- rep(0, K)
# create progress bar in case that you want to see iterations progress
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  Yl <- sapply(1:N, function(i){PostYl(Beta = Beta, i)})
  Beta <- PostBeta(Yl = Yl)
  PostBetas[s,] <- Beta
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorBetas <- PostBetas[keep,]
colnames(PosteriorBetas) <- c("Intercept", "SHI", "Female", "Age", "Age2", "Est2", "Est3", "Fair", "Good", "Excellent")
summary(coda::mcmc(PosteriorBetas))
```

5. **Choice of fishing mode continues.**

   - Run Algorithm A9 to report the results of the Geweke [11], Raftery [33], and Heidelberger [16] tests using our GUI.
   - Use the command *rmnpGibbs* to replicate the fishing mode choice example.

   Answer

R. code. Fishing choice mode, results our GUI

```
GewekeTestLocationCoef
Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5
cte_1  cte_2  cte_3 NAS_1_1 NAS_1_2 NAS_1_3    AS_1   AS_2
-1.821 -0.714  0.792  2.275 -3.944 -2.071  1.627 -2.729

RafteryTestLocationCoef
Quantile (q) = 0.5
Accuracy (r) = +/- 0.025
Probability (s) = 0.95
Burn-in  Total   Lower bound  Dependence
(M)      (N)     (Nmin)       factor (I)
cte_1    780     365690  1537          238.0
cte_2    360     193950  1537          126.0
cte_3    660     340120  1537          221.0
NAS_1_1  120     70320   1537           45.8
NAS_1_2  475     243960  1537          159.0
NAS_1_3  440     248930  1537          162.0
AS_1     3010    1438135 1537          936.0
AS_2     550     297770  1537          194.0

HeidelTestLocationCoef
Stationarity start     p-value
test         iteration
cte_1    passed       6001      6.54e-01
cte_2    failed         NA      3.72e-02
cte_3    failed         NA      4.99e-02
NAS_1_1  failed         NA      4.77e-07
NAS_1_2  failed         NA      1.82e-05
NAS_1_3  failed         NA      1.19e-04
AS_1     passed       2001      3.71e-01
AS_2     passed       8001      4.48e-01
Halfwidth Mean     Halfwidth
test
cte_1    passed    -0.34236 0.017025
cte_2    <NA>           NA       NA
cte_3    <NA>           NA       NA
NAS_1_1  <NA>           NA       NA
NAS_1_2  <NA>           NA       NA
NAS_1_3  <NA>           NA       NA
AS_1     passed    -0.00708 0.000306
AS_2     passed     0.27982 0.009994
```

R. code. Fishing choice mode, library bayesm

```
remove(list = ls()); set.seed(12345)
Data <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/3Fishing.csv"
    , sep = ",", header = TRUE, quote = "")
attach(Data); str(Data)
p <- 4; na <- 2; nd <- 1; N <- dim(Data)[1]
Xa <- Data[,2:9]
Xd <- matrix(income, N, 1)
X <- bayesm::createX(p = p, na = na, nd = nd, Xa = Xa, Xd =
    Xd, INT = TRUE, base = p, DIFF = TRUE)
df <- list(y = mode, X = X, p = 4)
# 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-1)
Prior <- list(betabar = b0, A = B0i, nu = a0, V = Psi0)
# MCMC parameters
mcmc <- 100000; thin <- 5
Mcmc <- list(R = mcmc, keep = thin)
Results <- bayesm::rmnpGibbs(Data = df, Prior = Prior, Mcmc
    = Mcmc)
betatilde <- Results$betadraw / sqrt(Results$sigmadraw[,1])
attributes(betatilde)$class <- "bayesm.mat"
summary(coda::mcmc(betatilde))
Quantiles for each variable:
          2.5%         25%         50%         75%       97.5%
var1 -6.371e-01 -4.670e-01 -3.834e-01 -3.028e-01 -1.422e-01
var2 -2.149e-01 -6.681e-02 -7.870e-03  4.414e-02  1.493e-01
var3 -8.873e-01 -6.071e-01 -4.982e-01 -4.040e-01 -2.446e-01
var4  1.859e-05  4.577e-05  5.952e-05  7.326e-05  9.861e-05
var5 -2.589e-05 -3.050e-06  8.212e-06  2.089e-05  5.330e-05
var6  4.549e-05  7.451e-05  9.144e-05  1.098e-04  1.555e-04
var7 -9.425e-03 -7.612e-03 -6.586e-03 -5.573e-03 -3.136e-03
var8  1.214e-01  2.068e-01  2.592e-01  3.154e-01  4.477e-01
sigmadraw <-  Results$sigmadraw / Results$sigmadraw[,1]
attributes(sigmadraw)$class = "bayesm.var"
summary(coda::mcmc(sigmadraw))
```

6. **Simulation exercise: Multinomial logit model continues.**

   Perform inference in the simulated multinomial logit model using the command *rmnlIndepMetrop* from the *bayesm* package in **R**, and compare the results with those obtained from our GUI.

   Answer

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))
table(y)
L <- length(table(y))
dat <-data.frame(mode,X1[,1],X2[,1],X3[,1],X1[,2],X2[,2],X3[,2],X1[,3],X2[,3],X3[,3],X4)
colnames(dat) <- c("mode","V1.1","V1.2","V1.3","V2.1","V2.2","V2.3","V3.1","V3.2","V3.3","V4")
attach(dat)
LongData <- mlogit::mlogit.data(dat, shape = "wide", varying=2:10, choice = "mode")
Xa <- cbind(LongData$V1, LongData$V2, LongData$V3)
Xa <- cbind(X1[,1],X2[,1],X3[,1],X1[,2],X2[,2],X3[,2],X1[,3],X2[,3],X3[,3])
na <- 3
Xd <- X4
X <- bayesm::createX(p = L, na = na, nd = 1, Xa = Xa, Xd = Xd, base = L)
DataMlogit <- list(y=y, X = X, p = L)
# MCMC parameters
mcmc <- 11000+1
thin <- 5
df <- 6
mcmcpar <- list(R = mcmc, keep = 5, nu = df)
PostBeta <- bayesm::rmnlIndepMetrop(Data = DataMlogit, Mcmc = mcmcpar)
summary(PostBeta[["betadraw"]])
```

7. **Simulation of the ordered probit model.**

   Simulate an ordered probit model where the first regressor follows $N(6, 5)$ and the second follows $G(1, 1)$. The location vector is $\boldsymbol{\beta} = [0.5 \ -0.25 \ 0.5]^\top$, and the cutoffs are $\boldsymbol{\alpha} = [0 \ 1 \ 2.5]^\top$. Program a Metropolis-within-Gibbs sampling algorithm from scratch to perform inference in this simulation.

   Answer

R. code. Simulation of the ordered probit model

```
rm(list = ls()); set.seed(010101); N <- 1000
x1 <- rnorm(N, 6, 5); x2 <- rgamma(N, shape = 1, scale = 1)
X <- cbind(1, x1, x2)
beta <- c(0.5, -0.25, 0.5); cutoffs <- c(0, 1, 2.5)
e <- rnorm(N,0,1)
y_latent <- X%*%beta + e; y <- rep(0,N)
for (i in 1:N) {
  if (y_latent[i] < cutoffs[1]){
    y[i] <- 0}else{
    if (y_latent[i] >= cutoffs[1] & y_latent[i] < cutoffs
    [2]) {
      y[i] <- 1
    }else{
      if (y_latent[i] >= cutoffs[2] & y_latent[i] < cutoffs
    [3]) {
        y[i] <- 2
      }else{y[i] <- 3
      }
    }
  }
}
# Likelihood function
LogLikOP <- function(param){
  beta_g <- param[1:ncol(X)]
  delta <- param[(ncol(X)+1):(ncol(X) + dplyr::n_distinct(y)
     - 1)]
  Xbeta <- X%*%beta_g
  logLik <- 0
  for (i in 1:length(y)){
    if (y[i]==0){logLiki <- log(pnorm(-Xbeta[i]))
    }else if (y[i]==1){
      logLiki <- log(pnorm(exp(delta[1]) - Xbeta[i]) - pnorm
    (-Xbeta[i]))
    }else if (y[i]==2){
      logLiki <- log(pnorm(exp(delta[2]) + exp(delta[1]) -
    Xbeta[i]) - pnorm(exp(delta[1]) - Xbeta[i]))
    }else {logLiki <- log(1 - pnorm(exp(delta[2]) + exp(
    delta[1]) - Xbeta[i]))
    }
    logLik <- logLik + logLiki
  }
  return(-logLik)
}
# ML Estimation
param0 <- rep(0, ncol(X) + n_distinct(y)-2)
mle <- optim(param0, LogLikOP, hessian = T, method = "BFGS")
mle$par
exp(mle$par[length(beta)+1])
exp(mle$par[length(beta)+1])+exp(mle$par[length(beta)+2])
CovarML <- solve(mle$hessian)
```

R. code. Simulation of the ordered probit model

```r
# M-H within Gibbs
mhop <- function(param0, G){
  betasamples <- matrix(c(0), nrow = G, ncol = ncol(X))
  betasamples[1,] <- param0[1:ncol(X)]
  tau <- matrix(c(0), nrow = G, ncol = dplyr::n_distinct(y)
    - 2)
  tau[1,] <- param0[(ncol(X)+1):(ncol(X) + dplyr::n_distinct
    (y) - 2)]
  yl <- rep(0,length(y)); ar <- rep(0,G); B1 <- solve(t(X)%*
    %X+solve(B0))
  pb <- txtProgressBar(min = 0, max = G, style = 3)
  for(g in 2:G){
    bg <- betasamples[g-1,]; tg <- tau[g-1,]
    #Random walk M-H for delta
    delta_prime <- tg + mvtnorm::rmvnorm(1, mean = rep(0,2),
     sigma = VarProp)
    alpha <- min(1,(mvtnorm::dmvnorm(delta_prime, mean = d0,
     sigma = D0)*exp(-LogLikOP(c(bg, delta_prime)) +
    LogLikOP(c(bg, tg))))/mvtnorm::dmvnorm(tg, mean = d0,
    sigma = D0))
    if(is.nan(alpha) | is.na(alpha)) {
      alpha <- 0
    }
    #Acceptance step
    u <- runif(1, min = 0, max = 1)
    if(u<=alpha){tau[g,] <- delta_prime; ar[g] <- 1
    }else{tau[g,] <- tg
    }
    #Generation of latent variables
    for (i in 1:length(y)){
      if (y[i]==0) {
        yl[i] <- EnvStats::rnormTrunc(1, mean = X[i,]%*%bg,
    sd = 1, max = 0)
      }else if(y[i]==1){
        yl[i] <- EnvStats::rnormTrunc(1, mean = X[i,]%*%bg,
    sd = 1, min = 0, max = exp(tau[g,1]))
      }else if(y[i]==2){
        yl[i] <- EnvStats::rnormTrunc(1, mean = X[i,]%*%bg,
    sd = 1, min = exp(tau[g,1]), max = exp(tau[g,2])+exp(tau
    [g,1]))
      }else{
        yl[i] <- EnvStats::rnormTrunc(1, mean = X[i,]%*%bg,
    sd = 1, min = exp(tau[g,2])+exp(tau[g,1]))
      }
    }
    #Gibbs sampling for beta
    if(sum(is.nan(yl))>0 | sum(is.na(yl))>0 | sum(yl)==Inf){
      betasamples[g,] <- betasamples[g-1,]
    }else{
      b1 <- B1%*%(t(X)%*%yl + solve(B0)%*%b0)
      betasamples[g,] <- mvrnorm(1, mu = b1, Sigma = B1)
    }
    setTxtProgressBar(pb, g)
  }
  close(pb)
  return(cbind(betasamples, tau, ar))
}
```

R. code. Simulation of the ordered probit model

```
#Hyperparameters
d0 <- rep(0,2)
D0 <- diag(2)*10000
b0 <- rep(0,ncol(X))
B0 <- diag(ncol(X))*10000
#Estimation
param0 <- rep(0, ncol(X) + dplyr::n_distinct(y)-1)
G <- 1000
tun <- 1
VarProp <- tun*solve(solve(CovarML[4:5, 4:5]) + solve(D0))
param_sample <- mhop(param0, G)
#Burn in
B <- round(0.2*G)
param_sample <- param_sample[(B+1):G,]
mcmc0 <- coda::mcmc(param_sample[, 1:(ncol(X) + dplyr::n_
    distinct(y) - 2)])
summary(mcmc0)
Iterations = 1:800
Thinning interval = 1
Number of chains = 1
Sample size per chain = 800
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean       SD  Naive SE Time-series SE
0.49120 0.08465 0.0029929       0.007140
-0.24919 0.01222 0.0004319       0.001269
0.49440 0.03942 0.0013937       0.002739
0.06716 0.06419 0.0022695       0.008479
0.41926 0.07414 0.0026212       0.009479
2. Quantiles for each variable:
2.5%       25%       50%     75%   97.5%
0.31947  0.43558  0.49710  0.5479  0.6496
-0.27180 -0.25740 -0.24948 -0.2408 -0.2238
0.42229  0.46706  0.49253  0.5189  0.5762
-0.06338  0.02328  0.06819  0.1104  0.1932
0.25857  0.37195  0.41742  0.4672  0.5558
summary(coda::mcmc(cbind(exp(param_sample[, 4]),exp(param_
    sample[, 4])+exp(param_sample[, 5]))))
Iterations = 1:800
Thinning interval = 1
Number of chains = 1
Sample size per chain = 800
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean      SD Naive SE Time-series SE
[1,] 1.072 0.06854 0.002423       0.008999
[2,] 2.597 0.13593 0.004806       0.019700
2. Quantiles for each variable:
2.5%   25%   50%   75% 97.5%
var1 0.9386 1.024 1.071 1.117 1.213
var2 2.3224 2.500 2.603 2.686 2.877
```

All posterior mean estimates are close to the population parameters, and the 95% credible intervals encompass the population parameters. We use the definition of γ in the last line of the code.

8. **Simulation of the negative binomial model continues.**

   Perform inference in the simulated negative binomial model using the *bayesm* package in **R**.

   Answer

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)
table(y)
# MCMC parameters
mcmc <- 10000
burnin <- 1000
thin <- 5
iter <- mcmc + burnin
keep <- seq(burnin, iter, thin)
sbeta <- 2.93/sqrt(k); salpha <- 2.93
# Hyperparameters: Priors
B0 <- 1000*diag(k); b0 <- rep(0, k)
alpha0 <- 0.5; delta0 <- 0.1
DataNB <- list(y = y, X = X)
mcmcNB <- list(R = mcmc, keep = thin, s_beta = sbeta, s_
    alpha = salpha)
PriorNB <- list(betabar = b0, A = solve(B0), a = alpha0, b =
     delta0)
ResultBayesm <- bayesm::rnegbinRw(Data = DataNB, Mcmc =
    mcmcNB, Prior = PriorNB)
summary(ResultBayesm$alphadraw)
summary(ResultBayesm$betadraw)
```

9. **The market value of soccer players in Europe continues.**

   Perform the application on soccer players' market value with left censoring

at one million euros in our GUI using Algorithm **??** and the hyperparameters of the example.

Answer

These are the results of running the application in our GUI. These are very similar compared with the results in the book.

R. code. The value of soccer players with left censoring in our GUI

```
Summary
Iterations = 10001:60000
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
(Intercept)  1.014765 2.626395 1.175e-02      1.659e-02
Perf          0.033935 0.004529 2.025e-05      2.245e-05
Age           1.027285 0.212413 9.499e-04      1.326e-03
Age2         -0.021914 0.003989 1.784e-05      2.528e-05
NatTeam       0.847412 0.124884 5.585e-04      6.423e-04
Goals         0.010092 0.001644 7.351e-06      7.712e-06
Exp           0.175324 0.069653 3.115e-04      3.727e-04
Exp2         -0.005696 0.002950 1.319e-05      1.527e-05
sigma2        0.983337 0.096194 4.302e-04      6.724e-04
2. Quantiles for each variable:
             2.5%        25%        50%        75%       97.5%
(Intercept) -4.212563 -0.732856  1.038281  2.796603  6.051e
    +00
Perf          0.025121  0.030873  0.033896  0.036988  4.286e
    -02
Age           0.618743  0.883011  1.026760  1.168860  1.450e
    +00
Age2         -0.029854 -0.024580 -0.021886 -0.019203 -1.424e
    -02
NatTeam       0.605030  0.763154  0.846671  0.930593  1.095e
    +00
Goals         0.006893  0.008987  0.010097  0.011201  1.330e
    -02
Exp           0.038522  0.128032  0.175261  0.222110  3.118e
    -01
Exp2         -0.011484 -0.007671 -0.005685 -0.003706  5.502e
    -05
sigma2        0.813118  0.915664  0.977298  1.043661  1.189e
    +00
```

10. **The market value of soccer players in Europe continues.**

    Program from scratch the Gibbs sampling algorithm for the example of soccer players' market value at the 0.75 quantile.

    Answer

R. code. The value of soccer players, 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(Value); X <- cbind(1, Perf, Age, Age2, NatTeam,
    Goals, Exp, Exp2)
RegLS <- lm(y ~ X -1)
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 <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 1
# Quantile
tau <- 0.5; theta <- (1-2*tau)/(tau*(1-tau))
psi2 <- 2/(tau*(1-tau)); an2 <- 2+theta^2/psi2
# Gibbs sampler
PostBeta <- function(e){
  Bn <- solve(B0i + psi2^(-1)*t(X)%*%diag(1/e)%*%X)
  bn <- Bn%*%(B0i%*%b0 + psi2^(-1)*t(X)%*%diag(1/e)%*%(y-
    theta*e))
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostE <- function(Beta, i){
  dn2 <-(y[i]-X[i,]%*%Beta)^2/psi2
  ei <- GIGrvg::rgig(1, chi = dn2, psi = an2, lambda = 1/2)
  return(ei)
}
PostBetas <- matrix(0, mcmc+burnin, k)
Beta <- RegLS$coefficients
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  e <- sapply(1:N, function(i){PostE(Beta = Beta, i)})
  Beta <- PostBeta(e = e)
  PostBetas[s,] <- Beta
  setTxtProgressBar(pb, s)
}
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))
```

11. Use the *bayesboot* package to perform inference in the simulation exercise of Section **??** and compare the results with those obtained from our GUI, setting $S = 10{,}000$.

Answer

R. code. The value of soccer players, quantile regression

```
########################## Bayesian bootstrap: Simulation
    ##########################
rm(list = ls())
set.seed(010101)
N <- 1000 # Sample size
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)
}
Reg(data)
S <- 10000
BB <- bayesboot::bayesboot(data = data, statistic = Reg, R =
    S)
plot(BB)
```

7

Multivariate models

Solutions of Exercises

1. Show that $\mathbb{E}[u_1 \text{PAER}] = \frac{\alpha_1}{1-\beta_1\alpha_1}\sigma_1^2$, assuming that $\mathbb{E}[u_1 u_2] = 0$ and $\text{Var}(u_1) = \sigma_1^2$, in the example on the effect of institutions on per capita GDP.

   Answer

   The point of departure is the following *simultaneous structural* economic model:

$$\log(\text{pcGDP95}_i) = \beta_1 + \beta_2 \text{PAER}_i + \beta_3 \text{Africa} + \beta_4 \text{Asia} + \beta_5 \text{Other} + u_{1i}, \tag{7.1}$$

$$\text{PAER}_i = \alpha_1 + \alpha_2 \log(\text{pcGDP95}_i) + \alpha_3 \log(\text{Mort}_i) + u_{2i}, \tag{7.2}$$

   where *pcGDP95*, *PAER* and *Mort* are 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 time of colonization. *Africa*, *Asia* and *Other* are dummies for continents, with *America* as the baseline group.

   Replacing Equation 7.2 into Equation 7.1, and solving for $\log(pcGDP95)$,

$$\log(\text{pcGDP95}_i) = \pi_1 + \pi_2 \log(\text{Mort}_i) + \pi_3 \text{Africa} + \pi_4 \text{Asia} + \pi_5 \text{Other} + e_{1i}, \tag{7.3}$$

   where $\pi_1 = \frac{\beta_1+\beta_2\alpha_1}{1-\beta_2\alpha_2}$, $\pi_2 = \frac{\beta_2\alpha_3}{1-\beta_2\alpha_2}$, $\pi_3 = \frac{\beta_3}{1-\beta_2\alpha_2}$, $\pi_4 = \frac{\beta_4}{1-\beta_2\alpha_2}$, $\pi_5 = \frac{\beta_5}{1-\beta_2\alpha_2}$, and $e_1 = \frac{\beta_2 u_2+u_1}{1-\beta_2\alpha_2}$.

   Then, replacing Equation 7.3 into Equation 7.2, and solving for *PAER*,

$$\text{PAER}_i = \gamma_1 + \gamma_2 \log(\text{Mort}_i) + \gamma_3 \text{Africa} + \gamma_4 \text{Asia} + \gamma_5 \text{Other} + e_{2i}, \tag{7.4}$$

   where $\gamma_1 = \frac{\alpha_1+\alpha_2\beta_1}{1-\beta_2\alpha_2}$, $\gamma_2 = \frac{\alpha_3}{1-\beta_2\alpha_2}$, $\gamma_3 = \frac{\alpha_2\beta_3}{1-\beta_2\alpha_2}$, $\gamma_4 = \frac{\alpha_2\beta_4}{1-\beta_2\alpha_2}$, $\gamma_5 = \frac{\alpha_2\beta_5}{1-\beta_2\alpha_2}$, and $e_2 = \frac{\alpha_2 u_1+u_2}{1-\beta_2\alpha_2}$.

   Then, $\mathbb{E}[u_1 \text{PAER}] = \mathbb{E}\left[u_1(\gamma_1 + \gamma_2 \log(\text{Mort}_i) + \gamma_3 \text{Africa} + \gamma_4 \text{Asia} + \gamma_5 \text{Other} + e_{2i})\right] = \mathbb{E}\left[u_1\left(\frac{\alpha_2 u_1+u_2}{1-\beta_2\alpha_2}\right)\right] = \frac{\alpha_2}{1-\beta_2\alpha_2}\sigma_1^2$ given the assumptions.

2. Show that $\beta_1 = \pi_1/\gamma_1$ in the example on the effect of institutions on per capita GDP.

   Answer

   Given that $\pi_2 = \frac{\beta_2\alpha_3}{1-\beta_2\alpha_2}$ and $\gamma_2 = \frac{\alpha_3}{1-\beta_2\alpha_2}$ from the previous exercise, consequently, $\beta_2 = \pi_2/\gamma_2$.

3. **The effect of institutions on per capita GDP 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.

   Answer

R code. The effect of institutions on per capita GDP

```
rm(list = ls())
set.seed(010101)
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]
# Hyperparameters
B0 <- matrix(0, K, M)
c0 <- 100
V0 <- c0*diag(K)
Psi0 <- 5*diag(M)
a0 <- 5
S <- 10000 #Number of posterior draws
betadraw = matrix(double(S*K*M), ncol=K*M)
Sigmadraw = matrix(double(S*M*M), ncol=M*M)
pb <- txtProgressBar(min = 0, max = S, style = 3)
for (s in 1:S) {
  Results <- bayesm::rmultireg(Y, X, Bbar = B0, A = solve(V0
    ), nu = a0, V = Psi0)
  betadraw[s,] <- Results$B
  Sigmadraw[s,] <- Results$Sigma
  setTxtProgressBar(pb, s)
}
close(pb)
summary(coda::mcmc(betadraw))
summary(coda::mcmc(Sigmadraw))
```

4. **Demand and supply simulation**

Consider 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 and q^s represent demand and supply, respectively; p, y, pc, ps, and er denote price, income, complementary price, substitute price, and exchange rate. The variables pc and ps 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 $er \sim N(15, 1)$.

- Derive the *reduced-form* model under the equilibrium condition $q^d = q^s$, which 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. Discuss any potential issues. *Hint:* Are all structural parameters exactly identified?

Answer

We should equal demand and supply, and solve for price,

$$p = \pi_1 + \pi_2 er + \pi_3 y + \pi_4 pc + \pi_5 ps + v_1,$$

where $\pi_1 = \frac{\alpha_1 - \beta_1}{\beta_2 - \alpha_2}$, $\pi_2 = \frac{\alpha_3}{\beta_2 - \alpha_2}$, $\pi_3 = \frac{-\beta_3}{\beta_2 - \alpha_2}$, $\pi_4 = \frac{-\beta_4}{\beta_2 - \alpha_2}$, $\pi_5 = \frac{-\beta_5}{\beta_2 - \alpha_2}$, and $v_1 = \frac{u_2 - u_1}{\beta_2 - \alpha_2}$ given $\beta_2 \neq \alpha_2$, that is, the equations should be independent. This condition is given by economic theory due to $\beta_2 < 0$ and $\alpha_2 > 0$, the effect of price on demand and supply should be negative and positive, respectively.

The equation of price into the demand equation gives

$$q = \tau_1 + \tau_2 er + \tau_3 y + \tau_4 pc + \tau_5 ps + v_2,$$

where $\tau_1 = \beta_1 + \beta_2 \pi_1$, $\tau_2 = \beta_2 \pi_2$, $\tau_3 = \beta_2 \pi_3 + \beta_3$, $\tau_4 = \beta_2 \pi_4 + \beta_4$, $\tau_5 = \beta_2 \pi_5 + \beta_5$, and $v_2 = \beta_2 v_1 + u_1$. We can use the equations para π_k and τ_k, $k = \{1, 2, 3, 4, 5\}$ to simulate the *reduced-form* equations.

Observe that estimating the *reduced-form* equations, we can get the *structural* parameters for the demand equation, $\beta_2 = \tau_2 / \pi_2$ ($\pi_2 \neq 0$), $\beta_3 = \tau_3 - \beta_2 \pi_3$, $\beta_4 = \tau_4 - \beta_2 \pi_4$ and $\beta_5 = \tau_5 - \beta_2 \pi_5$, whereas the *structural* parameters of the supply equation cannot be recovered just in one way, $\alpha_2 = \beta_2 + \beta_3 / \pi_3 = \beta_2 + \beta_4 / \pi_4 = \beta_2 + \beta_5 / \pi_5$. This in turn implies different

values for $\alpha_3 = \pi_2(\beta_2 - \alpha_2)$. This is because the demand equation is *exactly identified*, whereas the supply equation is *over identified*.

In this exercise, $K = 5$, $M = 2$, $K_1 = 4$, $K_2 = 2$, $M_1 = 2$ 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 (demand and supply) satisfy the necessary condition of identification, the demand would be *exactly identified*, and the supply equation would be *over identified*.

Regarding the rank condition (necessary and sufficient), let's see the identification matrix:

TABLE 7.1
Identification matrix.

q	p	constant	er	y	pc	ps
1	-β_2	-β_1	0	-β_3	-β_4	-β_5
1	-α_2	$-\alpha_1$	-α_3	0	0	0

The demand equation excludes the exchange rate (*exclusion restriction*), and given $\alpha_3 \neq 0$, that is, the exchange rate is relevant in the supply equation, then the rank condition is satisfied in the demand equation. The supply equation excludes the income, the complementary price and substitute price (*exclusion restrictions*), then as far as $\beta_k \neq 0$ for any $k = \{3, 4, 5\}$, the rank condition is satisfied in the supply equation.

The following code shows how to do the simulation, and perform inference in this exercise. We can see that all 95% credible intervals encompass the population parameters, and the posterior means are very close to them.

R code. Demand and supply simulation

```
rm(list = ls()); set.seed(12345)
B0 <- 5; B1 <- -0.5; B2 <- 0.8; B3 <- -0.4; B4 <- 0.7; SD <- 0.5
A0 <- -2; A1 <- 0.5; A2 <- -0.4; SS <- 0.5
P0 <- (A0-B0)/(B1-A1); P2 <- -B2/(B1-A1); P3 <- -B3/(B1-A1); P1 <- A2/(B1-A1); P4 <- -B4/(B1-A1)
T0 <- B0+B1*P0; T2 <- B2+B1*P2; T3 <- B3+B1*P3; T1 <- B1*P1; T4 <- B4+B1*P4;
n <- 5000
ED <- rnorm(n, 0, SD); ES <- rnorm(n, 0, SS)
VP <- (ES-ED)/(B1-A1); UQ <- B1*VP+ED
y <- rnorm(n, 10, 1); pc <- rnorm(n, 5, 1); er <- rnorm(n, 15, 1); ps <- rnorm(n, 5, 1);
p <- P0+P1*er+P2*y+P3*pc+P4*ps+VP
q <- T0+T1*er+T2*y+T3*pc+T4*ps+UQ
#Inference
Y <- cbind(p, q); X <- cbind(1, er, y, pc, ps)
M <- dim(Y)[2]; K <- dim(X)[2]
# Hyperparameters
B0 <- matrix(0, K, M); c0 <- 100; V0 <- c0*diag(K)
Psi0 <- 5*diag(M); a0 <- 5; S <- 10000 #Posterior draws
betadraw = matrix(double(S*K*M), ncol=K*M)
Sigmadraw = matrix(double(S*M*M), ncol=M*M)
pb <- txtProgressBar(min = 0, max = S, style = 3)
for (s in 1:S) {
  Results <- bayesm::rmultireg(Y, X, Bbar = B0, A = solve(V0), nu = a0, V = Psi0)
  betadraw[s,] <- Results$B
  Sigmadraw[s,] <- Results$Sigma
  setTxtProgressBar(pb, s)
}
close(pb)
summary(coda::mcmc(betadraw))
summary(coda::mcmc(Sigmadraw))
beta2 <- betadraw[,7]/betadraw[,2] # Effect of price on demand
summary(coda::mcmc(beta2))
beta3 <- betadraw[,8] - beta2*betadraw[,3] # Effect of income on demand
summary(coda::mcmc(beta3))
beta4 <- betadraw[,9] - beta2*betadraw[,4] # Effect of complementary price on demand
summary(coda::mcmc(beta4))
beta5 <- betadraw[,10] - beta2*betadraw[,5] # Effect of substitute price  on demand
summary(coda::mcmc(beta5))
```

5. **Utility demand continues**

   - Run the **Utility demand** application using our GUI and the dataset *Utilities.csv*. *Hint:* This file must be modified to match the structure required by our GUI (see the dataset *5Institutions.csv* in the *DataApp* folder of our GitHub repository—**https://github.com/besmarter/BSTApp**—as a template).
   - Program the Gibbs sampling algorithm for this application from scratch.

   Answer

R code. Utility demand in Colombia, Gibss sampler

```
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)
y <- c(y1, y2, y3)
X <- as.matrix(Matrix::bdiag(X1, X2, X3))
M <- 3; K1 <- dim(X1)[2]; K2 <- dim(X2)[2]; K3 <- dim(X3)[2]
K <- K1 + K2 + K3; N <- length(y1)
# Hyperparameters
b0 <- rep(0, K); c0 <- 100; B0 <- c0*diag(K)
B0i <- solve(B0); Psi0 <- 5*diag(M); Psi0i <- solve(Psi0)
a0 <- M; IN <- diag(N); an <- a0 + N
#Posterior draws
S <- 6000; burnin <- 1000; thin <- 1
tot <- S+burnin
# Gibbs functions
PostBeta <- function(Sigma){
  Aux <- solve(Sigma)%x%IN
  Bn <- solve(B0i + t(X)%*%Aux%*%X)
  bn <- Bn%*%(B0i%*%b0 + t(X)%*%Aux%*%y)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostSigma <- function(Beta){
  B1 <- Beta[1:K1]; B2 <- Beta[(K1+1):(K1+K2)]; B3 <- Beta[(
    K1+K2+1):(K1+K2+K3)]
  U1 <- y1 - X1%*%B1; U2 <- y2 - X2%*%B2; U3 <- y3 - X3%*%B3
  U <- cbind(U1, U2, U3)
  Psin <- solve(Psi0i + t(U)%*%U)
  Sigmai <- rWishart::rWishart(1, df = an, Sigma = Psin)
  Sigma <- solve(Sigmai[,,1])
  return(Sigma)
}
PostBetas <- matrix(0, tot, K)
PostSigmas <- matrix(0, tot, M*(M+1)/2); Beta <- rep(1, K)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  Sigma <- PostSigma(Beta = Beta); PostSigmas[s,] <-
    matrixcalc::vech(Sigma)
  Beta <- PostBeta(Sigma = Sigma);  PostBetas[s,] <- Beta
  setTxtProgressBar(pb, s)
}
close(pb); keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]; summary(coda::mcmc(Bs))
Sigmas <- PostSigmas[keep,]; summary(coda::mcmc(Sigmas))
```

R code. Utility demand in Colombia, results

```
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]; summary(coda::mcmc(Bs))
Sigmas <- PostSigmas[keep,]; summary(coda::mcmc(Sigmas))
Quantiles for each variable:
                2.5%      25%       50%      75%    97.5%
Const          0.43304  1.04508  1.337751  1.64728  2.25228
LnPriceElect  -2.40343 -2.05533 -1.882159 -1.70463 -1.35979
LnPriceWater  -0.44027 -0.38524 -0.356049 -0.32676 -0.27021
LnPriceGas    -0.21665 -0.13807 -0.098877 -0.05712  0.01673
IndSocio1     -0.87408 -0.78454 -0.736561 -0.68898 -0.59137
IndSocio2     -0.24504 -0.18344 -0.150827 -0.11845 -0.06029
Altitude      -0.27153 -0.23866 -0.221539 -0.20468 -0.17325
Nrooms         0.04598  0.06191  0.069971  0.07837  0.09379
HouseholdMem   0.06610  0.07999  0.087320  0.09406  0.10708
Lnincome       0.03897  0.05516  0.063130  0.07152  0.08751
Const          0.88247  1.73546  2.162290  2.60793  3.48450
LnPriceElect  -0.83128 -0.31258 -0.051838  0.20446  0.72087
LnPriceWater  -0.49760 -0.40946 -0.363138 -0.31943 -0.23498
LnPriceGas     0.05880  0.16932  0.228734  0.28677  0.39226
IndSocio1     -0.63928 -0.50417 -0.427978 -0.35529 -0.21024
IndSocio2     -0.50363 -0.40976 -0.359576 -0.30929 -0.21097
Nrooms         0.05733  0.08126  0.093291  0.10550  0.12935
HouseholdMem   0.09984  0.12098  0.132126  0.14265  0.16290
Const         -2.26403 -1.57842 -1.201124 -0.84494 -0.15985
LnPriceElect  -2.41128 -2.00330 -1.786150 -1.57292 -1.17185
LnPriceWater  -0.10791 -0.03978 -0.003399  0.03329  0.10082
LnPriceGas    -0.76450 -0.67575 -0.627062 -0.57949 -0.48807
IndSocio1     -0.91346 -0.80094 -0.743421 -0.68334 -0.57228
IndSocio2     -0.31613 -0.24086 -0.202828 -0.16367 -0.08824
Altitude       0.24986  0.28987  0.310784  0.33183  0.37126
Nrooms         0.06071  0.07899  0.089606  0.09949  0.11842
HouseholdMem   0.14485  0.16148  0.169509  0.17797  0.19467
Sigmas <- PostSigmas[keep,]
summary(coda::mcmc(Sigmas))
        2.5%     25%     50%     75%   97.5%
var1 0.19664 0.20567 0.21072 0.21593 0.22636
var2 0.08201 0.09289 0.09865 0.10447 0.11623
var3 0.05222 0.06015 0.06452 0.06903 0.07799
var4 0.47390 0.49575 0.50789 0.52046 0.54446
var5 0.07355 0.08672 0.09381 0.10061 0.11427
var6 0.29259 0.30618 0.31384 0.32140 0.33700
```

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 example, set $\gamma_2 =$

0.2 and compare the posterior mean estimates 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.

Answer

R code. Simulation exercise, sampling properties of instrumental variable

```
rm(list = ls())
set.seed(010101)
N <- 100
k <- 2
B <- rep(1, k)
G <- c(1, 0.2)
s12 <- 0.8
SIGMA <- matrix(c(1, s12, s12, 1), 2, 2)
z <- rnorm(N); # w <- 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,]
VarX <- G[2]^2+1 # Population variance of x
EU1U2 <- s12 # Covariance U1
BiasPopB2 <- EU1U2/VarX
# 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)
1.781513 1.470648
```

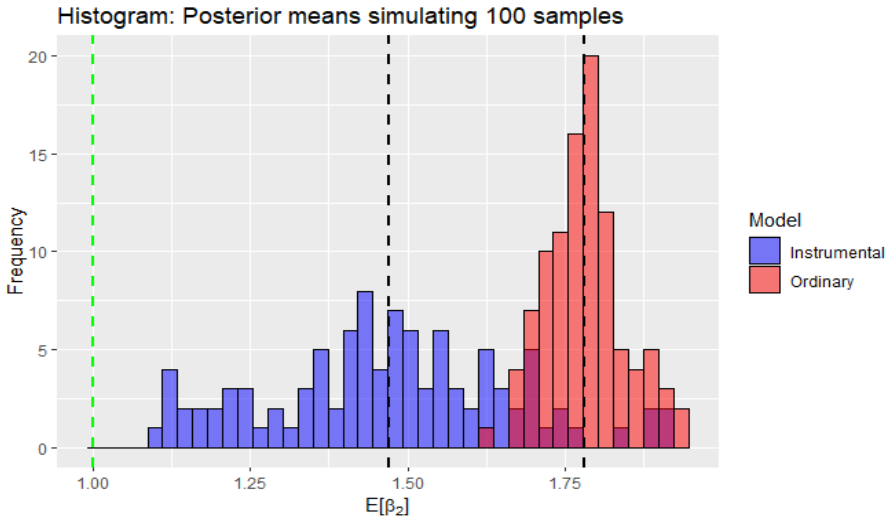

FIGURE 7.1
Histogram of posterior means to assess the effects of *weakness* of the instrument: Ordinary and instrumental models.

Figure 7.1 shows that having a *weak instrument* generates poor performance of the posterior means of the variable instrumental model. Observe that the mean of the posterior means is less precise and biased. This is due to an identification problem in the likelihood function of y and x_s given z (see [40, Chap. 7]).

R code. Simulation exercise, sampling properties of instrumental variable

```
# Endogenous instruments
rm(list = ls())
set.seed(010101)
N <- 100
k <- 2
B <- rep(1, k)
G <- c(1, 1)
delta <- 0.2 # Effect of instrument on y
s12 <- 0.8
SIGMA <- matrix(c(1, s12, s12, 1), 2, 2)
z <- rnorm(N); # w <- 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 + delta*z + U[,1,]
VarX <- G[2]^2+1 # Population variance of x
EU1U2 <- s12 # Covariance U1
BiasPopB2 <- EU1U2/VarX
# 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)
1.508009 1.217419
```

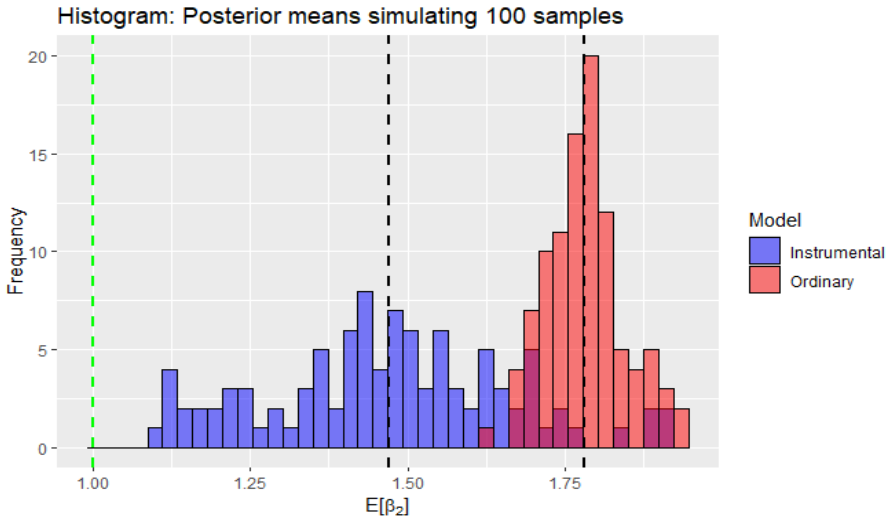

FIGURE 7.2
Histogram of posterior means to assess the effects of *endogeneity* of the instrument: Ordinary and instrumental models.

A simple way to assess the effects of *endogeneity* of the instrument in the instrumental variable model is to introduce the instrument into the *structural equation*, but assume that $\mathbb{E}[yz|x_s] = 0$. In particular, we generate $y_i = \beta_1 + \beta_2 x_{si} + \beta_3 z_i + \mu_i$, but estimate $y_i = \beta_1 + \beta_2 x_{si} + \mu_i$. Figure 7.2 shows that assuming that an instrument is exogenous, when actually it is not, generates bias in the posterior mean of the instrumental variable model. Observe that *exogeneity* of the instrument is equivalent to the dogmatic prior belief that the instrument does not affect the structural equation. Thus, [7] propose to assess the plausibility of the *exclusion restrictions* ($\beta_3 = 0$ and $\gamma_2 \neq 0$) 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.

7. **Simulation exercise of instrumental variables continues II**

   Program the Gibbs sampling algorithm for the instrumental variable model from scratch.

   Answer

   The following code shows how to implement the Gibbs sampling algorithm from scratch. We observe that all posterior means are close to the population values, and 95% credible intervals encompass them.

R code. Simulation exercise, Gibss sampling instrumental model

```
rm(list = ls()); set.seed(010101)
N <- 100; k <- 2; kz <- 2
B <- rep(1, k); G <- rep(1, kz)
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 <- 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]
X <- cbind(w, x)
# Hyperparameters
b0 <- rep(0, k); c0 <- 1000; B0 <- c0*diag(k)
B0i <- solve(B0); g0 <- rep(0, kz)
G0 <- 1000*diag(kz); G0i <- solve(G0)
nu <- 3; Psi0 <- nu*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 + N
# 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. Simulation exercise, Gibss sampling instrumental model

```
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 <- txtProgressBar(min = 0, max = tot, style = 3)
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)
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]
Gs <- PostGammas[keep,]
Sigmas <- PostSigmas[keep,]
summary(coda::mcmc(Bs))
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean      SD Naive SE Time-series SE
[1,] 1.1519 0.1683  0.00238       0.006321
[2,] 0.9896 0.1146  0.00162       0.005177
summary(coda::mcmc(Gs))
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean      SD Naive SE Time-series SE
[1,] 1.133 0.1068 0.001511       0.003096
[2,] 1.030 0.1066 0.001507       0.004193
summary(coda::mcmc(Sigmas))
1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
Mean      SD Naive SE Time-series SE
[1,] 1.3883 0.3149 0.004454       0.012707
[2,] 0.9992 0.2133 0.003017       0.007166
[3,] 1.1637 0.1685 0.002382       0.002481
```

8. **The effect of institutions on per capita GDP continues II**

Estimate the structural Equation 7.1 using the instrumental variable model, where $\log(Mort)$ serves as the instrument for PAER. Compare the estimated effect of property rights on per capita GDP with that obtained in the example on the effect of institutions on per capita GDP. Use the file *6Institutions.csv* to conduct this exercise in our GUI, setting $\boldsymbol{B}_0 = 100\boldsymbol{I}_5$,

$\boldsymbol{\beta}_0 = \mathbf{0}_5$, $\boldsymbol{\gamma}_0 = \mathbf{0}_2$, $\boldsymbol{G}_0 = 100\boldsymbol{I}_2$, $\alpha_0 = 3$, and $\boldsymbol{\Psi}_0 = 3\boldsymbol{I}_2$. Set the MCMC parameters as follows: 50,000 iterations, 1,000 burn-in, and a thinning interval of 5.

Answer

The following code shows how to do this using the command *rivGibss* from the *bayesm* package. We see that the posterior mean of the effect of property rights on per capita GDP is 0.72, and the 95% credible interval is (0.47, 1.00). The posterior mean estimate of this effect was 0.98 in the example, and the 95% credible interval was (0.56, 2.87) (see Section 8.1 in the book). We can observe that the indirect approach, where the *reduced-form* parameters are estimated, and then, plug into the equations of the *structural parameters* produce less efficient posterior estimations.

R code. The effects of institutions on GDP, variable instrumental model

```
rm(list = ls()); set.seed(010101)
DataInst <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/6Institutions
    .csv", sep = ",", header = TRUE, quote = "")
attach(DataInst)
y <- logpcGDP95; x <- PAER
w <- cbind(1, Africa, Asia, Other); Z <- cbind(1, logMort)
# Hyperparameters
k <- 5; kz <- 2; b0 <- rep(0, k); c0 <- 100
B0 <- c0*diag(k); B0i <- solve(B0); g0 <- rep(0, kz)
G0 <- 100*diag(kz); G0i <- solve(G0); nu <- 5
Psi0 <- nu*diag(2)
# MCMC parameters
mcmc <- 50000; burnin <- 10000; tot <- mcmc + burnin; thin
    <- 5
# Gibbs sampling
Data <- list(y = y, x = x, w = w, z = Z)
Mcmc <- list(R = mcmc, keep = thin, nprint = 100)
Prior <- list(md = g0, Ad = G0i, mbg = b0, Abg = B0i, nu =
    nu, V = Psi0)
RestIV <- bayesm::rivGibbs(Data = Data, Mcmc = Mcmc, Prior =
     Prior)
summary(RestIV[["deltadraw"]])
summary(coda::mcmc(RestIV[["betadraw"]]))
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
0.759497       0.140874       0.001409       0.002477
2. Quantiles for each variable:
2.5%    25%    50%    75%  97.5%
0.5110 0.6654 0.7502 0.8424 1.0707
summary(RestIV[["gammadraw"]])
summary(RestIV[["Sigmadraw"]])
```

9. **Multivariate probit with different regressors**

   Simulate the model

   $$y_{i1}^* = 0.5 - 1.2x_{i11} + 0.7x_{i12} + 0.8x_{i3} + \mu_{i1}, \quad y_{i2}^* = 1.5 - 0.8x_{i21} + 0.5x_{i22} + \mu_{i2},$$

   with

   $$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix},$$

where all regressors follow a standard normal distribution and $N = 5000$. Use $\boldsymbol{\beta}_0 = \mathbf{0}$, $\boldsymbol{B}_0 = 1000\boldsymbol{I}$, $\alpha_0 = 4$, and $\boldsymbol{\Psi}_0 = 4\boldsymbol{I}_2$. Set the number of iterations to 2,000 and the thinning parameter to 5.

- Perform inference under the setting of Section 7.4, assuming that x_{i3} may affect y_{i2}.
- Program a Gibbs sampling algorithm that accounts for different regressors in each binary decision, that is, where x_{i3} does not affect y_{i2}.

Answer

The following code shows how to perform inference assuming same number of regressors in both binary results. We can check that all 95% credible intervals encompass the population values, and the posterior mean estimates are very close to the population parameters.

R code. Multivariate probit model: Same number of regressors in each binary response

```
remove(list = ls())
n <-5000  #Individuals
p <-2  #Number dependent variables (For instance to buy or not to buy different products)
xo <- 2  #Number of choice dependent variables (For instance price of products)
xi <- 1  #Number of regressors that dependt on individuals (For instance income)
B1 <- c(0.5,-1.2,0.7,0.8)
B2 <- c(1.5,-0.8,0.5,0) # The last regressor is not relevant for the second product
Cor <- matrix(c(1,0.5,0.5,1),p,p)
yX <- NULL; yl <- NULL
for (i in 1:n){
  ei <- MASS::mvrnorm(1, c(0,0), Cor);  xs <- rnorm(xi)
  x1i <- c(1, rnorm(xo), xs);   yl1i <- sum(x1i*B1)
  y1i <- yl1i + ei[1] > 0;  x2i <- c(1, rnorm(xo), xs)
  yl2i <- sum(x2i*B2);  y2i <- yl2i + ei[2] > 0
  yXi <- rbind(c(y1i, x1i), c(y2i, x2i))
  yXi <- cbind(i, yXi)
  colnames(yXi) <- c("id", "y", "cte", "x1o", "x2o", "xi")
  yX <- rbind(yX, yXi)
  yl <- c(yl, c(yl1i + ei[1], yl2i + ei[2]))
}

nd <- 4 # Number of regressors
Xd <- as.matrix(yX[,3:6])
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 = yX)
```

R code. Multivariate probit model: Same number of regressors in each binary response

```
y<-NULL
X<-NULL
for(i in 1:dim(Dat$y)[3]){
  y<-c(y,Dat$y[,,i])
  X<-rbind(X,Dat$X[,,i])
}
DataMP = list(p=p, y=y, X=X)
# Hyperparameters
k <- dim(X)[2]
b0 <- rep(0, k)
c0 <- 1000
B0 <- c0*diag(k)
B0i <- solve(B0)
a0 <- p - 1 + 3
Psi0 <- a0*diag(p)
Prior <- list(betabar = b0, A = B0i, nu = a0, V = Psi0)
# MCMC parameters
mcmc <- 2000
burnin <- 1000
thin <- 5
Mcmc <- list(R = mcmc, keep = thin)
Results <- bayesm::rmvpGibbs(Data = DataMP, Mcmc = Mcmc,
    Prior = Prior)
betatildeEq1 <- Results$betadraw[,1:4] / sqrt(Results$
    sigmadraw[,1])
summary(coda::mcmc(betatildeEq1))
betatildeEq2 <- Results$betadraw[,5:8] / sqrt(Results$
    sigmadraw[,4])
summary(coda::mcmc(betatildeEq2))
sigmadraw12 <-  Results$sigmadraw[,2] / (Results$sigmadraw
    [,1]*Results$sigmadraw[,4])^0.5
summary(coda::mcmc(sigmadraw12))
```

The following code shows the implementation of a Gibbs sampling algorithm using the *bayesm* package, particularly, the *rmvpGibbs* command.

R code. Multivariate probit model: Different number of regressors in each binary response, the rmvpGibbs command

```
# Omitting last regressor for y2
Xnew <- X[,-8]
DataMP = list(p=p, y=y, X=Xnew)
# Hyperparameters
k <- dim(Xnew)[2]
b0 <- rep(0, k)
c0 <- 1000
B0 <- c0*diag(k)
B0i <- solve(B0)
Prior <- list(betabar = b0, A = B0i, nu = a0, V = Psi0)
Results <- bayesm::rmvpGibbs(Data = DataMP, Mcmc = Mcmc,
    Prior = Prior)
betatildeEq1 <- Results$betadraw[,1:4] / sqrt(Results$
    sigmadraw[,1])
summary(coda::mcmc(betatildeEq1))
betatildeEq2 <- Results$betadraw[,5:7] / sqrt(Results$
    sigmadraw[,4])
summary(coda::mcmc(betatildeEq2))
sigmadraw12 <-  Results$sigmadraw[,2] / (Results$sigmadraw
    [,1]*Results$sigmadraw[,4])^0.5
summary(coda::mcmc(sigmadraw12))
```

The following code shows how to implement a Gibbs sampling algorithm from scratch.

R code. Multivariate probit model: Different number of regressors in each binary response, Gibbs from scratch

```
PostBetaNew <- function(Sigma, Yl){
  Sigmai <- solve(Sigma); C <- chol(Sigmai)
  Xstar <- matrix(0, n*p, k); Ystar <- matrix(0, n*p, 1)
  for(i in seq(1, n*p, 2)){
    Xstari <- C%*%Xnew[i:(i+1),]
    Xstar[i:(i+1),] <- Xstari
    Ystari <- C%*%Yl[i:(i+1)]
    Ystar[i:(i+1)] <- Ystari
  }
  Bn <- solve(B0i + t(Xstar)%*%Xstar)
  bn <- Bn%*%(B0i%*%b0 + t(Xstar)%*%Ystar)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
PostSigmaNew <- function(Beta, Yl){
  U <- matrix(0, p, p)
  for(i in seq(1, n*p, 2)){
    Ui <- (Yl[i:(i+1)] - Xnew[i:(i+1),]%*%Beta)%*%t(Yl[i:(i
    +1)] - Xnew[i:(i+1),]%*%Beta)
    U <- U + Ui
  }
  Psin <- solve(Psi0i + U)
  Sigmai <- rWishart::rWishart(1, df = an, Sigma = Psin)
  Sigma <- solve(Sigmai[,,1])
}
PostYlNew <- function(Beta, Sigma, Yl){
  w1 <- Sigma[1, ]
  w2 <- Sigma[2, ]
  f1 <- w1[-1]*Sigma[2, 2]^(-1)
  f2 <- w2[-2]*Sigma[1, 1]^(-1)
  t11 <- Sigma[1, 1] - w1[-1]*Sigma[2, 2]^(-1)*w1[-1]
  t22 <- Sigma[2, 2] - w2[-2]*Sigma[1, 1]^(-1)*w2[-2]
  Ylnew <- Yl
  for(i in seq(1, n*p, 2)){
    Ylmean1 <- Xnew[i,]%*%Beta + f1*(Yl[i+1] - Xnew[i+1,]%*%
    Beta)
    if(y[i] == 1){
      Ylnew[i] <- truncnorm::rtruncnorm(1, a = 0, b = Inf,
    mean = Ylmean1, sd = t11^0.5)
    }else{
      Ylnew[i] <- truncnorm::rtruncnorm(1, a = -Inf, b = 0,
    mean = Ylmean1, sd = t11^0.5)
    }
    Ylmean2 <- Xnew[i+1,]%*%Beta + f2*(Ylnew[i] - Xnew[i,]%*
    %Beta)
    if(y[i+1] == 1){
      Ylnew[i+1] <- truncnorm::rtruncnorm(1, a = 0, b = Inf,
     mean = Ylmean2, sd = t22^0.5)
    }else{
      Ylnew[i+1] <- truncnorm::rtruncnorm(1, a = -Inf, b =
    0, mean = Ylmean2, sd = t22^0.5)
    }
  }
  return(Ylnew)
}
```

R code. Multivariate probit model: Different number of regressors in each binary response, Gibbs from scratch

```
an <- a0 + n
Psi0i <- solve(Psi0)
K1 <- 4; K2 <- 3; K <- K1 + K2
tot <- mcmc + burnin
PostBetas <- matrix(0, tot, K)
PostSigmas <- matrix(0, tot, p*(p+1)/2)
Beta <- rep(1, K)
Sigma <- diag(p)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  Yl <- PostYlNew(Beta = Beta, Sigma = Sigma, Yl = yl)
  Sigma <- PostSigmaNew(Beta = Beta, Yl = Yl)
  Beta <- PostBetaNew(Sigma = Sigma, Yl = Yl)
  PostBetas[s,] <- Beta
  PostSigmas[s,] <- matrixcalc::vech(Sigma)
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]
betatilde1 <- Bs[,1:K1] / sqrt(PostSigmas[keep,1])
summary(coda::mcmc(betatilde1))
betatilde2 <- Bs[,(K1+1):(K1+K2)] / sqrt(PostSigmas[keep,3])
summary(coda::mcmc(betatilde2))
sigmadraw12 <-  PostSigmas[keep,2] / (PostSigmas[keep,1]*
    PostSigmas[keep,3])^0.5
summary(coda::mcmc(sigmadraw12))
```

8

Time series models

Solutions of 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.

   Answer

   The following code shows how to perform this simulation and perform the filtering recursion using the *dlm* package in **R**.

R code. Simulation: Scenarios signal-to-noise ratio

```
rm(list = ls()); set.seed(010101)
T <- 100; sig2 <- 0.5^2; r <- 0.5; sigW2 <- sig2*r
x <- rnorm(T, mean = 1, sd = 0.1*sig2^0.5)
e <- rnorm(T, mean = 0, sd = sig2^0.5)
w <- rnorm(T, mean = 0, sd = sigW2^0.5)
K <- 1
Bt <- matrix(NA, T, K); Bt[1] <- 1
yt <- rep(NA, T)
yt[1] <- x[1]*Bt[1] + 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]
}
# State space model
ModelReg <- function(par){
  Mod <- dlm::dlmModReg(x, dV = exp(par[1]), dW = exp(par
    [2]), m0 = rep(1, K),
  C0 = sigW2*diag(K), addInt = FALSE)
  return(Mod)
}
sig2New <- sig2*25
RegFilter1 <- dlm::dlmFilter(yt, ModelReg(c(sig2New, sig2New
    *0.5)))
ytfil1 <- RegFilter1[["m"]][-1]
RegFilter2 <- dlm::dlmFilter(yt, ModelReg(c(sig2New, sig2New
    *0.1)))
ytfil2 <- RegFilter2[["m"]][-1]
Time <- 1:T
df1 <- as.data.frame(cbind(Time,yt, ytfil1, ytfil2))
library(ggplot2)
ggplot(df1, aes(x=Time)) +
geom_line(aes(y=yt), colour="black", linewidth=1, linetype
    =1) +
geom_line(aes(y=ytfil1), colour="green", linewidth=1, alpha
    =0.9, linetype=2) +
geom_line(aes(y=ytfil2), color="blue", linewidth=1, alpha
    =0.9, linetype=3)
```

Figure 8.1 shows the results of the filtering recursion. We see in this figure that the filtered recursion associated with the high signal-to-noise ratio (green dashed line) follows closer the data than the low signal-to-noise

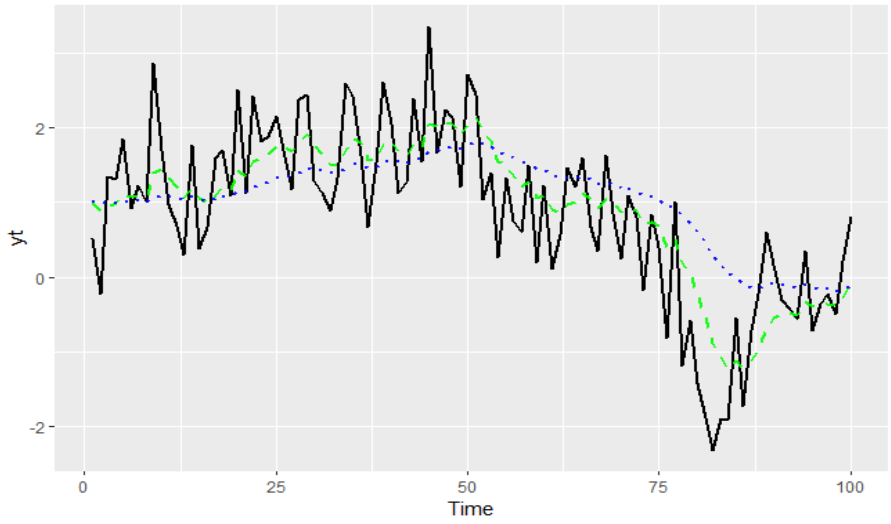

FIGURE 8.1
Signal-to-noise ratio: Dynamic linear model.

ratio scenario (blue dotted line). We see that the latter is smoother as it gives more weight to the prior mean, which takes into account historical information.

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.

   Answer

   The following code shows how to perform this simulation and perform the filtering recursion from scratch in **R**.

R code. Simulation: Filtering and smoothing recursions from scratch

```
rm(list = ls()); set.seed(010101)
T <- 100; sig2 <- 1; r <- 0.5; sigW2 <- sig2*r
xt <- rnorm(T, mean = 1, sd = 0.1*sig2^0.5)
e <- rnorm(T, mean = 0, sd = sig2^0.5)
w <- rnorm(T, mean = 0, sd = sigW2^0.5)
K <- 1; Bt <- matrix(NA, T, K); Bt[1] <- 0
yt <- rep(NA, T)
for(t in 1:T){
  if(t == 1){
    Bt[t,] <- w[t]
  }else{
    Bt[t,] <- Bt[t-1,] + w[t]
  }
  yt[t] <- xt[t]%*%Bt[t,] + e[t]
}
# Filtering
KFR <- function(y, x, b1, B1, Omega, sig2){
  e <- as.numeric(y - t(x)%*%b1) # Error
  R <- B1 + Omega
  q <- as.numeric(t(x)%*%R%*%x + sig2)
  K <- (R%*%x)/q
  bt <- b1 + K*e
  Bt <- R - R%*%x%*%t(x)%*%R/q
  Result <- list(bt = bt, Bt = Bt)
  return(Result)
}
KFresb <- list(); KFresB <- list()
b0 <- 0; B0 <- sigW2
for(t in 1:T){
  KFrest <- KFR(y = yt[t], x = xt[t], b1 = b0, B1 = B0,
    Omega = sigW2, sig2 = sig2)
  b0 <- KFrest[["bt"]]; B0 <- KFrest[["Bt"]]
  KFresb[[t]] <- b0; KFresB[[t]] <- B0
}
ModelReg <- function(par){
  Mod <- dlm::dlmModReg(xt, dV = exp(par[1]), dW = exp(par
    [2]), m0 = rep(0, K),
  C0 = sigW2*diag(K), addInt = FALSE)
  return(Mod)
}
RegFilter1 <- dlm::dlmFilter(yt, ModelReg(c(sig2, sigW2)))
plot(Bt, type = "l")
lines(unlist(KFresb), type = "l", col = "red")
lines(RegFilter1[["m"]][-1], type = "l", col = "blue")
```

R code. Simulation: Filtering and smoothing recursions from scratch

```
# Smoothing
BSR <- function(b, B, slead, Slead, Omega){
  b = bfilter[t]; B = Bfilter[t]; slead = slead; Slead =
    slead; Omega = sigW2
  R1 <- B + Omega
  s <- b + B%*%solve(R1)%*%(slead - b)
  S <- B - B%*%solve(R1)%*%(R1 - Slead)%*%solve(R1)%*%B
  Result <- list(st = s, St = S)
  return(Result)
}
bfilter <- unlist(KFresb); Bfilter <- unlist(KFresB)
slead <- bfilter[T]; Slead <- Bfilter[T]
smooth <- rep(slead, T); Smooth <- rep(Slead, T);
for(t in (T-1):1){
  BSRrest <- BSR(b = bfilter[t], B = Bfilter[t], slead =
    slead, Slead = Slead, Omega = sigW2)
  slead <- BSRrest[["st"]]; Slead <- BSRrest[["st"]]
  smooth[t] <- slead
}
RegSmoth <- dlm::dlmSmooth(yt, ModelReg(c(sig2, sigW2)))
VarSmooth <- dlm::dlmSvd2var(u = RegSmoth[["U.S"]], RegSmoth
    [["D.S"]])
SDVarSmoothB2 <- sapply(2:(T+1), function(t){VarSmooth[[t]][
    K,K]^0.5})
plot(Bt, type = "l")
lines(smooth, type = "l", col = "orange")
lines(RegSmoth[["s"]][-1], type = "l", col = "green")
```

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*.

Answer

R code. Simulation: MCMC algorithm with simulation smoother

```
rm(list = ls()); set.seed(010101)
T <- 200; sig2 <- 0.1; tau2 <- 5; tau <- tau2^0.5; sigW2 <-
    sig2/tau2
xt <- rnorm(T, mean = 1, sd = 2*sig2^0.5)
zt <- rnorm(T, mean = 1, sd = 2*sig2^0.5)
alpha <- 2
e <- MASS::mvrnorm (T, mu = c(0, 0), Sigma = sig2*diag(2))
K <- 1
h1 <- matrix(c(1, 0), K + 1, 1); h2 <- matrix(c(0, 1/tau), K
     + 1, 1); t(h1)%*%h2
e1 <- e%*%h1; e2 <- e%*%h2; var(e1); var(e2)
Bt <- matrix(NA, T, K); Bt[1] <- 0
yt <- rep(NA, T)
for(t in 1:T){
  if(t == 1){
    Bt[t,] <- e2[t]
  }else{
    Bt[t,] <- Bt[t-1,] + e2[t]
  }
  yt[t] <- zt[t]*alpha + xt[t]*Bt[t] + e1[t]
}
# Recursion functions
# Filter
KFRDSnew <- function(y, z, x, b, B, alpha, tau2){
  e <- as.numeric(y - z*alpha - t(x)%*%b)
  q <- as.numeric(t(x)%*%B%*%x + t(h1)%*%h1)
  K <- (B%*%x)/q
  bt <- b + K*e
  h2 <- matrix(c(0, 1/tau2^0.5))
  Bt <- B - B%*%x%*%t(K) + t(h2)%*%h2
  Result <- list(bt = bt, Bt = Bt, et = e, Kt = K, qt = q)
  return(Result)
}
# Smooth
BSRDSnew <- function(r, M, Q, e, K, x, sig2, tau2){
  h2 <- matrix(c(0, 1/tau2^0.5))
  Lambda <- t(h2)%*%h2
  C <- Lambda - Lambda%*%M%*%t(Lambda)
  xi <- rnorm(1, 0, sd = (sig2*C)^0.5)
  L <- 1 - K*x
  V <- Lambda%*%M%*%L
  rtl <- x*e/Q + t(L)*r - t(V)%*%solve(C)%*%xi
  Mtl <- x%*%t(x)/Q + t(L)%*%M%*%L + t(V)%*%solve(C)%*%V
  eta <- Lambda%*%r + xi
  Result <- list(rtl = rtl, Mtl = Mtl, etat = eta)
  return(Result)
}
# Gibbs functions
PostSig2 <- function(bbt, alpha, tau2){
  an <- T*(K + 1) + alpha0
  term1 <- diff(c(0, bbt))
  term2 <- c(yt - alpha*zt - xt*bbt)
  dn <- delta0 + sum(term1^2)*tau2  + sum(term2^2)
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
```

R code. Simulation: MCMC algorithm with simulation smoother

```
PosAlpha <- function(bbt, sig2){
  An <- solve(A0i + sig2^(-1)*t(zt)%*%zt)
  term <- yt-xt*bbt
  an <- An%*%(A0i%*%a0 + sig2^(-1)*t(zt)%*%term)
  Alpha <- MASS::mvrnorm(1, an, An)
  return(Alpha)
}
PostTau2 <- function(bbt, sig2){
  v1n <- v0 + T
  term1 <- diff(c(0,bbt))
  v2n <- v0 + sig2^(-1)*sum(term1^2)
  tau2 <- rgamma(1, v1n/2, v2n/2)
  return(tau2)
}
# Hyperparameter
alpha0 <- 1; delta0 <- 1; v0 <- 5
a0 <- 0; A0 <- 1; A0i <- 1/A0
tau2post <- 2; sig2post <- 0.2; alphapost <- 0
# Inference
S <- 2000; burnin <- 500; tot <- S + burnin; thin <- 5
# Posterior draws
sig2s <- rep(NA, tot); tau2s <- rep(NA, tot)
alphas <- rep(NA, tot); betas <- matrix(NA, tot, T)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
```

R code. Simulation: MCMC algorithm with simulation smoother

```
for (s in 1:tot){
  b0 <- 0; B0 <- sigW2
  KFDSresb <- list(); KFDSresB <- list()
  KFDSresE <- list(); KFDSresK <- list(); KFDSresQ <- list()
  for(t in 1:T){
    KFDSrest <- KFRDSnew(y = yt[t], z = zt[t], x = xt[t], b
    = b0, B = B0, alpha = alphapost, tau2 = tau2post)
    b0 <- KFDSrest[["bt"]]; B0 <- KFDSrest[["Bt"]]
    KFDSresb[[t]] <- b0; KFDSresB[[t]] <- B0
    KFDSresE[[t]] <- KFDSrest[["et"]]; KFDSresK[[t]] <-
    KFDSrest[["Kt"]]; KFDSresQ[[t]] <- KFDSrest[["qt"]]
  }
  et <- unlist(KFDSresE); Qt <- unlist(KFDSresQ); Kt <-
    unlist(KFDSresK)
  rT <- 0; MT <- 0
  BSRDSreseta <- rep(0, T); BSRDSresetaMean <- rep(0, T)
  for(t in (T-1):1){
    BSRDSrest <- BSRDSnew(r = rT, M = MT, Q = Qt[t+1], e =
    et[t+1], K = Kt[t+1], x = xt[t+1], sig2 = sig2post, tau2
     = tau2post)
    rT <- BSRDSrest[["rtl"]]; MT <- BSRDSrest[["Mtl"]]
    BSRDSreseta[t+1] <- BSRDSrest[["etat"]]
  }
  Bt1DSpost <- rep(0,T)
  for(t in 1:T){
    if(t == 1){
      Bt1DSpost[t] <- BSRDSreseta[t]
    }else{
      Bt1DSpost[t] <- Bt1DSpost[t-1] + BSRDSreseta[t]
    }
  }
  sig2post <- PostSig2(bbt = Bt1DSpost, alpha = alphapost,
    tau2 = tau2post)
  alphapost <- PosAlpha(bbt = Bt1DSpost, sig2 = sig2post)
  tau2post <- PostTau2(bbt = Bt1DSpost, sig2 = sig2post)
  sig2s[s] <- sig2post
  tau2s[s] <- tau2post
  alphas[s] <- alphapost
  betas[s,] <- Bt1DSpost
  setTxtProgressBar(pb, s)
}
close(pb); keep <- seq((burnin+1), tot, thin)
betasF <- betas[keep,]; sig2sF <- coda::mcmc(sig2s[keep])
tau2sF <- coda::mcmc(tau2s[keep]); alphasF <- coda::mcmc(
    alphas[keep])
summary(sig2sF); plot(sig2sF)
summary(tau2sF); plot(tau2sF)
summary(alphasF); plot(alphasF)
library(fanplot); df <- as.data.frame(betasF)
plot(NULL, main="Percentiles", xlim = c(1, T+1), ylim = c
    (-4, 1), xlab = "Time", ylab = TeX("$\\beta_{t1}$"))
fan(data = df); lines(colMeans(betasF), col = "black", lw =
    2)
lines(Bt, col = "blue")
```

Figure 8.2 shows the path of the state vector (blue line), and the results from the *simulation smoother* using posterior draws, the black line is the mean, and the red-yellow shadows are different deciles.

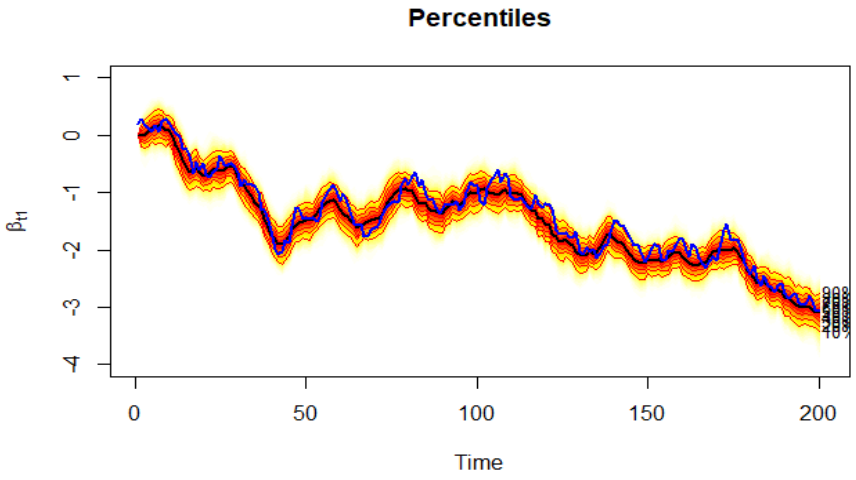

FIGURE 8.2
State vector: Dynamic linear model.

4. Show that the posterior distribution of $\boldsymbol{\phi}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{y}, \boldsymbol{X}$ 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 $\boldsymbol{\phi}$.

Answer

The conditional posterior distribution can be got as follows:

$$\begin{aligned}
\pi(\boldsymbol{\phi}|\boldsymbol{\beta}, \sigma^2, \boldsymbol{y}) &\propto \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{\phi})^\top(\boldsymbol{\mu} - \boldsymbol{U}\boldsymbol{\phi})\right\} \\
&\quad \times \exp\left\{-\frac{1}{2}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)^\top\boldsymbol{\Phi}_0^{-1}(\boldsymbol{\phi} - \boldsymbol{\phi}_0)\right\} \\
&\propto \exp\left\{-\frac{1}{2}[\boldsymbol{\phi}^\top(\sigma^{-2}\boldsymbol{U}^\top\boldsymbol{U} + \boldsymbol{\Phi}_0^{-1})\boldsymbol{\phi} - 2\boldsymbol{\phi}^\top(\sigma^{-2}\boldsymbol{U}^\top\boldsymbol{\mu} + \boldsymbol{\Phi}_0^{-1}\boldsymbol{\phi}_0) \right. \\
&\quad \left. + \boldsymbol{\phi}_n^\top\boldsymbol{\Phi}_n^{-1}\boldsymbol{\phi}_n - \boldsymbol{\phi}_n^\top\boldsymbol{\Phi}_n^{-1}\boldsymbol{\phi}_n]\right\} \\
&\propto \exp\left\{-\frac{1}{2}[(\boldsymbol{\phi} - \boldsymbol{\phi}_n)^\top\boldsymbol{\Phi}_n^{-1}(\boldsymbol{\phi} - \boldsymbol{\phi}_n)]\right\},
\end{aligned}$$

where $\boldsymbol{\Phi}_n = (\sigma^{-2}\boldsymbol{U}^\top\boldsymbol{U} + \boldsymbol{\Phi}_0^{-1})^{-1}$ and $\boldsymbol{\phi}_n = \boldsymbol{\Phi}_n(\sigma^{-2}\boldsymbol{U}^\top\boldsymbol{\mu} + \boldsymbol{\Phi}_0^{-1}\boldsymbol{\phi}_0)$. The

last expression is the kernel of a multivariate normal distribution with mean $\boldsymbol{\phi}_n$ and variance matrix $\boldsymbol{\Phi}_n$.

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 $Var[Y_t] = \frac{\sigma^2(1-\phi_2)}{1-\phi_2-\phi_1^2-\phi_1^2\phi_2-\phi_2^2+\phi_2^3}$.

   Answer

   A necessary condition to achieve second-order stationarity in the $AR(2)$ process is that roots of of the polynomial $1 - \phi_1 L - \phi_2 L^2 = 0$ lie outside the unit circle. This is achieved if $|\phi_2| < 1$, $\phi_1 + \phi_2 < 1$ and $\phi_2 - \phi_1 < 1$.

   Let's calculate the expected value:

$$\begin{aligned}\mathbb{E}[Y_t] &= \bar{\mu} \\ &= \mathbb{E}[\mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \epsilon_t] \\ &= \mu + \phi_1 \mathbb{E}[Y_{t-1}] + \phi_2 \mathbb{E}[Y_{t-2}] + \mathbb{E}[\epsilon_t] \\ &= \mu + \phi_1 \bar{\mu} + \phi_2 \bar{\mu},\end{aligned}$$

   then, $\bar{\mu}(1 - \phi_1 - \phi_2) = \mu$, then, $\mathbb{E}[Y_t] = \bar{\mu} = \frac{\mu}{1-\phi_1-\phi_2}$.

   See that $Y_t = \mu + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \epsilon_t$ implies that $Y_t = \bar{\mu}(1 - \phi_1 - \phi_2) + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \epsilon_t$, which means that $Y_t - \bar{\mu} = \phi_1(Y_{t-1} - \bar{\mu}) + \phi_2(Y_{t-2} - \bar{\mu}) + \epsilon_t$, and setting $z_t = Y_t - \bar{\mu}$, we have $z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \epsilon_t$ where $\mathbb{E}[z_t] = 0$. Then, we can calculate the variance of Y_t using z_t, given that both have same variance.

   Let's define $\gamma_s = Cov(z_t, z_{t-s}) = \mathbb{E}[z_t z_{t-s}]$, then the Yule-Walker equations are:

$$\begin{aligned}\gamma_0 &= \phi_1\gamma_1 + \phi_2\gamma_2 + \sigma^2 \\ \gamma_1 &= \phi_1\gamma_0 + \phi_2\gamma_1 \\ \gamma_2 &= \phi_1\gamma_1 + \phi_2\gamma_0.\end{aligned}$$

   Solving this system of equations for γ_0, γ_1 and γ_2, we find that $\gamma_0 = \frac{\sigma^2(1-\phi_2)}{1-\phi_2-\phi_1^2-\phi_1^2\phi_2-\phi_2^2+\phi_2^3}$.

6. Program a Hamiltonian Monte Carlo (HMC) taking into account the stationary restrictions on ϕ_1 and ϕ_2, and ϵ_0 such that the acceptance rate is near 65%.

   Answer

   We know from the previous exercise that $|\phi_2| < 1$, $\phi_1+\phi_2 < 1$ and $\phi_2-\phi_1 < 1$ satisfied the stationary condition of the process. Thus, we should impose this restrictions in the HMC. The following code does this.

R code. AR(2) model: Hamiltonian Monte Carlo with stationary restrictions

```
# 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. AR(2) model: Hamiltonian Monte Carlo with stationary restrictions

```
StatRest <- function(phi1, phi2){
  if(abs(phi2) < 1 & phi1 + phi2 < 1 & phi2 - phi1 < 1){
    check <- 1
  }else{
    check <- 0
  }
  return(check)
}
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 == 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 & StatRest(phi1 = theta[2], phi2 = theta[3])
     == 1){
    thetaNew <- c(theta)
  }else{
    thetaNew <- thetat
  }
  rest <- list(theta = thetaNew, Prob = alpha)
  return(rest)
}
```

R code. AR(2) model: Hamiltonian Monte Carlo with stationary restrictions

```
# Posterior draws
S <- 1000; burnin <- 1000; thin <- 2; tot <- S + burnin
thetaPost <- matrix(NA, tot, K)
ProbAcept <- rep(NA, tot)
# theta0 <- theta0
theta0 <- c(mean(y), 0, 0, exp(var(y)))
M <- solve(VarPost); epsilon0 <- 0.4
pb <- txtProgressBar(min = 0, max = tot, style = 3)
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
  ProbAcept[s] <- HMCs$Prob
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
thetaF <- coda::mcmc(thetaPost[keep,])
summary(thetaF)
plot(thetaF)
summary(exp(thetaF[,K]))
plot(exp(thetaF[,K]))
ProbAceptF <- coda::mcmc(ProbAcept[keep])
summary(ProbAceptF)
```

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 number of particles.

Answer

We observe that, initially, sequential importance sampling performs relatively well. However, particle degeneracy occurs around ($t = 50$), causing the algorithm's performance to deteriorate rapidly (see Figure 8.3). We can see in Figure 8.4 how the effective sample size (ESS) of the particles decreases rapidly.

R code. Stochastic volatility model: Sequential importance sampling

```
rm(list = ls()); set.seed(010101)
T <- 250; K <- 2; 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
particles <- matrix(NA, N, T)   # Particles
logalphas <- matrix(NA, N, T)   # Incremental importance
    weights
ESS <- rep(NA, T) # Effective sample size
particles[, 1] <- rnorm(N, mu, sigma / sqrt(1 - phi^2))
log_Weights[, 1] <- dnorm(y[1], 0, sd = exp(0.5*particles
    [,1]), log = TRUE)
Weights[, 1] <- exp(log_Weights[, 1])
WeightsST[, 1] <- Weights[, 1] / sum(Weights[, 1])
ESS[1] <- (sum(WeightsST[, 1]^2))^(-1)
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)
  log_Weights[, t] <- log_Weights[, t - 1] + logalphas[, t]
  Weights[, t] <- exp(log_Weights[, t])
  WeightsST[, t] <- Weights[, t] / sum(Weights[, t])
  ESS[t] <- (sum(WeightsST[, t]^2))^(-1)
}
FilterDist <- colSums(particles * WeightsST)
SDFilterDist <- (colSums(particles^2 * WeightsST) -
    FilterDist^2)^0.5
MargLik <- colMeans(Weights)
library(dplyr); library(ggplot2); require(latex2exp)
ggplot2::theme_set(theme_bw())
df <- tibble(t = seq(1, T), mean = FilterDist, lower =
    FilterDist - 1.96*SDFilterDist, upper = FilterDist +
    1.96*SDFilterDist, x_true = h)
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)
plot(ESS, type = "l", ylab = "Effective sample size", xlab =
     "Time")
```

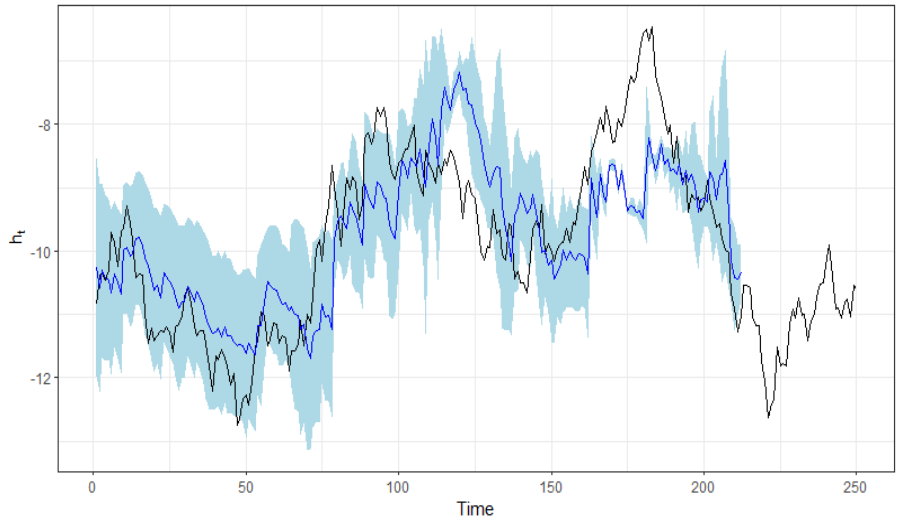

FIGURE 8.3
Stochastic volatility model: Sequential importance sampling (SIS).

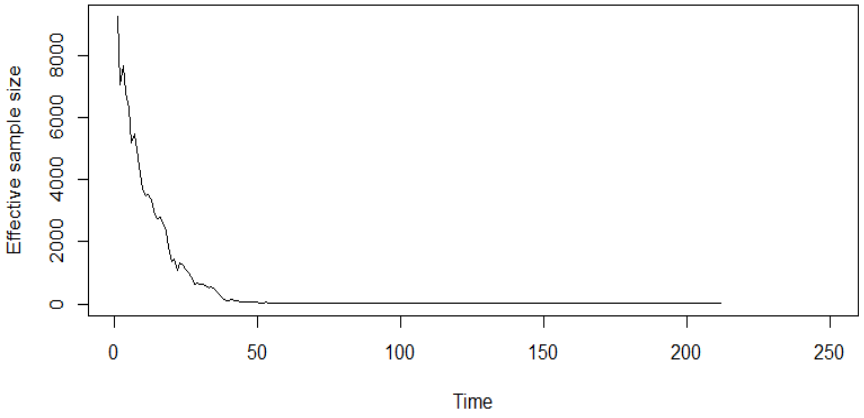

FIGURE 8.4
Stochastic volatility model: Effective sample size (ESS) of the particles.

This code shows how to program the SMC algorithm performing multinomial resampling when the effective sample size es lower than 50% the initial number of particles. It seems that in this example is better to perform re-

sampling every time period as the mean recursion (blue and purple lines) are not good following in state (black line). However, the bands corresponding to plus/minus two standard deviations (light blue shading) encompass the state vector; although, it is too wide.

R code. Stochastic volatility model: Sequential Monte Carlo

```
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 import. weig.
ESS <- rep(NA, T) # Efective sample size
cutoff <- 0.5
particles[, 1] <- rnorm(N, mu, sigma / sqrt(1 - phi^2))
log_Weights[, 1] <- dnorm(y[1], 0, sd = exp(0.5*particles
    [,1]), log = TRUE)
Weights[, 1] <- exp(log_Weights[, 1])
WeightsST[, 1] <- Weights[, 1] / sum(Weights[, 1])
ESS[1] <- (sum(WeightsST[, 1]^2))^(-1)
ind <- sample(1:N, size = N, replace = TRUE, prob =
    WeightsST[, 1]) # Resample
particles[, 1] <- particles[, 1] # Resampled particles
particlesT[, 1] <- particles[ind, 1] # Resampled particles
WeightsST[, 1] <- rep(1/N, N) # Resampled weights
pb <- txtProgressBar(min = 0, max = T, style = 3)
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])
  ESS[t] <- (sum(WeightsST[, t]^2))^(-1)
  particlesT[, t] <- particles[, t]
  if(ESS[t] < N*cutoff){
    ind <- sample(1:N, size = N, replace = TRUE, prob =
    WeightsST[, t])
    particles[, 1:t] <- particles[ind, 1:t]
    WeightsST[, t] <- 1/N
  }else{if(t == T & ESS[t] < N*cutoff)
    ind <- sample(1:N, size = N, replace = TRUE, prob =
    WeightsST[, t])
    particlesT[, 1:t] <- particles[ind, 1:t]
  }
  setTxtProgressBar(pb, t)
}
close(pb)
```

R code. Stochastic volatility model: Sequential Monte Carlo

```
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")
plot(ESS, 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)
```

8. Estimate the vanilla stochastic volatility model using the dataset *17ExcRate.csv* provided by [36] of the exchange rate log daily returns from USD/EUR, USD/GBP and GBP/EUR one year before and after the WHO declared the COVID-19 pandemic on 11 March 2020.

Answer

The following code shows the results of modeling the exchange rate log

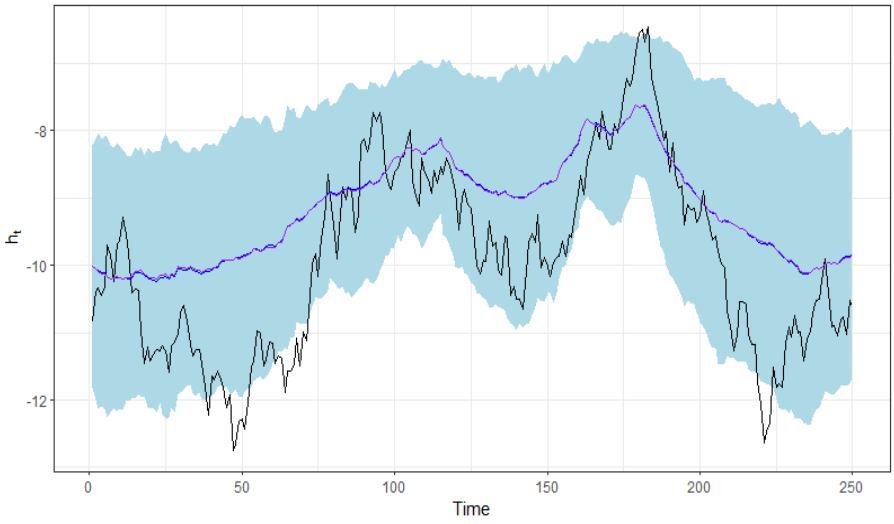

FIGURE 8.5
Stochastic volatility model: Sequential Monte Carlo (SMC).

daily returns from USD/EUR using a stochastic volatility model. Similar code can be used to estimate the other two models.

R code. Stochastic volatility model: Exchange rate log daily returns from USD/EUR

```
rm(list = ls()); set.seed(010101)
DataExcRate <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/17ExcRate.csv
    ", sep = ",", header = TRUE, quote = "")
attach(DataExcRate)
MCMC <- 10000; burnin <- 10000; thin <- 5
y <- USDEUR - mean(USDEUR)
plot(y, type = "l")
res <- stochvol::svsample(y, 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)])
Iterations = 10005:20000
Thinning interval = 5
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
mu     -11.2886 0.28916 0.0064657       0.006466
phi      0.9618 0.02202 0.0004923       0.001421
sigma    0.1664 0.04501 0.0010064       0.003722
2. Quantiles for each variable:
         2.5%      25%      50%      75%    97.5%
mu     -11.81067 -11.4287 -11.2902 -11.1448 -10.7699
phi      0.90719   0.9512   0.9657   0.9774   0.9913
sigma    0.09615   0.1327   0.1607   0.1926   0.2666
plot(res)
ht <- res[["latent"]][[1]]
library(dplyr)
library(ggplot2)
require(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 = 1:length(y),
mean = x_means,
lower = x_quantiles[1, ],
upper = x_quantiles[2, ])
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 = mean), colour = "blue", linewidth = 0.5)
     +
  ylab(TeX("$h_{t}$")) + xlab("Time")
  print(p)
}
plot_filtering_estimates(df)
```

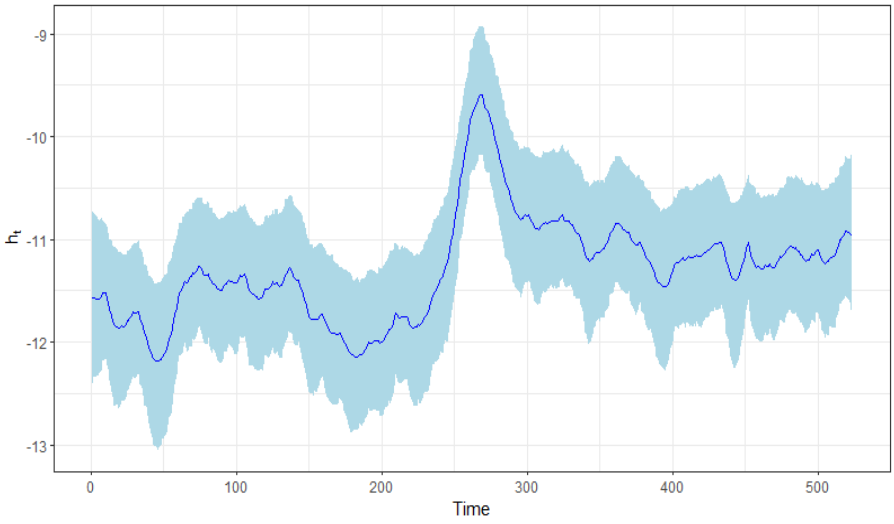

FIGURE 8.6
Stochastic volatility model: Exchange rate log daily returns from USD/EUR.

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*.

Answer

Using the recursion $\boldsymbol{\Phi}_s = \sum_{l=1}^{s} \boldsymbol{\Phi}_{s-l}\boldsymbol{A}_l$, $\boldsymbol{A}_l = \mathbf{0}$, $l > p$ and $s = 1, 2, ..,$ we get $\boldsymbol{\Phi}_s = \boldsymbol{A}_1^s$. In addition, we know that $\boldsymbol{\Theta}_s = \boldsymbol{\Phi}_S\boldsymbol{P}$. Then, we can get these impulse response functions.

The following code shows how to perform from scratch the impulse response

functions. Figure 8.7 shows the *forecast error impulse response* of y_{2t} to shocks in μ_{3t}. We see that the 95% (blue shadow) and 90% (light blue shadow) credible intervals of the impulse response functions embrace the true impulse response (black line); although the mean impulse response (blue line) overestimates the true impulse. In addition, we can see that the results from the *bvartools* package have the same patterns compared to the ones that we get programming the algorithm from scratch. Remember that the *bvartools* package uses the inverse Wishart distribution as prior for $\mathbf{\Sigma}$, where the hyperparameters are the degrees of freedom of the error term, and the prior error variance of endogenous variables.

R code. Vector autoregressive model: Simulation

```
rm(list = ls()); set.seed(010101)
T <- 300; M <- 3
SIGMA <- matrix(c(2.25, 0, 0, 0, 1, 0.5, 0, 0.5, 0.74), M, M
    )
U <- MASS::mvrnorm(T, rep(0, M), SIGMA)
A <- matrix(c(0.5, 0.1, 0, 0, 0.1, 0.2, 0, 0.3, 0.3), M, M )
v <- c(1.4, 1.9, 1.1); Y <- matrix(NA, T, M)
Y[1, ] <- solve(diag(M)-A)%*%v + U[1,]
for(t in 2:T){
  Y[t, ] <- v + A%*%Y[t-1,] + U[t,]
}
plot(Y[,1], type = "l"); plot(Y[,2], type = "l"); plot(Y
    [,3], type = "l")
y1 <- Y[-1,1]; y2 <- Y[-1,2]; y3 <- Y[-1,3]
X1 <- cbind(1, lag(Y)); X1 <- X1[-1,]
X2 <- cbind(1, lag(Y)); X2 <- X2[-1,]
X3 <- cbind(1, lag(Y)); X3 <- X3[-1,]
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
MCMC <- 10000; thin <- 5; burnin <- 1000; tot <- MCMC +
    burnin
Mcmc <- list(R = tot, keep = thin)
PosteriorDraws <- bayesm::rsurGibbs(Data = list(regdata =
    regdata), Mcmc = Mcmc, Prior = Prior)
keep <- seq(round(burnin/thin)+1, round(tot/thin))
Bs <- PosteriorDraws[["betadraw"]][keep,]
SIGMAs <- PosteriorDraws[["Sigmadraw"]][keep,]
S <- dim(Bs)[1]
As <- array(NA, c(M, M, S))
for(s in 1:S){
  As[,,s] <- matrix(Bs[s, -c(1, 5, 9)], M, M, byrow = TRUE)
}
# Impulse response functions
H <- 10; Ptrue <- t(chol(SIGMA))
Phistrue <-  array(NA, c(M, M, H))
Phistrue[,,1] <- diag(M); PhistrueAccum <-  Phistrue
PhistrueAccum[,,1] <- diag(M)
Thetastrue <- Phistrue; Thetastrue[,,1] <- Ptrue
ThetastrueAccum <-  Thetastrue; ThetastrueAccum[,,1] <-
    Ptrue
for(h in 2:H){
  Phistrue[,,h] <- Phistrue[,,h-1]%*%A
  PhistrueAccum[,,h] <- PhistrueAccum[,,h-1] + Phistrue[,,h]
  Thetastrue[,,h] <- Phistrue[,,h]%*%Ptrue
  PhistrueAccum[,,h] <- ThetastrueAccum[,,h-1] + Thetastrue
    [,,h]
}
P <- array(NA, c(M, M, S))
```

R code. Vector autoregressive model: Simulation

```
for(s in 1:S){
  P[,,s] <- t(chol(matrix(SIGMAs[s,], M, M)))
}
Phis <- list(); PhisAcum <- list()
Thetas <- list(); ThetasAcum <- list()
for(h in 0:H){
  if(h == 0){
    Phis[[h+1]] <- array(diag(M), c(M,M,S))
    Thetas <- Phis
    for(s in 1:S){
      Thetas[[h+1]][,,s] <- Phis[[h+1]][,,s]%*%P[,,s]
    }
    PhisAcum[[h+1]] <- Phis[[h+1]]
    ThetasAcum[[h+1]] <- Thetas[[h+1]]
  }else{
    Phis[[h+1]] <- array(diag(M), c(M,M,S))
    Thetas[[h+1]] <- Phis[[h+1]]
    for(s in 1:S){
      Phis[[h+1]][,,s] <- Phis[[h]][,,s]%*%As[,,s]
      Thetas[[h+1]][,,s] <- Phis[[h+1]][,,s]%*%P[,,s]
    }
    PhisAcum[[h+1]] <- PhisAcum[[h]] + Phis[[h+1]]
    ThetasAcum[[h+1]] <- ThetasAcum[[h]] + Thetas[[h+1]]
  }
}
library(dplyr); library(ggplot2); require(latex2exp)
ggplot2::theme_set(theme_bw())
plot_filtering_estimates <- function(df) {
  p <- ggplot(data = df, 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 = x_true), colour = "black", alpha = 1,
  linewidth = 0.5) +
  geom_line(aes(y = mean), colour = "blue", linewidth = 0.5)
     +
  ylab("Impulse response") + xlab("Time") + xlim(0,H-1)
  print(p)
}
```

R code. Vector autoregressive model: Simulation

```
IR <- function(m, j, Accumulated = "FALSE", Type = "Ordinary"){
  if(Type == "Cholesky"){
    IRtrue <- Thetastrue[m,j,]
    if(Accumulated == FALSE){
      RES <- Thetas; IRtrue <- IRtrue
    }else{
      RES <- ThetasAcum; IRtrue <- cumsum(IRtrue)
    }
    Mean <- sapply(1:H, function(h){mean(RES[[h]][m,j,])})
    LimInf1 <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.025)})
    LimSup1 <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.975)})
    LimInf <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.05)})
    LimSup <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.95)})
  }else{
    IRtrue <- Phistrue[m,j,]
    if(Accumulated == FALSE){
      RES <- Phis; IRtrue <- IRtrue
    }else{
      RES <- PhisAcum; IRtrue <- cumsum(IRtrue)
    }
    Mean <- sapply(1:H, function(h){mean(RES[[h]][m,j,])})
    LimInf1 <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.025)})
    LimSup1 <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.975)})
    LimInf <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.05)})
    LimSup <- sapply(1:H, function(h){quantile(RES[[h]][m,j,], 0.95)})
  }
  df <- tibble(t = 0:(H-1), mean = Mean, lower1 = LimInf1,
    upper1 = LimSup1, lower = LimInf, upper = LimSup, x_true = IRtrue)
  Fig <- plot_filtering_estimates(df)
  return(Fig)
}
m <- 2; j <- 3
IR(m,j, Accumulated = "FALSE", Type = "Cholesky")
Ynew <- ts(Y)
model <- bvartools::gen_var(Ynew, p = 1, deterministic = "const", iterations = MCMC, burnin = burnin) # Model
model <- bvartools::add_priors(model, coef = list(v_i = b0[1], v_i_det = b0[1]), sigma = list(df = a0, scale = V[1,1]/a0)) # Prior
object <- bvartools::draw_posterior(model) # Pposterior
ir <- bvartools::irf.bvar(object, impulse = "Series 2", response = "Series 3") # Calculate IR
plot(ir) # Plot IR
```

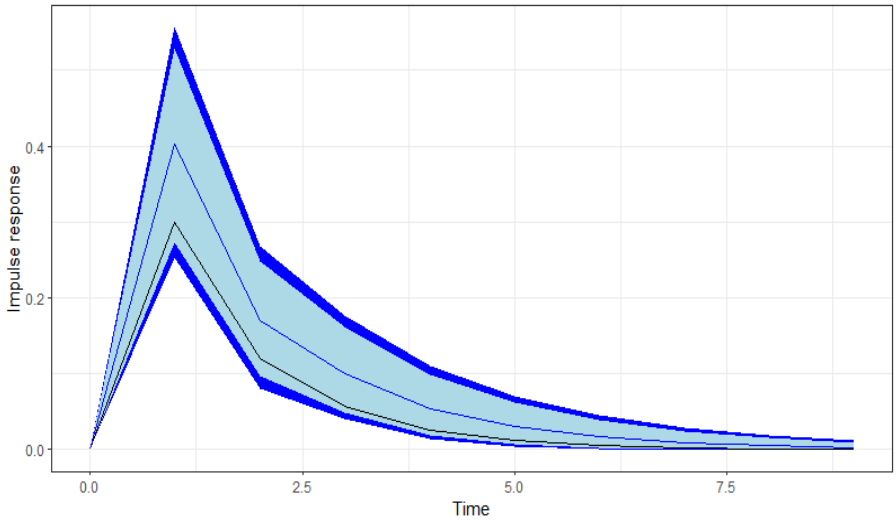

FIGURE 8.7
Forecast error impulse response: Response in y_{2t} to shock in μ_{3t}.

9

Longitudinal/Panel data models

Solutions of Exercises

1. Show that the posterior distribution of $\boldsymbol{\beta}|\sigma^2, \boldsymbol{D}$ is $N(\boldsymbol{\beta}_n, \boldsymbol{B}_n)$, where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{X}_i)^{-1}$ and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{y}_i)$.

   Answer

$$\begin{aligned}
\pi(\boldsymbol{\beta}|\sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) &\propto \exp\left\{-\frac{1}{2}\sum_{i=1}^N (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta})^\top \boldsymbol{V}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta})\right\} \\
&\quad \times \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left(-2\boldsymbol{\beta}^\top\left(\sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1}\boldsymbol{y}_i + \boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0\right) + \boldsymbol{\beta}^\top\left(\sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1}\boldsymbol{X}_i + \boldsymbol{B}_0^{-1}\right)\boldsymbol{\beta}\right)\right\} \\
&= \exp\left\{-\frac{1}{2}(-2\boldsymbol{\beta}^\top \boldsymbol{B}_n^{-1}\boldsymbol{B}_n\left(\sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1}\boldsymbol{y}_i + \boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0\right) + \boldsymbol{\beta}^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta})\right\} \\
&= \exp\left\{-\frac{1}{2}(-2\boldsymbol{\beta}^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{\beta}^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta})\right\}.
\end{aligned}$$

   We can complete the square in this expression by adding and subtracting $\boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n$. Thus,

$$\begin{aligned}
\pi(\boldsymbol{\beta}|\sigma^2, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) &\propto \exp\left\{-\frac{1}{2}(-2\boldsymbol{\beta}^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n + \boldsymbol{\beta}^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta} + \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n - \boldsymbol{\beta}_n^\top \boldsymbol{B}_n^{-1}\boldsymbol{\beta}_n)\right\} \\
&\propto \exp\left\{-\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)\right\}.
\end{aligned}$$

   This is the kernel of a multivariate random variable with mean $\boldsymbol{\beta}_n$ and variance matrix $\boldsymbol{B}_n$.

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 $\boldsymbol{\beta}_0 = \mathbf{0}_5$, $\boldsymbol{B}_0 = \boldsymbol{I}_5$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 5$ and $\boldsymbol{D}_0 = \boldsymbol{I}_1$.
- Perform inference in this example assuming that $\mu_{it}|\tau_{it} \sim N(0, \sigma^2/\tau_{it})$ and $\tau_{it} \sim G(v/2, v/2)$ setting $v = 5$.

Answer

R code. The relationship between productivity and public investment, programming from scratch the Gibbs sampler

```r
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)
N <- length(unique(id))
y <- log(gsp)
NT <- length(y)
X <- cbind(1, log(pcap), log(pc), log(emp), unemp)
K1 <- dim(X)[2]
W <- matrix(rep(1, NT), NT, 1)
K2 <- dim(W)[2]
mcmc <- 10000; burnin <- 5000; thin <- 1; tot <- mcmc +
    burnin
b0 <- rep(0, K1); B0 <- diag(K1); B0i <- solve(B0)
r0 <- 5; R0 <- diag(K2); a0 <- 0.001; d0 <- 0.001
PostBeta <- function(sig2, D){
  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 <- sig2*diag(Ti) + 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. The relationship between productivity and public investment, programming from scratch the Gibbs sampler

```
Postb <- function(Beta, sig2, D){
  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]
    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(as.matrix(bis))
}
PostSig2 <- function(Beta, bs){
  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, ]
    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)
}
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)
}
```

R code. The relationship between productivity and public investment, programming from scratch the Gibbs sampler

```
PostBetas <- matrix(0, tot, K1); PostDs <- matrix(0, tot, K2
    *(K2+1)/2)
PostSig2s <- rep(0, tot); Postbs <- array(0, c(N, K2, tot))
RegLS <- lm(y ~ X - 1); SumLS <- summary(RegLS)
Beta <- SumLS[["coefficients"]][,1]
sig2 <- SumLS[["sigma"]]^2; D <- diag(K2)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  bs <- Postb(Beta = Beta, sig2 = sig2, D = D)
  D <- PostD(bs = bs)
  Beta <- PostBeta(sig2 = sig2, D = D)
  # Beta <- PostBetaNew(sig2 = sig2, D = D)
  sig2 <- PostSig2(Beta = Beta, bs = bs)
  PostBetas[s,] <- Beta
  PostDs[s,] <- matrixcalc::vech(D)
  PostSig2s[s] <- sig2
  Postbs[, , s] <- bs
  setTxtProgressBar(pb, s)
}
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  1.727227  1.934457  2.037411  2.138829  2.325961
var2 -0.033664  0.001015  0.018974  0.037977  0.076626
var3  0.274521  0.303619  0.318302  0.332881  0.363323
var4  0.652937  0.692479  0.712605  0.730885  0.768415
var5 -0.008724 -0.007265 -0.006548 -0.005844 -0.004508
summary(coda::mcmc(Ds))
summary(coda::mcmc(sig2s))
# Convergence diagnostics
coda::geweke.diag(Bs)
coda::raftery.diag(Bs,q=0.5,r=0.05,s = 0.95)
coda::heidel.diag(Bs)
```

The likelihood function in the case with heteroskedasticity is proportional

to

$$p(\boldsymbol{\beta}, \boldsymbol{b}, \sigma^2, \boldsymbol{\tau} | \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) \propto \prod_{i=1}^{N} |\sigma^2 \boldsymbol{\Psi}_i|^{-1/2} \times \exp\left\{ -\frac{1}{2\sigma^2} (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top \boldsymbol{\Psi}_i^{-1} (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i) \right\},$$

where $\boldsymbol{b} = [\boldsymbol{b}_1^\top, \boldsymbol{b}_2^\top, \ldots, \boldsymbol{b}_N^\top]^\top$, $\boldsymbol{\tau} = [\tau_{it}]^\top$ and $\boldsymbol{\Psi}_i = diag\left\{\tau_{it}^{-1}\right\}$.

Following the same procedure as in Section 9.1 of the book, and Exercise 1 of this chapter, we have that

$$\boldsymbol{\beta} | \sigma^2, \boldsymbol{\tau}, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^{N} \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{X}_i)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^{N} \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{y}_i)$ and $\boldsymbol{V}_i = \sigma^2 \boldsymbol{\Psi}_i + \sigma_b^2 \boldsymbol{i}_{T_i} \boldsymbol{i}_{T_i}^\top$.

$$\boldsymbol{b}_i | \boldsymbol{\beta}, \sigma^2, \boldsymbol{\tau}, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim N(\boldsymbol{b}_{ni}, \boldsymbol{B}_{ni}),$$

where $\boldsymbol{B}_{ni} = (\sigma^{-2} \boldsymbol{W}_i^\top \boldsymbol{\Psi}_i^{-1} \boldsymbol{W}_i + \boldsymbol{D}^{-1})^{-1}$ and $\boldsymbol{b}_{ni} = \boldsymbol{B}_{ni}(\sigma^{-2} \boldsymbol{W}_i^\top \boldsymbol{\Psi}_i^{-1} (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta}))$.

$$\sigma^2 | \boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim IG(\alpha_n, \delta_n),$$

where $\alpha_n = \alpha_0 + \frac{1}{2}\sum_{i=1}^{N} T_i$ and $\delta_n = \delta_0 + \frac{1}{2}\sum_{i=1}^{N} (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top \boldsymbol{\Psi}_i^{-1} (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)$.

$$\boldsymbol{D} | \boldsymbol{b} \sim IW(d_n, \boldsymbol{D}_n),$$

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

$$\tau_{it} | \sigma^2, \boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim G(v_{1n}/2, v_{2ni}/2),$$

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 shows to implement this for the productivity application. The results of the *fixed effects* are very similar compared to the results without taken into account heteroskedasticity.

R code. The relationship between productivity and public investment, programming from scratch the Gibbs sampler with heteroskedasticity

```
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)
N <- length(unique(id))
y <- log(gsp)
NT <- length(y)
X <- cbind(1, log(pcap), log(pc), log(emp), unemp)
K1 <- dim(X)[2]
W <- matrix(rep(1, NT), NT, 1)
K2 <- dim(W)[2]
mcmc <- 10000; burnin <- 5000; thin <- 1; tot <- mcmc +
    burnin
b0 <- rep(0, K1); B0 <- diag(K1); B0i <- solve(B0)
r0 <- 5; R0 <- diag(K2); a0 <- 0.001; d0 <- 0.001; v <- 5
# Gibbs by hand
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. The relationship between productivity and public investment, programming from scratch the Gibbs sampler with heteroskedasticity

```
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. The relationship between productivity and public investment, programming from scratch the Gibbs sampler with heteroskedasticity

```
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 <- txtProgressBar(min = 0, max = tot, style = 3)
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
  setTxtProgressBar(pb, s)
}
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  1.61764  1.891242  2.029985  2.165677  2.427349
var2 -0.07088 -0.016921  0.013080  0.044681  0.107360
var3  0.27259  0.312197  0.333106  0.353678  0.395123
var4  0.59507  0.664490  0.699526  0.734612  0.796977
var5 -0.01046 -0.008533 -0.007586 -0.006666 -0.004918
summary(coda::mcmc(Ds))
summary(coda::mcmc(sig2s))
```

3. **Simulation exercise of the longitudinal normal model continues**

   Assume that $y_{it} = \beta_1 x_{it1} + \beta_2 x_{it2} + \beta_3 x_{it3} + \beta_4 z_{i1} + b_i + w_{it1} b_{i1} + \mu_{it}$ where $x_{itk} \sim N(0,1)$, $k = 1,2,3$, $z_{i1} \sim B(0.5)$, $w_{it1} \sim N(0,1)$, $b_i \sim N(0, 0.7^{1/2})$, $b_{i1} \sim N(0, 0.6^{1/2})$, $\mu_{it} \sim N(0, 0.1^{1/2})$ $\boldsymbol{\beta} = [0.4\ 0.6\ -0.6\ 0.7]^\top$,

$i = 1, 2, \ldots, 50$, and the sample size is 2000 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 $\boldsymbol{\Gamma}_0 = \boldsymbol{I}_4$. In addition, Set $\boldsymbol{\beta}_0 = \boldsymbol{0}_4$, $\boldsymbol{B}_0 = \boldsymbol{I}_4$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 2$ and $\boldsymbol{D}_0 = \boldsymbol{I}_2$.

- 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.

Answer

Following the same procedure as in Section 9.1 of the book, and Exercise 1 of this chapter, we have that

$$\boldsymbol{\beta}|\sigma^2, \boldsymbol{\gamma}, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}, \boldsymbol{Z} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1} \boldsymbol{X}_i)^{-1}$, $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^N \boldsymbol{X}_i^\top \boldsymbol{V}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{W}_i\boldsymbol{Z}_i\boldsymbol{\gamma}))$ and $\boldsymbol{V}_i = \sigma^2\boldsymbol{I}_{T_i} + \sigma_b^2 \boldsymbol{i}_{T_i}\boldsymbol{i}_{T_i}^\top$.

$$\boldsymbol{b}_i|\boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma}, \boldsymbol{D}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}, \boldsymbol{Z} \sim N(\boldsymbol{b}_{ni}, \boldsymbol{B}_{ni}),$$

where $\boldsymbol{B}_{ni} = (\sigma^{-2}\boldsymbol{W}_i^\top \boldsymbol{W}_i + \boldsymbol{D}^{-1})^{-1}$ and $\boldsymbol{b}_{ni} = \boldsymbol{B}_{ni}(\sigma^{-2}\boldsymbol{W}_i^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta}) + \boldsymbol{D}^{-1}\boldsymbol{Z}_i\boldsymbol{\gamma})$.

$$\sigma^2|\boldsymbol{\beta}, \boldsymbol{b}, \boldsymbol{\tau}, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W} \sim IG(\alpha_n, \delta_n),$$

where $\alpha_n = \alpha_0 + \frac{1}{2}\sum_{i=1}^N T_i$ and $\delta_n = \delta_0 + \frac{1}{2}\sum_{i=1}^N (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)^\top(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{W}_i\boldsymbol{b}_i)$.

$$\boldsymbol{D}|\boldsymbol{b}, \boldsymbol{\gamma}, \boldsymbol{Z} \sim IW(d_n, \boldsymbol{D}_n),$$

where $d_n = d_0 + N$ and $\boldsymbol{D}_n = d_0\boldsymbol{D}_0 + \sum_{i=1}^N (\boldsymbol{b}_i - \boldsymbol{Z}_i\boldsymbol{\gamma})(\boldsymbol{b}_i - \boldsymbol{Z}_i\boldsymbol{\gamma})^\top$. And

$$\boldsymbol{\gamma}|\boldsymbol{b}, \boldsymbol{D}, \boldsymbol{Z} \sim N(\boldsymbol{\gamma}_n, \boldsymbol{\Gamma}_n),$$

where $\boldsymbol{\Gamma}_n = (\sum_{i=1}^N \boldsymbol{Z}_i^\top \boldsymbol{D}^{-1}\boldsymbol{Z}_i + \boldsymbol{\Gamma}_0^{-1})^{-1}$ and $\boldsymbol{\gamma}_n = \boldsymbol{\Gamma}_n(\sum_{i=1}^N \boldsymbol{Z}_i^\top \boldsymbol{D}^{-1}\boldsymbol{b}_i + \boldsymbol{\Gamma}_0^{-1}\boldsymbol{\gamma}_0)$.

The following code shwos how to implement this longitudinal normal model where *random effects* are correlated with regressors. We set MCMC iterations, burn-in and thinning parameters equal to 15000, 5000 and 5, respectively. In addition, $\boldsymbol{\beta}_0 = \boldsymbol{0}_5$, $\boldsymbol{B}_0 = \boldsymbol{I}_5$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 2$, $\boldsymbol{D}_0 = \boldsymbol{I}_2$, $\boldsymbol{\gamma}_0 = \boldsymbol{0}_4$ and $\boldsymbol{\Gamma}_0 = \boldsymbol{I}_4$.

R code. Simulation exercise: Longitudinal normal model with fixed effects correlated to regressors

```
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); z1 <- rbinom(N, 1, 0.5)
zdata <- NULL
for(i in 1:NT){
  zdatai <- z1[id[i]]
  zdata <- c(zdata, zdatai)
}
X <- cbind(x1, x2, x3, zdata)
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, 0.7)
D <- c(0.7, 0.6)
sig2 <- 0.1
u <- rnorm(NT, 0, sd = sig2^0.5)
Z <- cbind(1, z1)
K3 <- dim(Z)[2]
G <- rep(1, K3*K2)
b <- matrix(0, N, K2)
for(i in 1:N){
  ZGi <- t(kronecker(diag(K2), Z[i, ]))%*%G
  b[i, ] <- MASS::mvrnorm(1, ZGi, diag(D))
}
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, zdata, w1, id))
mcmc <- 15000; burnin <- 5000; thin <- 10; tot <- mcmc +
    burnin
b0 <- rep(0, K1+1); B0 <- diag(K1+1); B0i <- solve(B0)
r0 <- K2; R0 <- diag(K2); a0 <- 0.001; d0 <- 0.001
g0 <- rep(0, K2*K3); G0 <- diag(K2*K3); G0i <- solve(G0)
Resultshreg <- MCMCpack::MCMChregress(fixed = y~x1 + x2 + x3
     + zdata, random = ~w1, group="id", data = Data, burnin
    = burnin, mcmc = mcmc, thin = thin,
mubeta = b0, Vbeta = B0, r = r0, R = R0, nu = a0, delta = d0
    )
Betas <- Resultshreg[["mcmc"]][,1:(K1+1)]
Sigma2RanEff <- Resultshreg[["mcmc"]][,c(K2*N+K1+2, 2*N+K1
    +1+K2^2)]
Sigma2 <- Resultshreg[["mcmc"]][,K2*N+K1+K2^2+2]
summary(Betas)
summary(Sigma2RanEff)
summary(Sigma2)
```

R code. Simulation exercise: Longitudinal normal model with fixed effects correlated to regressors

```
PostBeta <- function(sig2, Gamma, D){
  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 <- sig2*diag(Ti) + Wi%*%D%*%t(Wi)
    ViInv <- solve(Vi)
    Xi <- X[ids, ]
    XVXi <- t(Xi)%*%ViInv%*%Xi
    XVX <- XVX + XVXi
    Zi <- Z[i, ]
    yi <- y[ids]
    ZGi <- t(kronecker(diag(K2), Zi))%*%Gamma
    XVyi <- t(Xi)%*%ViInv%*%(yi - Wi%*%ZGi)
    XVy <- XVy + XVyi
  }
  Bn <- solve(B0i + XVX)
  bn <- Bn%*%(B0i%*%b0 + XVy)
  Beta <- MASS::mvrnorm(1, bn, Bn)
  return(Beta)
}
Postb <- function(Beta, sig2, Gamma, D){
  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]
    Wtei <- sig2^(-1)*t(Wi)%*%(yi - Xi%*%Beta)
    Bni <- solve(sig2^(-1)*t(Wi)%*%Wi + Di)
    Zi <- Z[i, ]
    ZGi <- t(kronecker(diag(K2), Zi))%*%Gamma
    bni <- Bni%*%(Wtei + Di%*%ZGi)
    bi <- MASS::mvrnorm(1, bni, Bni)
    bis[i, ] <- bi
  }
  return(bis)
}
PostSig2 <- function(Beta, bs){
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, ]
  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: Longitudinal normal model with fixed effects correlated to regressors

```
PostGamma <- function(bs, D){
  Di <- solve(D)
  ZDZ <- matrix(0, K2*K3, K2*K3)
  ZDb <- matrix(0, K2*K3, 1)
  for(i in 1:N){
    Zi <- Z[i, ]
    ZZi <- t(kronecker(diag(K2), Zi))
    ZDZi <- t(ZZi)%*%Di%*%ZZi
    ZDZ <- ZDZ + ZDZi
    bi <- bs[i,]
    ZDbi <- t(ZZi)%*%Di%*%bi
    ZDb <- ZDb + ZDbi
  }
  Gn <- solve(G0i + ZDZ); gn <- Gn%*%(G0i%*%g0 + ZDb)
  Gamma <- MASS::mvrnorm(1, gn, Gn)
  return(Gamma)
}
PostD <- function(bs, Gamma){
  rn <- r0 + N
  btb <- matrix(0, K2, K2)
  for(i in 1:N){
    bsi <- bs[i, ]
    Zi <- Z[i, ]
    ZGi <- t(kronecker(diag(K2), Zi))%*%Gamma
    btbi <- (bsi-ZGi)%*%t(bsi-ZGi)
    btb <- btb + btbi
  }
  Rn <- d0*R0 + btb
  Sigma <- MCMCpack::riwish(v = rn, S = Rn)
  return(Sigma)
}
PostBetas <- matrix(0, tot, K1); PostGammas <- matrix(0, tot
    , K2*K3)
PostDs <- matrix(0, tot, K2*(K2+1)/2); PostSig2s <- rep(0,
    tot)
Postbs <- array(0, c(N, K2, tot)); RegLS <- lm(y ~ X - 1)
SumLS <- summary(RegLS); Beta <- SumLS[["coefficients"]][,1]
sig2 <- SumLS[["sigma"]]^2; Gamma <- rep(0, K2*K3)
D <- diag(K2); b0 <- rep(0, K1); B0 <- diag(K1); B0i <-
    solve(B0)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  bs <- Postb(Beta = Beta, sig2 = sig2, Gamma = Gamma, D = D
    )
  D <- PostD(bs = bs, Gamma = Gamma)
  Beta <- PostBeta(sig2 = sig2, Gamma = Gamma, D = D)
  sig2 <- PostSig2(Beta = Beta, bs = bs)
  Gamma <- PostGamma(bs = bs, D = D)
  PostBetas[s,] <- Beta; PostDs[s,] <- matrixcalc::vech(D)
  PostSig2s[s] <- sig2; Postbs[, , s] <- bs
  PostGammas[s,] <- Gamma
  setTxtProgressBar(pb, s)
}
```

R code. Simulation exercise: Longitudinal normal model with fixed effects correlated to regressors

```
close(pb)
keep <- seq((burnin+1), tot, thin)
Bs <- PostBetas[keep,]
Ds <- PostDs[keep,]
bs <- Postbs[, , keep]
sig2s <- PostSig2s[keep]
Gs <- PostGammas[keep, ]
summary(coda::mcmc(Bs))
Quantiles for each variable:
      2.5%     25%     50%     75%   97.5%
var1  0.3629  0.3812  0.3880  0.3945  0.4120
var2  0.5844  0.6032  0.6098  0.6176  0.6381
var3 -0.6114 -0.5950 -0.5880 -0.5809 -0.5605
var4 -0.4455  0.4155  0.8598  1.3047  2.0654
summary(coda::mcmc(Ds))
Quantiles for each variable:
      2.5%     25%     50%      75%  97.5%
var1  0.5858  0.7798  0.9173  1.13381 2.0153
var2 -0.3914 -0.1901 -0.1104 -0.03585 0.1250
var3  0.4285  0.5495  0.6314  0.72342 0.9662
summary(coda::mcmc(sig2s))
Quantiles for each variable:
Quantiles for each variable:
2.5%     25%     50%     75%   97.5%
0.09609 0.11486 0.17790 0.35215 1.12537
summary(coda::mcmc(Gs))
Quantiles for each variable:
      2.5%    25%    50%   75% 97.5%
var1  0.6552 0.8998 1.0288 1.162 1.402
var2 -0.4463 0.4056 0.7882 1.242 2.144
var3  0.8583 1.0599 1.1623 1.271 1.482
var4  0.5525 0.8430 1.0038 1.147 1.416
```

In one hand, the 95% credible intervals of the model without taking into account the dependence between the *random effects* and z_{i1} overestimates the posterior mean of β_4 and the variance of b_{i1}. In addition, the credible intervals are very narrow. However, the variance of μ_i is well estimated. On the other hand, all the 95% credible intervals of the model taking into account the dependence between the *random effects* and z_{i1} encompass the population parameters. However, there is a lot of variability in β_4 posterior estimates. The posterior mean estimate of the variance of b_{i1} is close to the population value, and the location parameters associated with z_{i1} in the mean of the *random effects* are also well estimated by the posterior mean. However, the variance of the model is overestimated.

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.

   Answer

   This code replicates what we get using our GUI.

R code. Doctor visits in Germany, results

```
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 =
    0)
Betas <- Resultshlogit[["mcmc"]][,1:K1]
Sigma2RanEff <- Resultshlogit[["mcmc"]][,c(K2*N+K1+1, 2*N+K1
    +K2^2)]
summary(Betas)
summary(Sigma2RanEff)
summary(Sigma2)
Quantiles for each variable:
            2.5%        25%         50%         75%       97.5%
Intercept  -9.48e-01 -5.27e-01 -2.86e-01 -6.28e-02  2.99e-01
Age         4.88e-03  7.27e-03  8.35e-03  9.31e-03  1.09e-02
Male       -9.90e-01 -9.52e-01 -9.31e-01 -9.10e-01 -8.75e-01
Sport       1.80e-01  2.25e-01  2.55e-01  2.85e-01  3.47e-01
LogInc      1.38e-01  1.85e-01  2.15e-01  2.49e-01  3.01e-01
GoodHealth -9.56e-01 -9.18e-01 -8.97e-01 -8.75e-01 -8.36e-01
BadHealth   1.08e+00  1.14e+00  1.19e+00  1.23e+00  1.28e+00
D11         1.43e+00  1.53e+00  1.58e+00  1.62e+00  1.70e+00
D21        -2.67e-01  1.94e-02  1.85e-01  4.08e-01  5.81e-01
D12        -2.67e-01  1.94e-02  1.85e-01  4.08e-01  5.81e-01
D22         6.05e-01  7.99e-01  9.75e-01  1.21e+00  1.62e+00
sigma2      6.37e-02  7.76e-02  1.01e-01  1.15e-01  1.58e-01
```

We see in this example running our GUI that we get qualitatively same results as the results fixing the over-dispersion parameter in Section 9.2 in the book. However, there small numerical differences in the *fixed effects*,

and the differences are greater in the posterior results of the covariance matrix of the *random effects*.

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 1000 in an *unbalanced panel structure*.

   - Perform inference using the command $MCMChlogit$ fixing the over-dispersion parameter, and using $\boldsymbol{\beta}_0 = \mathbf{0}_4$, $\boldsymbol{B}_0 = \boldsymbol{I}_4$, $\alpha_0 = \delta_0 = 0.001$, $d_0 = 2$ and $\boldsymbol{D}_0 = \boldsymbol{I}_2$.
   - Program from scratch a Metropolis-within-Gibbs algorithm to perform inference in this simulation.

   Answer

   The following code shows the simulation setting, and the results using the *MCMChlogit* command. We see that all posterior estimates encompass the population values.

R code. Simulation exercise: Hierarchical longitudinal logit model using MCMChlogit

```
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)
}
pit <- 1/(1+exp(-yl))
y <- rbinom(NT, 1, prob = pit)
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
RegLogit <- glm(y ~ X - 1, family = binomial(link = "logit")
    )
SumLogit <- summary(RegLogit)

Beta0 <- SumLogit[["coefficients"]][,1]
sig20 <- sum(SumLogit[["deviance.resid"]]^2)/SumLogit[["df.
    residual"]]
Resultshlogit <- MCMCpack::MCMChlogit(fixed = y~x1 + x2 + x3
    , random = ~w1, 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, sigma2.start = sig20)
Betas <- Resultshlogit[["mcmc"]][,1:K1]
Sigma2RanEff <- Resultshlogit[["mcmc"]][,c(K2*N+K1+1, 2*N+K1
    +K2^2)]
Sigma2 <- Resultshlogit[["mcmc"]][,K2*N+K1+K2^2+1]
summary(Betas)
Quantiles for each variable:
                2.5%     25%     50%     75%   97.5%
beta.(Intercept)  0.4258  0.6165  0.7140  0.8155  1.0041
beta.x1           0.3294  0.4372  0.4999  0.5617  0.6684
beta.x2           0.5555  0.6763  0.7388  0.8062  0.9377
beta.x3          -0.9327 -0.7957 -0.7276 -0.6591 -0.5383
summary(Sigma2RanEff)
Quantiles for each variable:
2.5%    25%    50%    75% 97.5%
D11 0.3394 0.5366 0.6962 0.8652 1.338
D22 0.4101 0.6478 0.8091 1.0039 1.611
```

The following code shows the algorithm step by step.

R code. Simulation exercise: Hierarchical longitudinal logit model from scratch

```
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)
    pihati <- 1/(1+exp(-ylhati))
    ei <- rnorm(1, 0, sd = tuning)
    ylpropi <- ylhati + ei; pipropi <- 1/(1+exp(-ylpropi))
    logPosthati <- sum(dbinom(yi, 1, prob = pihati, log =
    TRUE) + dnorm(ylhati, ylhatmeani, sig2^0.5, log = TRUE))
    logPostpropi <- sum(dbinom(yi, 1, prob = pipropi, 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)
}
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)
}
```

R code. Simulation exercise: Hierarchical longitudinal logit model from scratch

```
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, ss){
  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)
  if(sig2 > ss){
    sig2 <- ss
  }else{
    sig2 <- sig2
  }
  return(sig2)
}
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)
RegLogit <- glm(y ~ X - 1, family = binomial(link = "logit")
    )
SumLogit <- summary(RegLogit)
Beta <- SumLogit[["coefficients"]][,1]
sig2 <- sum(SumLogit[["deviance.resid"]]^2)/SumLogit[["df.
    residual"]]
ss0 <- sig2; D <- diag(K2)
bs1 <- rnorm(N, 0, sd = D[1]^0.5)
bs2 <- rnorm(N, 0, sd = D[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 <- txtProgressBar(min = 0, max = tot, style = 3)
```

R code. Simulation exercise: Hierarchical longitudinal logit model from scratch

```
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, ss =
     ss0)
  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
  }
  setTxtProgressBar(pb, s)
}
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  0.3195  0.5266  0.6325  0.7452  0.9427
var2  0.1780  0.3563  0.4521  0.5436  0.7352
var3  0.3596  0.5197  0.6140  0.7165  0.9149
var4 -0.8994 -0.7153 -0.6125 -0.5135 -0.3240
summary(coda::mcmc(Ds))
Quantiles for each variable:
      2.5%         25%         50%        75%   97.5%
var1  0.0002003  0.0007473  0.0022547 0.0124378 0.09575
var2 -0.0534437 -0.0029207 -0.0001433 0.0009905 0.02296
var3  0.0002150  0.0009101  0.0026712 0.0119391 0.14863
summary(coda::mcmc(sig2s))
Quantiles for each variable:
2.5%    25%    50%    75%  97.5%
0.1321 0.5174 0.8674 1.0788 1.1909
```

The 95% credible intervals of the *fixed effects* encompass all the population values. However, we do not get good posterior estimates of the covariance matrix and over-dispersion parameters.

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.

   Answer

   This algorithm shows how to perform inference in this application programming from scratch the algorithm.

R code. Doctor visits in Germany: Hierarchical longitudinal Poisson model from scratch

```
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 = "")
library(dplyr)
Data <- Data %>%
filter(id <= 500)
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
NT <- dim(Data)[1]
N <- length(unique(id))
X <- cbind(1, Age, Male, Sport, LogInc, GoodHealth, BadHealth)
K1 <- dim(X)[2]
W <- cbind(1, Sozh)
K2 <- dim(W)[2]
y <- DocNum
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. Doctor visits in Germany: Hierarchical longitudinal Poisson model from scratch

```
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 <- matrix(W[ids, ], Ti, K2)
    Vi <- diag(Ti)*sig2 + Wi%*%D%*%t(Wi)
    ViInv <- solve(Vi); Xi <- matrix(X[ids, ], Ti, K1)
    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); Ti <- length(ids)
    Wi <- matrix(W[ids, ], Ti, K2)
    Xi <- matrix(X[ids, ], Ti, K1)
    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)
    Ti <- length(ids)
    Xi <- matrix(X[ids, ], Ti, K1); yi <- ylhat[ids]
    Wi <- matrix(W[ids, ], Ti, K2)
    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. Doctor visits in Germany: Hierarchical longitudinal Poisson model from scratch

```
PostBetas <- matrix(0, tot, K1)
PostDs <- matrix(0, tot, K2*(K2+1)/2)
Postbs <- array(0, c(N, K2, tot))
PostSig2s <- rep(0, tot); ccepts <- rep(NULL, tot)
RegPois <- glm(DocNum ~ Age + Male + Sport + LogInc +
    GoodHealth + BadHealth, 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 <- txtProgressBar(min = 0, max = tot, style = 3)
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
  }
  setTxtProgressBar(pb, s)
}
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 -1.250036 -0.269211  0.1995811  0.69139  1.59234
var2 -0.001542  0.004852  0.0083935  0.01147  0.01815
var3 -0.484097 -0.331748 -0.2586891 -0.19011 -0.04787
var4 -0.023677  0.123336  0.1973927  0.26883  0.41314
var5 -0.196732 -0.071421 -0.0001036  0.06601  0.19356
var6 -0.815499 -0.657182 -0.5784964 -0.49969 -0.36882
var7  0.613990  0.799534  0.8982645  1.00161  1.18880
```

We see that males with good perception of health status visit less frequently the physicians, and women with bad perception of health status visit more frequently the physicians.

10

Bayesian model average

Solutions of Exercises

1. The Gaussian linear model specifies $\mathbf{y} = \alpha\mathbf{i_N} + \mathbf{X_m}\boldsymbol{\beta_m} + \boldsymbol{\mu_m}$ 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 conditional posterior 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 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.

Answer

The marginal likelihood of this model is

$$\begin{aligned}
p(\mathbf{y}) &= \int_0^\infty\int_{R^K}\int_R \pi(\boldsymbol{\beta}|\sigma^2)\pi(\sigma^2)\pi(\alpha)p(\mathbf{y}|\boldsymbol{\beta},\sigma^2,\alpha)d\alpha d\boldsymbol{\beta}d\sigma^2 \\
&\propto \int_0^\infty\int_{R^K}\int_R (\sigma^2)^{-k_m/2}|g_m\boldsymbol{X}_m^\top\boldsymbol{X}_m|^{1/2}\exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}^\top(g_m\boldsymbol{X}_m^\top\boldsymbol{X}_m)\boldsymbol{\beta})\right\}(\sigma^2)^{-1} \\
&\quad\times(\sigma^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}-\alpha\boldsymbol{i}_N-\boldsymbol{X}_m\boldsymbol{\beta})^\top(\boldsymbol{y}-\alpha\boldsymbol{i}_N-\boldsymbol{X}_m\boldsymbol{\beta})\right\}d\alpha d\boldsymbol{\beta}d\sigma^2.
\end{aligned}$$

Taking into account that $(\boldsymbol{y}-\alpha\boldsymbol{i}_N-\boldsymbol{X}_m\boldsymbol{\beta})^\top(\boldsymbol{y}-\alpha\boldsymbol{i}_N-\boldsymbol{X}_m\boldsymbol{\beta}) = (\boldsymbol{y}-\boldsymbol{X}_m\boldsymbol{\beta})^\top(\boldsymbol{y}-\boldsymbol{X}_m\boldsymbol{\beta}) + N(\alpha-\bar{y})^2 - N\bar{y}^2$,

$$p(\mathbf{y}) \propto \int_0^{\infty} \int_{R^K} (\sigma^2)^{-k_m/2} |g_m \boldsymbol{X}_m^{\top} \boldsymbol{X}_m|^{1/2} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}^{\top}(g_m \boldsymbol{X}_m^{\top} \boldsymbol{X}_m)\boldsymbol{\beta})\right\} (\sigma^2)^{-1}$$
$$\times (\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}[(\boldsymbol{y} - \boldsymbol{X}_m\boldsymbol{\beta})^{\top}(\boldsymbol{y} - \boldsymbol{X}_m\boldsymbol{\beta}) - N\bar{y}^2]\right\}$$
$$\times \int_R \exp\left\{-\frac{1}{2\sigma^2}(N(\alpha - \bar{y})^2)\right\} d\alpha d\boldsymbol{\beta} d\sigma^2.$$

The last term is the kernel of a normal density function with mean $\bar{y}$ and variance σ^2/N, then

$$p(\mathbf{y}) \propto \int_0^{\infty} \int_{R^K} (\sigma^2)^{-k_m/2} |g_m \boldsymbol{X}_m^{\top} \boldsymbol{X}_m|^{1/2} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}^{\top}(g_m \boldsymbol{X}_m^{\top} \boldsymbol{X}_m)\boldsymbol{\beta})\right\} (\sigma^2)^{-1}$$
$$\times (\sigma^2)^{-N/2} (\sigma^2)^{1/2} \exp\left\{-\frac{1}{2\sigma^2}[(\boldsymbol{y} - \boldsymbol{X}_m\boldsymbol{\beta})^{\top}(\boldsymbol{y} - \boldsymbol{X}_m\boldsymbol{\beta}) - N\bar{y}^2]\right\} d\boldsymbol{\beta} d\sigma^2.$$

Collecting terms for $\boldsymbol{\beta}$, we have $(\boldsymbol{y} - \boldsymbol{X}_m\boldsymbol{\beta})^{\top}(\boldsymbol{y} - \boldsymbol{X}_m\boldsymbol{\beta}) - N\bar{y}^2 + \boldsymbol{\beta}^{\top}(g_m \boldsymbol{X}_m^{\top} \boldsymbol{X}_m)\boldsymbol{\beta} = (\boldsymbol{\beta} - \boldsymbol{\beta}_{mn})^{\top} \boldsymbol{B}_{mn}^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_{mn}) + \boldsymbol{y}^{\top}\boldsymbol{y} - N\bar{y}^2 - \boldsymbol{\beta}_{mn}^{\top} \boldsymbol{B}_{mn} \boldsymbol{\beta}_{mn}$ where $\boldsymbol{\beta}_{mn} = \boldsymbol{B}_{mn} \boldsymbol{X}^{\top} \boldsymbol{y}$ and $\boldsymbol{B}_{mn} = ((1 + g_m)\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}$. Then,

$$p(\mathbf{y}) \propto \int_0^{\infty} (\sigma^2)^{-k_m/2} |g_m \boldsymbol{X}_m^{\top} \boldsymbol{X}_m|^{1/2} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}^{\top}\boldsymbol{y} - N\bar{y}^2 - \boldsymbol{\beta}_{mn}^{\top} \boldsymbol{B}_{mn} \boldsymbol{\beta}_{mn})\right\} (\sigma^2)^{-1}$$
$$\times (\sigma^2)^{-N/2} (\sigma^2)^{1/2} \int_{R^K} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_{mn})^{\top} \boldsymbol{B}_{mn}^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}_{mn})\right\} d\boldsymbol{\beta} d\sigma^2.$$

The last term is the kernel of a multivariate normal density with mean $\boldsymbol{\beta}_{mn}$ and variance $\boldsymbol{B}_{mn}$ (proof of the first bullet). Then,

$$p(\mathbf{y}) \propto \int_0^{\infty} \left(\frac{g_m}{1 + g_m}\right)^{k_m/2} (\sigma^2)^{-(N-1)/2-1} \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{y}^{\top}\boldsymbol{y} - N\bar{y}^2 - \boldsymbol{\beta}_{mn}^{\top} \boldsymbol{B}_{mn} \boldsymbol{\beta}_{mn})\right\} d\sigma^2.$$

This is the kernel of an inverse-gamma density with parameters $\alpha_n = (N-1)/2$ and $\delta_n = \boldsymbol{y}^{\top}\boldsymbol{y} - N\bar{y}^2 - \boldsymbol{\beta}_{mn}^{\top} \boldsymbol{B}_{mn} \boldsymbol{\beta}_{mn}$, where $\boldsymbol{y}^{\top}\boldsymbol{y} - N\bar{y}^2 - \boldsymbol{\beta}_{mn}^{\top} \boldsymbol{B}_{mn} \boldsymbol{\beta}_{mn} = (\boldsymbol{y} - \boldsymbol{i}_N\alpha)^{\top}(\boldsymbol{y} - \boldsymbol{i}_N\alpha) - (1 + g_m)^{-1} \boldsymbol{y}^{\top}(\boldsymbol{X}(\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}\boldsymbol{X}^{\top})\boldsymbol{y}$. Then,

$$p(\boldsymbol{y}|\mathcal{M}_m) \propto \left(\frac{g_m}{1 + g_m}\right)^{k_m/2} \left[(\boldsymbol{y} - \bar{y}\boldsymbol{i}_N)^{\top}(\boldsymbol{y} - \bar{y}\boldsymbol{i}_N) - \frac{1}{1 + g_m}(\boldsymbol{y}^{\top} \boldsymbol{P}_{X_m} \boldsymbol{y})\right]^{-(N-1)/2},$$

2. **Determinants of export diversification I**

   [18] use BMA to study the determinants of export diversification. Use the dataset *10ExportDiversificationHHI.csv* to perform BMA using the BIC approximation and MC3 with 10000 iterations to check if these two approaches agree.

   Answer

   The first aspect to note is that the BIC approximation is faster by far than the MC3 algorithm. We see from the results that the two approaches show that the most relevant variables to determine export diversification are *avgedu5* (primary education) and *avgnatres* (natural resources). See [18] for details of all variables. The model with the highest PMP using MC3 includes these two variables (PMP = 0.16), while the model with the highest PMP using the BIC approximation in addition to these two variables also includes Portugal former colony and population (PMP = 0.03). Both methods agree that export diversification increases with primary education, and decreases with natural resources.

R code. Determinants of export diversification

```
rm(list = ls()); set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/10
    ExportDiversificationHHI.csv", sep = ",", header = TRUE,
     quote = "")
attach(Data)
y <- Data[,1]; X <- as.matrix(Data[,-1]); K <- dim(X)[2]
BMAglm <- BMA::bicreg(X, y, strict = FALSE, OR = 50)
summary(BMAglm)
BMAreg <- BMA::MC3.REG(y, X, num.its=10000)
Models <- unique(BMAreg[["variables"]])
nModels <- dim(Models)[1]
nVistModels <- dim(BMAreg[["variables"]])[1]
PMPmc3 <- 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])
  PMPmc3 <- c(PMPmc3, PMPm)
}
PMPmc3
PIPmc3 <- NULL
for(k in 1:K){
  PIPk <- sum(PMPmc3[which(Models[,k] == 1)])
  PIPmc3 <- c(PIPmc3, PIPk)
}
plot(PIPmc3)
Meansmc3 <- matrix(0, nModels, K)
Varsmc3 <- 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)
    Meansmc3[m, idXs] <- SumRegm[["coefficients"]][-1,1]
    Varsmc3[m, idXs] <- SumRegm[["coefficients"]][-1,2]^2
  }
}
BMAmeansmc3 <- colSums(Meansmc3*PMPmc3)
BMAsdmc3 <- (colSums(PMPmc3*Varsmc3)  + colSums(PMPmc3*(
    Meansmc3-matrix(rep(BMAmeansmc3, each = nModels),
    nModels, K))^2))^0.5
plot(BMAmeansmc3)
plot(BMAsdmc3)
Ratio <- BMAmeansmc3/BMAsdmc3
RessulMC3 <- as.data.frame(cbind(PIPmc3, Models[1,],
    BMAmeansmc3, BMAsdmc3, Ratio))
PMPmc3
```

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 of the book increasing the number of regressors to 40, this implies approximately 1.1e+12 models.

   Answer

   The following code shows how to perform BMA using MC3 with the consideration that all final S models should be different.

   After running the algorithm 50000 ($<< 2^{40}$) times, we can see that the PIP is 1 for variables x_1, x_5 and x_{10}, which are the variables in the data generating process (population statistical model). However, we can see that variable x_{23} has a high PIP (0.49), this makes that the PMP of the model including x_1, x_5, x_{10} and x_{23} is the highest, followed by the model including x_1, x_5 and x_{10}, which is the population statistical model. This highlights the relevance of performing BMA; selecting just one model based on the highest PMP would induce a mistake. Although selecting the median probability model would uncover the population statistical model.

   Estimating the BMA mean shows that we get values very close to the population values, a remarkable results is all other BMA posterior means are close to 0, including the mean coefficient of x_{23} despite that its PIP is almost 0.5. We calculate the posterior ratio between the mean and standard deviation, and get values higher than 2 just for x_1, x_5 and x_{10}, again given evidence for the data generating process.

R code. Markov chain Monte Carlo model composition

```
rm(list = ls()); set.seed(010101)
N <- 1000
K1 <- 6; K2 <- 4; K3 <- 30; K <- K1 + K2 + K3
X1 <- matrix(rnorm(N*K1,1 ,1), N, K1)
X2 <- matrix(rbinom(N*K2, 1, 0.5), N, K2)
X3 <- matrix(rnorm(N*K3,1 ,1), N, K3)
X <- cbind(X1, X2, X3); e <- rnorm(N, 0, 0.5)
B <- c(1,0,0,0,0.5,0,0,0,0,-0.7, rep(0, 30))
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 <- diag(N) - Xr%*%solve(t(Xr)%*%Xr)%*%t(Xr)
    # s2pos <- c(t(y)%*%PX%*%y/(1 + gr) + gr*(t(y - mean(y))
    %*%(y - mean(y)))/(1 + gr))
    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))
    # PX <- diag(N)
    # s2pos <- c(t(y)%*%PX%*%y/(1 + gr) + gr*(t(y - mean(y))
    %*%(y - mean(y)))/(1 + gr))
    s2pos <- c((t(y - mean(y))%*%(y - mean(y))))
    mllMod <- (kr/2)*log(gr/(1+gr))-(N-1)/2*log(s2pos)
  }
  return(mllMod)
}
Xnew <- apply(X, 2, scale); M <- 100
Models <- matrix(rbinom(K*M, 1, p = 0.5), ncol = K, nrow = M
     + 800)
Models <- unique(Models)[1: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 <- 50000; s <- 1
pb <- txtProgressBar(min = 0, max = iter, style = 3)
```

R code. Markov chain Monte Carlo model composition

```
while(s <= iter){
  ActModel <- Models[M,]
  idK <- which(ActModel == 1)
  Kact <- length(idK)
  Continue <- 0
  while(Continue == 0){
    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. Markov chain Monte Carlo model composition

```
    check <- NULL
  for(j in 1:M){
    if(sum(Models[j,] == CandModel) == K){
      checkj <- 0
      check <- c(check, checkj)
    }else{
      checkj <- 1
      check <- c(check, checkj)
    }
  }
  dimUniModels <- sum(check)
  if(dimUniModels == M){
    Continue <- 1
  }else{
    Continue <- 0
  }
}
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
setTxtProgressBar(pb, s)
}
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)
PIP <- NULL
for(k in 1:K){
PIPk <- sum(PMP[which(ModelsUni[,k] == 1)])
PIP <- c(PIP, PIPk)
}
PIP
```

R code. Markov chain Monte Carlo model composition

```
Means <- matrix(0, M, K)
Vars <- matrix(0, M, K)
for(m in 1:M){
  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 = M), M, K))^2))^0.5
plot(BMAmeans)
plot(BMAsd)
plot(BMAmeans/BMAsd)
```

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.

   Answer

   The following code shows how to perform BMA using the BIC approximation and MC3 in this simulation setting with endogeneity. We see from the results that the BIC approximation and MC3 do a good job with the PMP and the PIP, as the data generating process gets the highest PMP using both approaches, and the PIPs of the variables in the data generating process are equal 1. The critical point is the BMA posterior means of the endogenous regressors, as these are far from the population values. The population values of x_{i1} and x_{i2} are 0.5 and -1, whereas the posterior means are 0.97 and -0.52.

R code. BIC and MC3 in model with endogeneity

```
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(1:4)]
Xnew <- cbind(X, W)
BMAglm <- BMA::bicreg(Xnew, y, strict = FALSE, OR = 50)
summary(BMAglm)
BMAreg <- BMA::MC3.REG(y, Xnew, num.its=10000)
Models <- unique(BMAreg[["variables"]])
nModels <- dim(Models)[1]
nVistModels <- dim(BMAreg[["variables"]])[1]
K <- dim(Xnew)[2]
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)
}
PMP
PIP <- NULL
for(k in 1:K){
  PIPk <- sum(PMP[which(Models[,k] == 1)])
  PIP <- c(PIP, PIPk)
}
plot(PIP)
```

R code. BIC and MC3 in model with endogeneity

```
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 <- Xnew[, 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
plot(BMAmeans)
plot(BMAsd)
plot(BMAmeans/BMAsd)
```

The previous results are intuitive, as the PMP are calculated based on fit (and penalty for complexity), for instance, $BIC = k_m \log(N) - 2\log(p(\hat{\boldsymbol{\theta}_m}|\boldsymbol{y}))$, where $\hat{\boldsymbol{\theta}}_m$ is the maximum likelihood estimator. Observe that the fit is not affected by endogeneity. Thus, the PMP are well calculated. However, the coefficients are not well identified.

This suggests a simple strategy to perform BMA taking into account endogeneity, calculate the PMPs using standard BMA approaches, for instance, BIC approximation, and then estimate the BMA means using these PMPs, but estimating the different models using instrumental variables. This approach is easily implemented using packages from **R**. The following code does this:

R code. Easy IV BMA

```
BMAglm <- BMA::bicreg(Xnew, y, strict = FALSE, OR = 50)
summary(BMAglm)
PMPBIC <- BMAglm[["postprob"]]
ModelsBIC <- BMAglm[["which"]]
nModels <- dim(ModelsBIC)[1]
K <- dim(Xnew)[2]
Means <- matrix(0, nModels, K)
Vars <- matrix(0, nModels, K)
for(m in 1:nModels){
  idXs <- which(ModelsBIC[m,] == 1)
  if(length(idXs) == 0){
    Regm <- lm(y ~ 1)
  }else{
    Xm <- Xnew[, idXs]
    Regm <- ivreg::ivreg(y ~ Xm | z + W)
    SumRegm <- summary(Regm)
    Means[m, idXs] <- SumRegm[["coefficients"]][-1,1]
    Vars[m, idXs] <- SumRegm[["coefficients"]][-1,2]^2
  }
}
BMAmeans <- colSums(Means*PMPBIC)
BMAsd <- (colSums(PMPBIC*Vars)  + colSums(PMPBIC*(Means-
    matrix(rep(BMAmeans, each = nModels), nModels, K))^2))
    ^0.5
BMAmeans
5.589366e-01 -9.431664e-01  1.035593e+00 -1.967444e-05
    -1.429801e-04 -3.310215e-04  4.036846e-04 4.651796e-03
    -2.673730e-03  6.462503e-05  4.286529e-04 -1.829889e-04
     5.073229e-03 -1.007356e-04 -5.377972e-04  1.644475e-03
    -4.029205e-04 -1.457381e-04  6.421582e-04  2.211994e-04
     2.216503e-04
plot(BMAsd)
plot(BMAmeans/BMAsd)
17.764095075 -23.233601133  34.734610916  -0.005600369
    -0.039168667  -0.071895067   0.080909448 0.274994618
    -0.210854131   0.018790592   0.061271650  -0.047117887
     0.305639285  -0.028830411 -0.092463544   0.169455952
    -0.058299940  -0.038099792   0.102373567   0.056074670
     0.055500343

```

We observe that the Easy IV BMA means of the endogenous regressors are 0.59 and -0.94, which are closer to the population values (0.5 and -1) than the exogenous BMA means (0.97 and -0.52) and align closely with the IV BMA means calculated using conditional Bayes factors (0.51 and -0.98; see Section 10.2 in the book). Additionally, the ratios between the Easy IV

BMA means and their standard deviations exceed 2 in absolute value for these variables, and the t-intervals encompass the population values

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 [18] for details.

   Answer

   The results show that primary education (*avgedu5*) and natural resources (*avgnatres*) have again the highest PIP, 0.77 and 0.92, respectively. The former variable increases export diversification, and the latter decreases export diversification. However, it seems that endogeneity is not a concern in this application, as the 95% credible interval of σ_{12} is (-0.014, 0.024).

R code. IV BMA in export diversification

```
rm(list = ls())
set.seed(010101)
DataMain <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/11ExportDiversificationHHI.csv", sep = ",", header = TRUE, quote = "")
DataInst <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/12ExportDiversificationHHIInstr.csv", sep = ",", header = TRUE, quote = "")
attach(DataMain)
attach(DataInst)
y <- DataMain[,1]
X <- as.matrix(DataMain[,2])
W <- as.matrix(DataMain[,-c(1:2)])
Z <- as.matrix(DataInst)
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
EVmain <- regivBMA[["rho.bar"]] # Posterior mean outcome
EVmain
PIPaux <- regivBMA[["M.bar"]] # PIP auxiliary
PIPaux
EVaux <- regivBMA[["lambda.bar"]] # Posterior mean auxiliary
plot(EVaux[,1])
EVsigma <- regivBMA[["Sigma.bar"]] # Posterior mean variance matrix
EVsigma
summary(coda::mcmc(regivBMA[["Sigma"]][1,2,]))
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
3.944e-03      9.592e-03      9.592e-05      4.043e-04
2. Quantiles for each variable:
2.5%        25%        50%        75%      97.5%
-0.014155 -0.002415  0.003592  0.009988  0.024047
```

6. Show that the link function in the case of the Bernoulli distribution is $\log\left(\frac{\theta}{1-\theta}\right)$.

Answer

$$\begin{aligned} p(\mathbf{y}|\theta) &= \theta^{y}(1-\theta)^{1-y} \\ &= (1-\theta)\exp\left\{y\log\left(\frac{\theta}{1-\theta}\right)\right\} \\ &= \exp\left\{y\log\left(\frac{\theta}{1-\theta}\right)+\log(1-\theta)\right\}, \end{aligned}$$

then $\eta(\theta)=\log\left(\frac{\theta}{1-\theta}\right)$, and this density in the canonical form is $p(y|\eta)=\exp\{y\eta-\log(1+\exp(\eta))\}$ consequently, $\mathbb{E}[Y|\boldsymbol{x}]=\nabla\left(\log(1+\exp(\eta))\right)=\frac{\exp(\eta)}{1+\exp(\eta)}=\theta=\frac{\exp(\boldsymbol{x}^{\top}\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}^{\top}\boldsymbol{\beta})}$. Then, the link function in the Bernoulli case is the *logit* function.

7. [34, 35] perform variable selection using the file *13InternetMed.csv*. In this data set, the dependent variable is an indicator of Internet adoption (internet) for 5000 households in Medellín (Colombia) during the period 2006–2014. This dataset contains information about 18 potential determinants, which means 262144 (2^{18}) potential models just taking into account variable uncertainty (see these papers for details about the data set). Perform BMA using the logit link function using this data set.

Answer

R code. BMA logit in internet adoption

```
rm(list = ls())
set.seed(010101)
DataMain <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/13InternetMed
    .csv", sep = ",", header = TRUE, quote = "")
attach(DataMain)
### BIC approximation
y <- DataMain[,1]
X <- as.matrix(DataMain[,-1])
BMAglm <- BMA::bic.glm(X, y, strict = FALSE, OR = 50, glm.
    family = binomial(link="logit"))
summary(BMAglm)

```

The results show that the best model has a posterior model probability equal to 31%, and the second best model has a PMP equal to 30%. It seems that age, squared age, years of education of the head of the household, total expenses, having pay TV, any household member studying, and number of children in the household are relevant determinants of Internet adoption.

8. [42] 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 in the five most important leagues in Europe (see [42] for details). Use this data set to perform BMA using the gamma distribution with the log link function, and setting default values for Occam's window.

   Answer

R code. BMA gamma in market value of soccer players in Europe

```
rm(list = ls())
set.seed(010101)
DataMain <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/14
    ValueFootballPlayers.csv", sep = ",", header = TRUE,
    quote = "")
attach(DataMain)
### BIC approximation
y <- DataMain[,1]
X <- as.matrix(DataMain[,-1])
BMAglm <- BMA::bic.glm(X, y, strict = FALSE, OR = 50, glm.
    family = Gamma(link="log"))
summary(BMAglm)
```

   The results show that performance, age, squared age, participation in national team, scored goals, and appearance in the UEFA champions league have PIPs equal 1. All these variables increase the market value of soccer players, except squared age.

9. Use the dataset *15Fertile2.csv* from [48, p. 547] to perform BMA using the Poisson model with the log link. This data set has information about 1,781 women from Botswana in 1988 (for details, see **https://rdrr.io/cran/wooldridge/man/fertil2.html**, and take into account 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.

   Answer

R code. BMA Poisson in determinants of fertility in Bostwana

```
rm(list = ls())
set.seed(010101)
DataMain <- read.csv("https://raw.githubusercontent.com/
    besmarter/BSTApp/refs/heads/master/DataApp/15Fertil2.csv
    ", sep = ",", header = TRUE, quote = "")
attach(DataMain)
### BIC approximation
y <- DataMain[,1]
X <- as.matrix(DataMain[,-1])
BMAglm <- BMA::bic.glm(X, y, strict = FALSE, OR = 50, glm.
    family = poisson(link="log"))
summary(BMAglm)
```

We found that the best model, for which the PMP is equal to 27%, has as regressors age, squared age, age at first birth, use birth control, husband's years of education, woman's years of education, and living in an urban area. The first five variables have PIPs equal to 100. For instance, we found that women using birth control have approximately 15% fewer children on average than women who are not using birth control.

10. Perform BMA in the logit model using MC3 and the BIC approximation using the simulation setting of Section 10.3.

    Answer

R code. BMA Logit model

```
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); table(y)
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 = binomial(link = "logit"))
    model_bic <- BIC(model)
    mllMod <- -model_bic/2
  }else{
    model <- glm(y ~ 1, family = binomial(link = "logit"))
    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, ]
iter <- 25000; s <- 1
pb <- txtProgressBar(min = 0, max = iter, style = 3)
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. BMA Logit model

```
  }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)) # Let's reasonably
    assume same prior model probability for candidate and
    actual, and same carnality of neighbor models
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
setTxtProgressBar(pb, s)
}
close(pb)

```

R code. BMA Logit model

```
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 = binomial(link = "logit"))
  }else{
    Xm <- as.matrix(Xnew[, idXs])
    Regm <- glm(y ~ Xm, family = binomial(link = "logit"))
    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)
```

The simulation setting implies 2^{27} models, which implies approximately 135 million models in the model space. We run our MC3 algorithm using the BIC approximation with 25000 iterations. This takes by far more time that the BIC approximation from the *BMA* package, but it seems to do a good job finding the data generating process as the highest PMP is associated with it, the PIPs of x_{i1} and x_{i2} are 1, the posterior means are 0.85 and -1.1, which are very close to the population values (0.8 and -1.2), and the

t-ratios are by far higher than 2. The PIPs of the other regressors are lower than 20%, and the BMA means are close to 0.

11. Use the dataset *19ExchangeRateCOPUSD.csv* to estimate 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) is the annual variation rate of the exchange rate COP to USD, *TES_COL10* (i_t^{Col}) and *TES_USA10* (i_t^{USA}) are the annual return rates of Colombian and USA public debts in 10 years, *inflation_COL* (π_t^{Col}) and *inflation_USA* (π_t^{USA}) are the annual inflation rates, *varISE_COL* (g_t^{Col}) and *varISE_USA* (g_t^{USA}) are annual variation of economic activity indices, and *varCOL_M3* (m_t^{Col}) and *varUSA_M3* (m_t^{USA}) are annual variations of money supply. We have monthly variations between January 2006 and November 2023.

    Perform Bayesian model averaging using these four models to explain the annual variation of the exchange rate, get the posterior model probabilities, and plot the posterior mean and credible interval of β_{2t}^{Money}.

    Answer

R code. Dynamic Bayesian model average: Theories to explain (forecast) exchange rate

```
rm(list = ls()); set.seed(010101)
DataMain <- read.csv("https://raw.githubusercontent.com/besmarter/BSTApp/refs/heads/master/DataApp/19ExchangeRateCOPUSD.csv", sep = ",", header = TRUE, quote = "")
X <- DataMain[-c(1:12,227),c(4, 5, 7, 9, 11, 13, 14, 15)]
DifIntRat <- X[,8] - X[,7]; DifInf <- X[,1] - X[,2]
DifGrowth <- X[,3] - X[,4]; DifM3 <- X[,6] - X[,5]
XDif <- cbind(DifIntRat, DifInf, DifGrowth, DifM3)
matplot(XDif, type = "l")
y <- DataMain[-c(1:13),17] # Var exchange rate depends on lag of other variables
plot(y, type = "l")
IRP <- c(1, 0, 0, 0); PPP <- c(0, 1, 0, 0)
Taylor <- c(0, 1, 1, 0); Money <- c(0, 0, 1, 1)
Models <- rbind(IRP, PPP, Taylor, Money)
T0 <- 50
dma.test <- dma::dma(XDif, y, Models, lambda=.99, gamma=.99, initialperiod = T0)
plot(ts(dma.test[["pmp"]][,1], start = 2006, end = 2023, frequency = 12), type = "l", main = "Exchange rate models", ylab = "Posterior model probability", ylim = c(0,1)) lines(ts(dma.test[["pmp"]][,2], start = 2006, end = 2023, frequency = 12), col = "red") lines(ts(dma.test[["pmp"]][,3], start = 2006, end = 2023, frequency = 12), col = "green") lines(ts(dma.test[["pmp"]][,4], start = 2006, end = 2023, frequency = 12), col = "blue") legend(x = 2009, y = 0.6, legend = c("IRP", "PPP", "Taylor rule", "Money supply"), col = c("black", "red", "green", "blue"), lty=1:1, cex=0.8)
require(latex2exp)
plot_filtering_estimates <- function(df) {
  p <- ggplot(data = df, aes(x = t)) + geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.5, fill = "blue") + geom_line(aes(y = mean), colour = "black", linewidth = 0.8) + geom_hline(yintercept=0, colour = "red") + ylab(TeX("$\\beta_{Growth_t}^{Money}$")) + xlab("Time")
  print(p)
}
menathetaIRP <- dma.test[["thetahat"]][4,,4]
SDthetaIRP <- (dma.test[["Vtheta"]][4,,4])^0.5
LimInf <- menathetaIRP - 2*SDthetaIRP
LimSup <- menathetaIRP + 2*SDthetaIRP
library(ggplot2,tibble,dplyr)
df <- tibble(t = 1:214, mean = menathetaIRP, lower = LimInf, upper = LimSup)
Fig <- plot_filtering_estimates(df)
```

Figure 10.1 shows the posterior model probabilities. We see that at the beginning, between 2006 and 2007, the purchasing power parity theory (PPP) had the highest PMP (70% on average) to explain the annual variations of the COP/USD exchange rate. Then, the money supply theory (Money) got the highest PMP (80% on average) between 2008 and 2017. Finally, the interest rate parity got the highest PMP (80% on average) between 2018 and 2023.

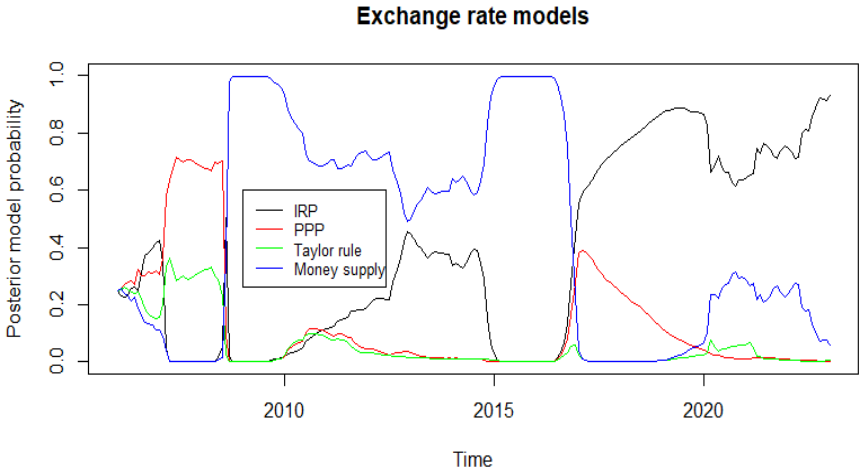

FIGURE 10.1
Dynamic posterior model probability: Models to explain variations of the COP to USD exchange rate.

Figure 10.2 shows the dynamic posterior mean (black line) and credible interval (blue shadow) of the coefficient related to the differential of economic growths (Colombia vs the USA) in the money supply equation to explain the annual variations of the exchange rate. We observe that increasing this economic growth differential implies appreciation of the COP relatively to USD.

12. Perform a simulation of the dynamic logistic model, where there are 7 ($2^3 - 1$, the model without regressors is excluded) competing models originated 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 $1/T$, $\beta_{3t} = \left\{ \begin{matrix} -1, & 1 < t \leq 0.5T \\ 0, & 0.5T < t \leq T \end{matrix} \right\}$, and $\beta_4 = 1.2$. Then, $\boldsymbol{x}_t^\top \boldsymbol{\beta}_t = \beta_1 + \beta_{2t} x_{2t} + \beta_{3t} x_{3t} + \beta_4 x_{4t}$, where $P[Y_t = 1 | \boldsymbol{x}_t, \boldsymbol{\beta}_t] = \exp(\boldsymbol{x}_t^\top \boldsymbol{\beta}_t)/(1+\exp(\boldsymbol{x}_t^\top \boldsymbol{\beta}_t))$, $t = 1, 2, \ldots, 1100$. Use the function *logistic.dma* from the *dma* package to get the posterior model probabilities

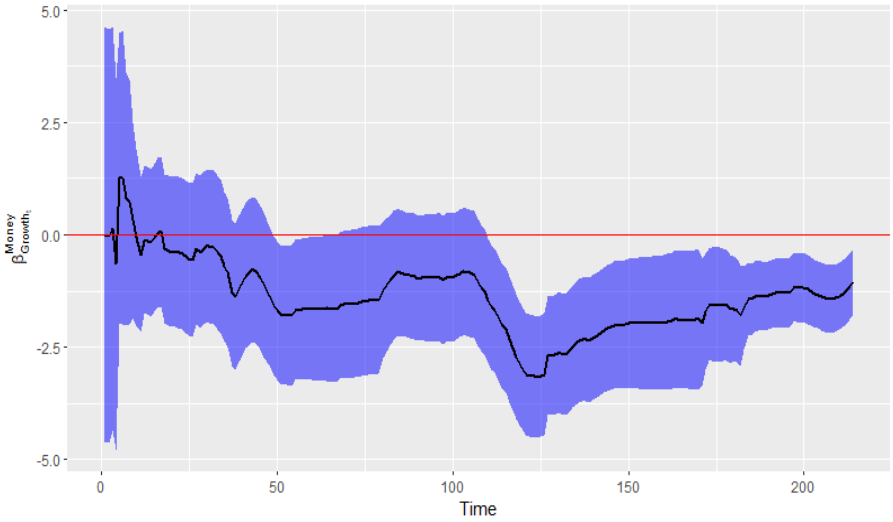

FIGURE 10.2
State coefficient money supply theory: Differential of economic growths (Colombia vs the USA).

setting the forgetting parameter of the models equal to 0.99, and then to 0.95. Compare the results.

Answer

The following code shows the simulation exercise, and the results of the dynamic Bayesian model average of logistic models.

R code. Dynamic Bayesian model average: Logit model

```
rm(list = ls()); set.seed(010101)
T <- 1100; 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 <- 0.5
B3t <- c(rep(-1,round(a*T)), rep(0,round((1-a)*T)))
B4 <- 1.2
yl <- B1 + X[,1]*B2t + X[,2]*B3t + X[,3]*B4
py <- exp(yl)/(1 + exp(yl))
y <- rbinom(T, 1, prob = py)
table(y)
T0 <- 100
dma.test <- dma::logistic.dma(X, y, combs[-1,], lambda =
    0.99, alpha = 0.99, initialsamp = T0)
plot(dma.test[["pmp"]][7, -c(1:T0)], type = "l", col = "
    green", main = "Posterior model probability: Model all
    regressors vs model regressors 1 and 3", xlab = "Time",
    ylab = "PMP", ylim = c(0, 1))
lines(dma.test[["pmp"]][5,-c(1:T0)], col = "red")
legend(x = 0, y = 0.9, legend = c("Model: All regressors", "
    Model: Regressors 1 and 3"), col = c("green", "red"),
    lty=1:1, cex=0.8)
dma.test1 <- dma::logistic.dma(X, y, combs[-1,], lambda =
    0.99, alpha = 0.95, initialsamp = T0)
plot(dma.test1[["pmp"]][7, -c(1:T0)], type = "l", col = "
    green", main = "Posterior model probability: Model all
    regressors vs model regressors 1 and 3", xlab = "Time",
    ylab = "PMP", ylim = c(0, 1))
lines(dma.test1[["pmp"]][5,-c(1:T0)], col = "red")
legend(x = 0, y = 0.9, legend = c("Model: All regressors", "
    Model: Regressors 1 and 3"), col = c("green", "red"),
    lty=1:1, cex=0.8)
```

Figure 10.3 shows the posterior model probabilities of the model with all the regressors (green line), and the model with regressors 2 and 4 (red line), setting the forgetting parameter of the models equal to 0.99. In one hand, we see that the model with all regressors, which is the data generating process in the first period ($t \leq 0.5T$), gets high posterior model probabilities. We see that the PMPs decrease around $T = 700$, whereas the change in the data generating process is at $T = 550$. This means that there is a lack to identify the data generating process, which is a consequence of the high forgetting parameter of the models. Consequently, the PMPs of the model with regressors 2 and 4 increases around $T = 700$, but it is lower than the

PMPs of the model with all regressors, despite that in this time period the data generating process is given by the model with regressors 2 and 4.

Figure 10.4 shows the posterior model probabilities using a forgetting parameter equal to 0.95. We can see that as the forgetting parameter decreases, the process of identifying the data generating process is faster, and consequently, the PMP of the model with regressors 2 and 4 increases around $T = 550$, which is the turning point in the data generating process. This does not mean that we should set small forgetting parameters. These parameters should be set based on model performance, for instance, small mean squared error in a cross validation period.

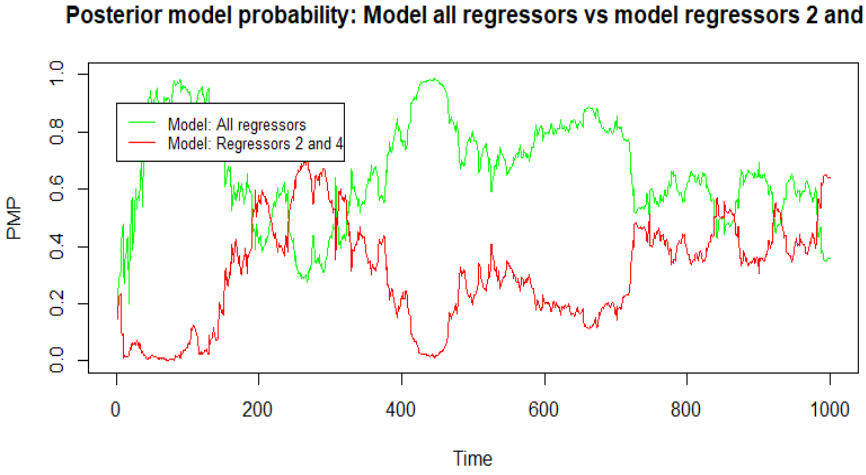

FIGURE 10.3
Posterior model probability: Dynamic Bayesian model average, forgetting parameter equal to 0.99.

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 $\boldsymbol{\Theta}$.

Answer

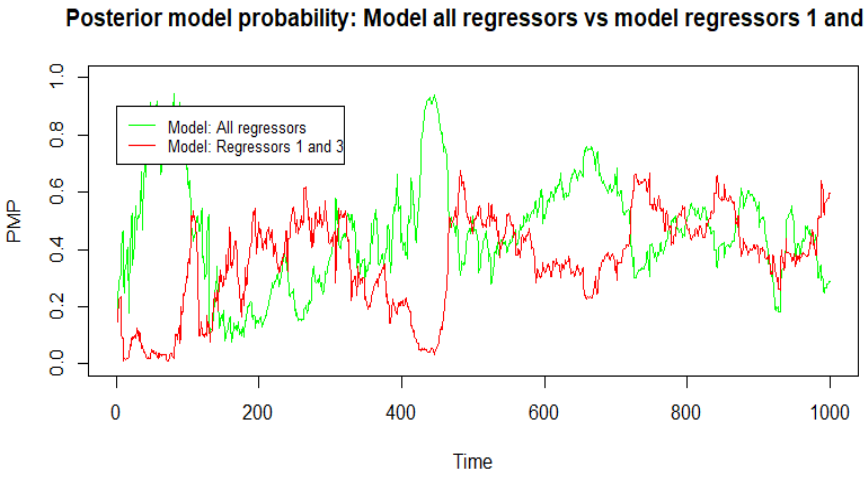

FIGURE 10.4
Posterior model probability: Dynamic Bayesian model average, forgetting parameter equal to 0.95.

Given a probability density function $q(\boldsymbol{\theta})$, whose support is in $\boldsymbol{\Theta}$, then

$$\begin{aligned}
\mathbb{E}\left[\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m,\mathcal{M}_m)}\middle|\boldsymbol{y},\mathcal{M}_m\right] &= \int_{\boldsymbol{\Theta}}\left[\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m,\mathcal{M}_m)}\right]\pi(\boldsymbol{\theta}|\boldsymbol{y},\mathcal{M}_m)d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\Theta}}\left[\frac{q(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m,\mathcal{M}_m)}\right]\frac{\pi(\boldsymbol{\theta}|\mathcal{M}_m)p(\boldsymbol{y}|\boldsymbol{\theta}_m,\mathcal{M}_m)}{p(\boldsymbol{y}|\mathcal{M}_m)}d\boldsymbol{\theta} \\
&= \frac{1}{p(\boldsymbol{y}|\mathcal{M}_m)}\int_{\boldsymbol{\Theta}} q(\boldsymbol{\theta})d\boldsymbol{\theta} \\
&= \frac{1}{p(\boldsymbol{y}|\mathcal{M}_m)}.
\end{aligned}$$

Part III

Advanced methods: A brief introduction

11

Semi-parametric and non-parametric models

Solutions of Exercises

1. Simulate a semi-parametric regression where

$$y_i = 0.5x_{i1} - 1.2x_{i2} + \mu_i,$$
$$p(\mu_i) = 0.7\phi(\mu_i \mid -0.5, 0.5^2) + 0.3\phi(\mu_i \mid 1, 0.8^2).$$

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 $\alpha_{h0} = \delta_{h0} = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_2$, $\boldsymbol{B}_0 = \boldsymbol{I}_2$, $\mu_{h0} = 0$, $\sigma^2_{h0} = 10$, 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).

Answer

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 all parameters, and the posterior density estimate closely matches the population density. See Figure 11.1, where the black line represents the population density, the blue line denotes the posterior mean density estimate, and the shaded light blue area corresponds to the 95% credible interval.

R code. Simulation: Semi-parametric Gaussian mixture

```
rm(list = ls()); set.seed(010101)
# Simulate data from a 2-component mixture model
n <- 1000
x1 <- rnorm(n); x2 <- rnorm(n)
B <- c(0.5, -1.2); X <- cbind(x1, x2)
z <- rbinom(n, 1, 0.3)  # Latent class indicator
mu <- ifelse(z == 0, rnorm(n, -0.5, 0.5), rnorm(n, 1, 0.8))
plot(density(mu))
y <- X%*%B + mu
# 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: Semi-parametric Gaussian mixture

```
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(); Reg <- lm(y ~ X)
Beta <- Reg$coefficients[2:3]; Res <- Reg$residuals
plot(density(Res))
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]))
Psi <- rep(NA, n); Hs <- length(MU)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
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
  setTxtProgressBar(pb, s)
}
close(pb)
```

R code. Simulation: Semi-parametric Gaussian mixture

```
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,]))})
PosteriorSIGMA <- matrix(NA, length(keep), 2)
PosteriorMU <- matrix(NA, length(keep), 2)
PosteriorLAMBDA <- matrix(NA, length(keep), 2)
l <- 1
for (s in keep){
  PosteriorSIGMA[l,] <- PostSigma2[[s]][1:2]
  PosteriorMU[l,] <- PostMu[[s]][1:2]
  PosteriorLAMBDA[l,] <- PostLambda[[s]][1:2]
  l <- l + 1
}
summary(coda::mcmc(PosteriorSIGMA))
summary(coda::mcmc(PosteriorMU))
summary(coda::mcmc(PosteriorLAMBDA))
mus <- seq(-3, 4, 0.01)
DensityEst <- function(par, eval){
  lambda1 <- par[1]; lambda2 <- par[2]
  mu1 <- par[3]; mu2 <- par[4]
  sd1 <- par[5]; sd2 <- par[6]
  Dens <- lambda1 * dnorm(eval, mu1, sd1) + lambda2 * dnorm(
    eval, mu2, sd2)
  return(Dens)
}
par <- cbind(PosteriorLAMBDA, PosteriorMU, PosteriorSIGMA
    ^0.5)
DensEval <- matrix(NA, length(keep), length(mus))
DensPop <- rep(NA, length(mus))
for(r in 1:length(mus)){
  for(l in 1:length(keep)){
    DensEval[l, r] <- DensityEst(par = par[l,], eval = mus[r
    ])
  }
  DensPop[r] <- DensityEst(par = c(0.7, 0.3, -0.5, 1, 0.5,
    0.8), eval = mus[r])
}
library(dplyr); library(ggplot2); require(latex2exp)
DataDens <- tibble(t = mus, Pop = DensPop, lower = apply(
    DensEval, 2, quantile, probs = 0.025), upper = apply(
    DensEval, 2, quantile, probs = 0.975), meanT = colMeans(
    DensEval))
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 = Pop, color = "True Value"),
    linewidth = 0.5) +  geom_line(aes(y = meanT, color = "
    Estimate"), linewidth = 0.5) +
  scale_color_manual(values = c("True Value" = "black", "
    Estimate" = "blue")) + xlab(TeX("$\\mu$")) + ylab("
    Density") +   labs(title = "Density: Stochastic Errors",
     color = "") + theme_minimal()
  print(p)
}
plot_filtering_estimates(DataDens)
```

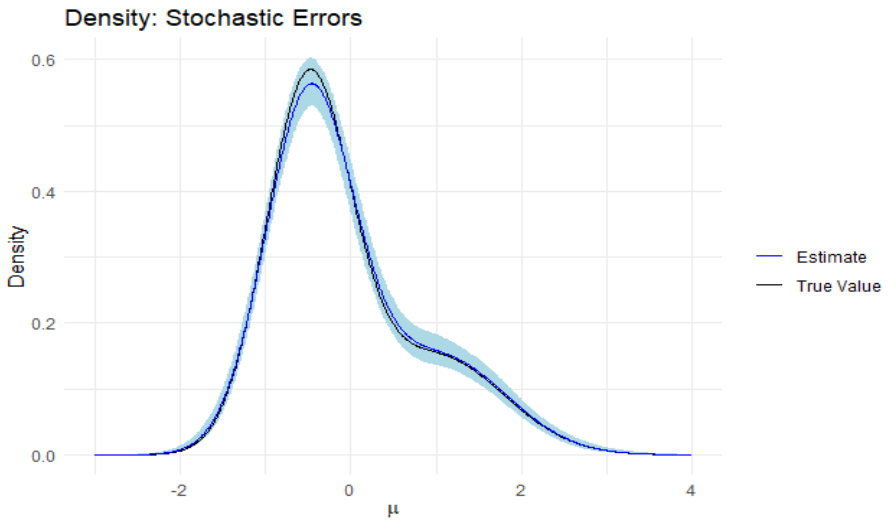

FIGURE 11.1
Posterior density estimate: Stochastic errors.

2. 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.

   Answer:

   The following code demonstrates how to perform inference in this model. The results indicate the presence of four clusters, with mean probabilities of 0.64, 0.14, 0.12 and 0.10, and 95% credible intervals of (0.60, 0.67), (0.12,0.16), (0.10,0.14) and (0.08, 0.12), respectively.

The 95% posterior credible interval for the own-price elasticities of the most representative group, which comprises two-thirds of the sample, is $(-0.55, -0.26)$. Additionally, the 95% credible interval for the cross-price elasticities with respect to cocaine is $(0.01, 0.35)$, while that with respect to crack is $(0.07, 0.43)$.

These results suggest that marijuana is an inelastic good for these individuals and that there is substitution between marijuana and both cocaine and crack: an increase in the prices of the latter leads to higher marijuana demand in the most representative cluster of individuals. However, these findings should be interpreted with caution due to potential endogeneity issues that must be addressed for a rigorous scientific analysis. The primary goal of this exercise is pedagogical.

R code. Marijuana in Colombia: Gaussian 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); summary(Reg)
k <- dim(X)[2]; N <- dim(X)[1]
# Plot
library(ggplot2)
ggplot(Data, aes(x = LogMarijuana)) + geom_density(fill = "
    blue", alpha = 0.3) + labs(title = "Density Plot", x = "
    y", y = "Density") + theme_minimal()
# Hyperparameters
d0 <- 0.001; a0 <- 0.001
b0 <- rep(0, k); B0 <- diag(k); B0i <- solve(B0)
H <- 5; a0h <- rep(1/H, H)
# MCMC parameters
mcmc <- 2000; burnin <- 500
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){
  if(length(yh) == 1){
    Xh <- matrix(Xh, k, 1)
    Bn <- solve(B0i + sig2h^(-1)*Xh%*%t(Xh))
    bn <- Bn%*%(B0i%*%b0 + sig2h^(-1)*Xh%*%yh)
  }else{
    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)
}
PostBetas <- list(); PostSigma2 <- list(); PostLambda <-
    list()
PostPsi <- matrix(0, tot, N); Psi <- sample(1:H, N, replace
    = TRUE, prob = a0h)
Beta <- matrix(rep(Reg$coefficients, H), k, H); Hs <- H
```

R code. Marijuana in Colombia: Gaussian mixture

```
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  Sig2 <- rep(NA, Hs)
  for(h in 1:Hs){
    idh <- which(Psi == h)
    Sig2[h] <- PostSig2(Betah = Beta[, h], Xh = X[idh,], yh
    = y[idh])
  }
  Hs <- length(Sig2)
  Beta <- matrix(NA, k, Hs)
  for(h in 1:Hs){
    idh <- which(Psi == h)
    Beta[, h] <- PostBeta(sig2h = Sig2[h], Xh = X[idh,], yh
    = y[idh])
  }
  Lambda <- sort(MCMCpack::rdirichlet(1, a0h[1:Hs] + table(
    Psi)), decreasing = TRUE)
  for(i in 1:N){
    lambdai <- NULL
    for(h in 1:Hs){
      idh <- which(Psi == h)
      lambdaih <- Lambda[h]*dnorm(y[i], X[i,]%*%Beta[,h],
    Sig2[h]^0.5)
      lambdai <- c(lambdai, lambdaih)
    }
    Psi[i] <- sample(1:Hs, 1, prob = lambdai)
  }
  Hs <- length(table(Psi))
  PostBetas[[s]] <- Beta; PostSigma2[[s]] <- Sig2
  PostLambda[[s]] <- Lambda; PostPsi[s, ] <- Psi
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorPsi <- PostPsi[keep,]
Clusters <- sapply(1:length(keep), function(i){length(table(
    PosteriorPsi[i,]))})
NClus <- 4
PosteriorSIGMA <- matrix(NA, length(keep), NClus)
PosteriorBETA <- array(NA, c(k,NClus,length(keep)))
PosteriorLAMBDA <- matrix(NA, length(keep), NClus)
l <- 1
for (s in keep){
  PosteriorSIGMA[l,] <- PostSigma2[[s]][1:NClus]
  PosteriorLAMBDA[l,] <- PostLambda[[s]][1:NClus]
  PosteriorBETA[,,l] <- PostBetas[[s]]
  l <- l + 1
}
summary(coda::mcmc(PosteriorSIGMA)); summary(coda::mcmc(
    PosteriorLAMBDA))
for(l in 1:NClus){
  PosteriorBeta <- t(PosteriorBETA[,l,])
  colnames(PosteriorBeta) <- c("Ct", names(Data[,-1]))
  print(summary(coda::mcmc(PosteriorBeta)))
}
```

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).

Answer:

This mixture model can be expressed in a hierarchical structure:

$$e_i \mid \boldsymbol{\theta}_i \overset{ind}{\sim} f_{\boldsymbol{\theta}_i}$$
$$\boldsymbol{\theta}_i \mid G \overset{iid}{\sim} G$$
$$G \mid \alpha, G_0 \sim DP(\alpha G_0),$$

where $e_i = y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}$ and $\boldsymbol{\theta}_i = [\mu_i \ \sigma_i^2]^\top$.

Thus,

$$s_i \sim \sum_{h=0}^{\infty} \lambda_h \delta_h,$$
$$e_i \mid s_i, \boldsymbol{\theta}_{s_i} \sim N(\mu_{s_i}, \sigma_{s_i}^2),$$

where $\lambda_h = P(\boldsymbol{\theta}_i = \boldsymbol{\theta}_h^*)$.

The conditional posterior distribution of $\boldsymbol{\theta}_i$ is

$$\boldsymbol{\theta}_i | \left\{\boldsymbol{\theta}_{i'}, \boldsymbol{s}_{i'} : i' \neq i\right\}, e_i, \alpha \sim \sum_{i' \neq i} \frac{N_h^{(i)}}{\alpha + N - 1} f_N(e_i|\mu_h, \sigma_h^2) + \frac{\alpha}{\alpha + N - 1} \int_{\mathcal{R}} \int_0^\infty f_N(e_i|\mu, \sigma^2) f_N\left(\mu \middle| \mu_0, \frac{\sigma^2}{\beta_0}\right) f_{IG}(\sigma^2|\alpha_0, \delta_0) d\sigma^2 d\mu,$$

where $N_h^{(i)}$ is the number of observations such that $s_{i'} = h$, $i' \neq i$.

Observe that the integral in the previous equation has exactly the same form as in the marginal likelihood presented in Section 3.2 in the book in the normal-normal/inverse gamma model. Thus,

$$\begin{aligned} p(e_i) &= \int_{\mathcal{R}} \int_0^\infty f_N(e_i|\mu, \sigma^2) f_N\left(\mu \middle| \mu_0, \frac{\sigma^2}{\beta_0}\right) f_{IG}(\sigma^2|\alpha_0, \delta_0) d\sigma^2 d\mu \\ &= \frac{\Gamma\left(\frac{\alpha_n}{2}\right)}{\Gamma\left(\frac{\alpha_0}{2}\right)} \frac{(\delta_0/2)^{\alpha_0/2}}{(\delta_n/2)^{\alpha_n/2}} \left(\frac{\beta_0}{\beta_n}\right)^{1/2} (\pi)^{-1/2}, \end{aligned}$$

where $\alpha_n = 1 + \alpha_0$, $\delta_n = \delta_0 + \frac{\beta_0}{1+\beta_0}(e_i - \mu_0)^2$ and $\beta_n = 1 + \beta_0$. Therefore, we sample s_i as follows,

$$s_i | \left\{\mu_{i'}, \sigma^2_{i'}, \boldsymbol{s}_{i'} : i' \neq i\right\}, e_i, \alpha \sim \left\{ \begin{array}{c} P(s_i = 0|\cdot) = q_0^* \\ P(s_i = h|\cdot) = q_h^*, h = 1, 2, \ldots, H^{(i)} \end{array} \right\},$$

where $H^{(i)}$ is the number of clusters excluding i, which may have its own cluster (singleton cluster), $q_c^* = \frac{q_c}{q_0 + \sum_h q_h}$, $q_c = \{q_0, q_h\}$, $q_h = \frac{N_h^{(i)}}{\alpha + N - 1} f_N(e_i | \mu_h, \sigma_h^2)$ and $q_0 = \frac{\alpha}{\alpha + N - 1} p(e_i)$.

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 μ_h is sample from $N(\mu_i, \sigma_h^2/\beta_n)$, where $\mu_i = \frac{e_i + \beta_0 \mu_0}{1+\beta_0}$.

Discarding $\boldsymbol{\theta}_h$'s from last step, we use $\boldsymbol{s}$ and total number of components to sample σ_h^2 from

$$IG\left(\frac{\alpha_0 + N_h}{2}, \frac{\delta_0 + \sum_{i \in \mathcal{C}_h}(e_i - \bar{e})^2 + \frac{\beta_0 N_h}{\beta_0 + N_h}(\bar{e} - \mu_0)^2}{2}\right),$$

where $\bar{e} = \frac{\sum_{i \in \mathcal{C}_h} e_i}{N_h}$, that is, the mean uses individuals in cluster $\mathcal{C}_h$. Note that when $N_h = 1$, $\delta_0 + \sum_{i \in \mathcal{C}_h}(e_i - \bar{e})^2 + \frac{\beta_0 N_h}{\beta_0 + N_h}(\bar{e} - \mu_0)^2$ is equal to δ_n.

We sample $\mu_h \mid \sigma_h^2, \boldsymbol{e} \sim N\left(\mu_{hn}, \frac{\sigma_h^2}{\beta_0 + N_h}\right)$, where $\mu_{hn} = \frac{\beta_0 \mu_0 + N_h \bar{e}}{\beta_0 + N_h}$, $h = 1, 2, \ldots, H$. Note that μ_{hn} is equal to μ_i when $N_h = 1$.

The posterior distribution of $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_n, \boldsymbol{B}_n),$$

where $\boldsymbol{B}_n = (\boldsymbol{B}_0^{-1} + \sum_{h=1}^{H} \sum_{i \in \mathcal{C}_h} \sigma_h^{-2} \boldsymbol{x}_i \boldsymbol{x}_i^\top)^{-1}$ and $\boldsymbol{\beta}_n = \boldsymbol{B}_n(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{h=1}^{H} \sum_{i \in \mathcal{C}_h} \sigma_h^{-2} \boldsymbol{x}_i (y_i - \mu_h))$.

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 $\alpha_0 = \delta_0 = 0.01$, $\boldsymbol{\beta}_0 = \mathbf{0}_2$, $\boldsymbol{B}_0 = \boldsymbol{I}_2$, 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.

   Answer:

   The following code implements the Gibbs sampling algorithm to perform inference using the DPM in the semi-parametric simulation. From the summary statistics of the posterior chains, we can see that the 95% credible intervals encompass the population values. Figure 11.2 shows the posterior

density of the intercept associated with the second cluster. We observe bimodality, with the two modes located around the population means of the clusters.

R code. Simulation: Semi-parametric Dirichlet process mixture

```
rm(list = ls())
set.seed(010101)
# Simulate data from a 2-component mixture model
n <- 1000
x1 <- rnorm(n); x2 <- rnorm(n)
B <- c(0.5, -1.2)
X <- cbind(x1, x2)
z <- rbinom(n, 1, 0.3)  # Latent class indicator
mu <- ifelse(z == 0, rnorm(n, -0.5, 0.5), rnorm(n, 1, 0.8))
y <- X%*%B + mu
k <- dim(X)[2]
N <- dim(X)[1]
# Hyperparameters
a0 <- 0.001; d0 <- 0.001
b0 <- rep(0, k); B0 <- diag(k)
B0i <- solve(B0)
a <- 0.1; b <- 0.1
mu0 <- 0
sig2mu0 <- 10
beta0 <- 0.1
# MCMC parameters
mcmc <- 5000; burnin <- 1000
tot <- mcmc + burnin; thin <- 2
# Gibbs sampling functions
PostSig2 <- function(Xh, yh, Beta){
  Nh <- length(yh)
  if(Nh == 1){
    Xh <- matrix(Xh, k, 1)
    XhB <- t(Xh)%*%Beta
  }else{
    Xh <- matrix(Xh, Nh, k)
    XhB <- Xh%*%Beta
  }
  eh <- matrix(yh, Nh, 1) - XhB
  an <- a0 + Nh
  dn <- d0 + sum((eh - mean(eh))^2) + beta0*Nh/(beta0 + Nh)*
    (mean(eh) - mu0)^2
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
Postmu <- function(sig2h, Beta, Xh, yh){
  Nh <- length(yh)
  if(Nh == 1){
    Xh <- matrix(Xh, k, 1)
    XhB <- t(Xh)%*%Beta
  }else{
    Xh <- matrix(Xh, Nh, k)
    XhB <- Xh%*%Beta
  }
  eh <- matrix(yh, Nh, 1) - XhB
  sig2mu <- sig2h/(beta0 + Nh)
  mun <- (beta0*mu0 + Nh*mean(eh))/(beta0 + Nh)
  mu <- rnorm(1, mun, sig2mu^0.5)
  return(mu)
}
```

R code. Simulation: Semi-parametric Dirichlet process mixture

```
PostBeta <- function(sig2, mu, X, y, Clust){
  XtX <- matrix(0, k, k); Xty <- matrix(0, k, 1)
  Hs <- length(mu)
  for(h in 1:Hs){
    idh <- which(Clust == h)
    if(length(idh) == 1){
      Xh <- matrix(X[idh,], k, 1); XtXh <- sig2[h]^(-1)*Xh%*%t(Xh)
      yh <- y[idh]; Xtyh <- sig2[h]^(-1)*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)
}
PostAlpha <- function(Clust, alpha){
  H <- length(unique(Clust))
  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, Beta){
  xh <- matrix(xh, k, 1); ei <- c(yh - t(xh)%*%Beta)
  betan <- beta0 + 1; an <- a0 + 1
  dn <- d0 +beta0/(beta0 + 1)*(ei - mu0)^2
  logpy <- (1/2)*log(1/pi)+(a0/2)*log(d0)-(an/2)*log(dn) +
    0.5*log(beta0/betan) + lgamma(an/2)-lgamma(a0/2)
  return(logpy)
}
PostS <- function(Beta, SIGMA, MU, Alpha, Clust, i){
  Nl <- table(Clust[-i]); H <- length(Nl)
  ei <- c(y[i] - t(X[i,])%*%Beta)
  qh <- sapply(1:H, function(h){(Nl[h]/(N+Alpha-1))*dnorm(ei
    , mean = MU[h], sd = SIGMA[h])})
  q0 <- (Alpha/(N+Alpha-1))*exp(LogMarLikLM(xh = X[i,], yh =
     y[i], Beta = Beta))
  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], Beta = Beta)
    SIGMA <- c(SIGMA, Sig2New^0.5)
    MuNew <- Postmu(sig2h = Sig2New, Beta = Beta, Xh = X[i
    ,], yh = y[i])
    MU <- cbind(MU, MuNew)
  }else{si == si}
  return(list(si = si, MU = MU, SIGMA = SIGMA))
}
```

R code. Simulation: Semi-parametric Dirichlet process mixture

```
PostBetas <- matrix(NA, tot, k)
PostSigma <- list(); PostMu <- list()
Posts <- matrix(0, tot, N); PostAlphas <- rep(0, tot)
Clust <- sample(1:3, N, replace=T, prob = c(0.5, 0.3, 0.2))
Reg <- lm(y ~ X)
SumReg <- summary(Reg)
BETA <- Reg$coefficients[-1]
SIGMA <- rep(SumReg$sigma, 3)
MU <- rep(Reg$coefficients[1], 3)
Alpha <- rgamma(1, a, b)
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  for(i in 1:N){
    Rests <- PostS(Beta = BETA, SIGMA = SIGMA, MU = MU,
    Alpha = Alpha, Clust = Clust, i = i)
    Clust[i] <- Rests$si
    MU <- Rests$MU; SIGMA <- Rests$SIGMA
  }
  sFreq <- table(Clust)
  lt <- 1
  for(li in as.numeric(names(sFreq))){
    Index <- which(Clust == li)
    if(li == lt){Clust[Index] <- li
    } else {Clust[Index] <- lt
    }
    lt <- lt + 1
  }
  Alpha <- PostAlpha(Clust = Clust, alpha = Alpha)
  Nl <- table(Clust); H <- length(Nl)
  SIGMA <- rep(NA, H)
  MU <- rep(NA, H)
  l <- 1
  for(h in unique(Clust)){
    Idh <- which(Clust == h)
    SIGMA[l] <- (PostSig2(Beta = BETA, Xh = X[Idh, ], yh = y
    [Idh]))^0.5
    MU[l] <- Postmu(Beta = BETA, sig2h = SIGMA[l]^2, Xh = X[
    Idh, ], yh = y[Idh])
    l <- l + 1
  }
  BETA <- PostBeta(sig2 = SIGMA, mu = MU, X = X, y = y,
    Clust = Clust)
  PostBetas[s,] <- BETA
  PostSigma[[s]] <- SIGMA
  PostMu[[s]] <- MU
  Posts[s, ] <- Clust
  PostAlphas[s] <- Alpha
  setTxtProgressBar(pb, s)
}
close(pb)
```

R code. Simulation: Semi-parametric Dirichlet process mixture

```
keep <- seq(burnin, tot, thin)
PosteriorBetas <- coda::mcmc(PostBetas[keep,])
summary(PosteriorBetas)
plot(PosteriorBetas)
PosteriorPsi <- Posts[keep,]
Clusters <- sapply(1:length(keep), function(i){length(table(
    PosteriorPsi[i,]))})
table(Clusters)
PosteriorSIGMA <- matrix(NA, length(keep), 2)
PosteriorMU <- matrix(NA, length(keep), 2)
l <- 1
for (s in keep){
  PosteriorSIGMA[l,] <- PostSigma[[s]][1:2]
  PosteriorMU[l,] <- PostMu[[s]][1:2]
  l <- l + 1
}
summary(coda::mcmc(PosteriorSIGMA))
summary(coda::mcmc(PosteriorMU))
plot(coda::mcmc(PosteriorMU))
DataMU2 <- data.frame(Posterior = PosteriorMU[,2])
library(ggplot2)
densMU <- ggplot(DataMU2, aes(x = Posterior)) + geom_density
    (fill = "blue", alpha = 0.3) + labs(x = "Intercept", y =
     "Density") + theme_minimal()
densMU
```

5. **Example: Simulation exercise continues**

   Fix the label-switching problem of the simulation exercise of the DPM using *random permutation of latent classes*.

   Answer:

   A simple strategy to address label switching in this simulation exercise is to fix the number of clusters to three, given that this is the typical number of clusters obtained after performing the DPM. Recall that, according to the simulation setup, there are only two clusters:

$$p(y_i \mid \boldsymbol{x}_i) = 0.5\phi(y_i \mid 2 + 1.5x_i, 1^2) + 0.5\phi(y_i \mid -1 + 0.5x_i, 0.8^2).$$

   Thus, we can use a finite Gaussian mixture model with $H = 3$.[1]

[1] Another approach is to use finite Dirichlet processes [28, Chap. 2] and set a random permutation of latent classes.

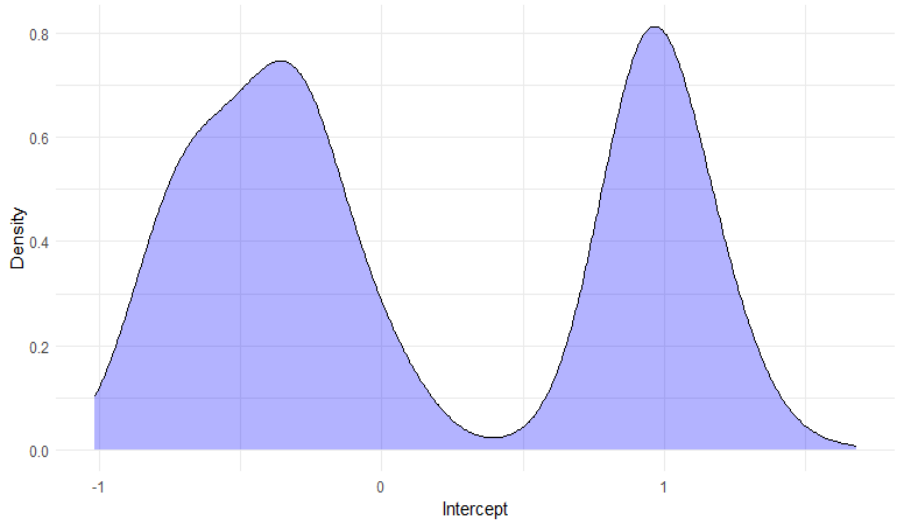

FIGURE 11.2
Posterior density estimate: Mean of the stochastic errors.

The following code illustrates the algorithm.

R code. Simulation: Random permutation of latent classes

```
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))
data <- data.frame(y, x)
# Hyperparameters
d0 <- 0.001; a0 <- 0.001
b0 <- rep(0, 2); B0 <- diag(2); B0i <- solve(B0)
H <- 3; a0h <- rep(1/H, H)
# MCMC parameters
mcmc <- 10000; burnin <- 2000
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, tot, 2)
PostBetas2 <- matrix(0, tot, 2)
PostBetas3 <- matrix(0, tot, 2)
PostSigma2 <- matrix(0, tot, H)
PostPsi <- matrix(0, tot, n)
PostLambda <- matrix(0, tot, H)
Reg <- lm(y ~ x)
SumReg <- summary(Reg)
BETA <- cbind(Reg$coefficients, Reg$coefficients, Reg$
    coefficients)
SIG2 <- rep(SumReg$sigma, H)
X <- cbind(1, x); Psi <- rep(NA, n)
Lambda1 <- 1/H; Lambda2 <- 1/H; Lambda3 <- 1/H
perutatms <- gtools::permutations(3, 3, 1:H)
```

R code. Simulation: Random permutation of latent classes

```
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  for(i in 1:n){
    lambdai1 <- Lambda1*dnorm(y[i], X[i,]%*%BETA[,1], SIG2
    [1]^0.5)
    lambdai2 <- Lambda2*dnorm(y[i], X[i,]%*%BETA[,2], SIG2
    [2]^0.5)
    lambdai3 <- Lambda3*dnorm(y[i], X[i,]%*%BETA[,3], SIG2
    [3]^0.5)
    Psi[i] <- sample(1:H, 1, prob = c(lambdai1, lambdai2,
    lambdai3))
  }
  NH <- NULL
  for(h in 1:H){
    Idh <- which(Psi == h)
    Nh <- length(Idh)
    NH <- c(NH, Nh)
    SIG2[h] <- PostSig2(Betah = BETA[,h], Xh = X[Idh, ], yh
    = y[Idh])
    BETA[,h] <- PostBeta(sig2h = SIG2[h], Xh = X[Idh, ], yh
    = y[Idh])
  }
  Lambda <- sort(MCMCpack::rdirichlet(1, a0h + NH),
    decreasing = TRUE)
  IndPer <- sample(1:dim(perutatms)[1], 1)
  Clust <- perutatms[IndPer, ]
  SIG2 <- SIG2[Clust]
  BETA <- BETA[,Clust]
  Lambda <- Lambda[Clust]
  for(i in 1:n){
    if(Psi[i] == 1){
      Psi[i] <- Clust[1]
    }else{
      if(Psi[i] == 2){
        Psi[i] <- Clust[2]
      }else{
        Psi[i] <- Clust[3]
      }
    }
  }
  Lambda1 <- Lambda[1]; Lambda2 <- Lambda[2]; Lambda3 <-
    Lambda[3]
  PostPsi[s, ] <- Psi
  PostSigma2[s, ] <- SIG2
  PostBetas1[s,] <- BETA[,1]
  PostBetas2[s,] <- BETA[,2]
  PostBetas3[s,] <- BETA[,3]
  PostLambda[s,] <- Lambda
  setTxtProgressBar(pb, s)
}
close(pb)
```

R code. Simulation: Random permutation of latent classes

```
keep <- seq((burnin+1), tot, thin)
PosteriorBetas1 <- coda::mcmc(PostBetas1[keep,])
PosteriorBetas2 <- coda::mcmc(PostBetas2[keep,])
PosteriorBetas3 <- coda::mcmc(PostBetas3[keep,])
PosteriorSigma2 <- coda::mcmc(PostSigma2[keep,])
summary(PosteriorSigma2)
PosteriorLambda <- coda::mcmc(PostLambda[keep,])
summary(PosteriorLambda)
DataBetas <- data.frame(Cluster1 = as.numeric(
    PosteriorBetas1[,2]), Cluster2 = as.numeric(
    PosteriorBetas2[,2]), Cluster3 = as.numeric(
    PosteriorBetas3[,2]))
library(ggplot2)
dens1 <- ggplot(DataBetas, aes(x = Cluster1)) + geom_density
    (fill = "blue", alpha = 0.3) + labs(x = "Cluster 1", y =
     "Density") + theme_minimal()
dens2 <- ggplot(DataBetas, aes(x = Cluster2)) + geom_density
    (fill = "blue", alpha = 0.3) +  labs(x = "Cluster 2", y
    = "Density") + theme_minimal()
dens3 <- ggplot(DataBetas, aes(x = Cluster3)) + geom_density
    (fill = "blue", alpha = 0.3) +  labs(x = "Cluster 3", y
    = "Density") + theme_minimal()
library(ggpubr)
ggarrange(dens1, dens2, dens3, labels = c("A", "B", "C"),
    ncol = 3, nrow = 1, legend = "bottom", common.legend =
    TRUE)

```

Figure 11.3 shows the posterior density estimates of the three clusters. We observe that they have similar shapes, as they are targeting the same posterior distributions. There are two modes with large values around the population parameters, and an additional mode around 0. This occurs because we are forcing three clusters, even though the data-generating process contains only two.

6. **Example: Simulation exercise continues II**

   Obtain the density estimate of y in the simulation exercise of the DPM, evaluated at $x = 0$.

   Answer:

   The following code shows how to perform inference, and get the density estimate using the DPM. Note that the density estimate is given by

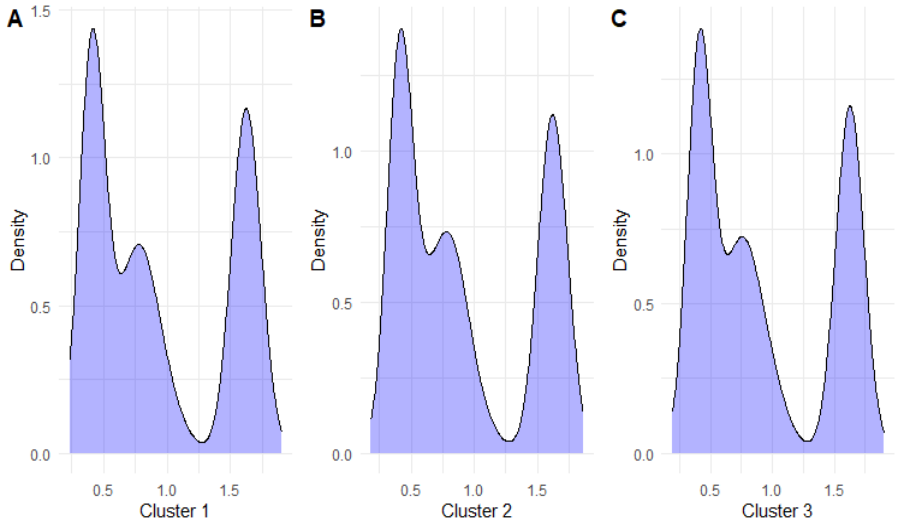

FIGURE 11.3
Posterior density estimates slope parameter: Population values are 1.5 and 0.5.

$$y \mid \boldsymbol{\theta}_h, \boldsymbol{s}, \alpha \sim \sum_{h=1}^{H} \frac{N_h}{\alpha + N + 1} f_N(y|\boldsymbol{x}_h^\top \boldsymbol{\beta}_h, \sigma_h^2) + \frac{\alpha}{\alpha + N + 1} p(y),$$

where N_h is the number of individuals in cluster h, H is the total observed clusters after estimation, and $p(y)$ is the marginal likelihood given in Section 11.1.2 in the book.

Figure 11.4 shows the density plots, population (black line), mean (blue line), and 95% credible interval (shaded blue). We observe that the DPM performs very well despite having more clusters than the actual data generating process.

R code. Simulation: Density from Dirichlet process mixture

```
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: Density from 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
  # 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: Density from Dirichlet process mixture

```
pb <- txtProgressBar(min = 0, max = tot, style = 3)
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
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorS<- Posts[keep,]
Clusters <- sapply(1:length(keep), function(i){length(table(
    PosteriorS[i,]))})
table(Clusters)
Clusters <- sapply(1:length(keep), function(i){table(
    PosteriorS[i,])})
PosteriorBeta <- PostBetas[keep]
PosteriorSigma <- PostSigma[keep]
PosteriorAlpha <- PostAlphas[keep]
```

R code. Simulation: Density from Dirichlet process mixture

```
DensityFunc <- function(yobs = 0, xobs = c(1, 0), s){
  dens <- NULL
  for(j in 1:length(Clusters[[s]])){
    densj <- Clusters[[s]][j]/(PosteriorAlpha[s] + N + 1) *
    dnorm(yobs, t(PosteriorBeta[[s]][,j])%*%xobs,
    PosteriorSigma[[s]][j])
    dens <- c(dens, densj)
  }
  dens <- c(dens, PosteriorAlpha[s]/(PosteriorAlpha[s] + N +
     1)*exp(LogMarLikLM(yh = yobs, xh = xobs)))
  return(sum(dens))
}
xobs <- c(1, 0)
ys <- seq(-3, 5, 0.05)
pdfy <- 0.5*dnorm(ys, 2 + 1.5*xobs[2], 1) + 0.5*dnorm(ys, -1
     + 0.5*xobs[2], 0.8)
DensEval <- matrix(NA, length(keep), length(ys))
for(r in 1:length(ys)){
  for(l in 1:length(keep)){
    DensEval[l, r] <- DensityFunc(yobs = ys[r], xobs = xobs,
     s = l)
  }
}
library(dplyr)
library(ggplot2)
require(latex2exp)
DataDens <- tibble(t = ys, Pop = pdfy, lower = apply(
    DensEval, 2, quantile, probs = 0.025), upper = apply(
    DensEval, 2, quantile, probs = 0.975), meanT = colMeans(
    DensEval))
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 = Pop, color = "True Value"), linewidth
     = 0.5) + geom_line(aes(y = meanT, color = "Estimate"),
    linewidth = 0.5) + scale_color_manual(values = c("True
    Value" = "black", "Estimate" = "blue")) + xlab(TeX("y"))
     + ylab("Density") + labs(title = "Density: Dependent
    variable", color = "") + theme_minimal()
  print(p)
}
plot_filtering_estimates(DataDens)
```

7. **Example: Consumption of marijuana in Colombia continues II**

   Perform the application of marijuana consumption with the following spec-

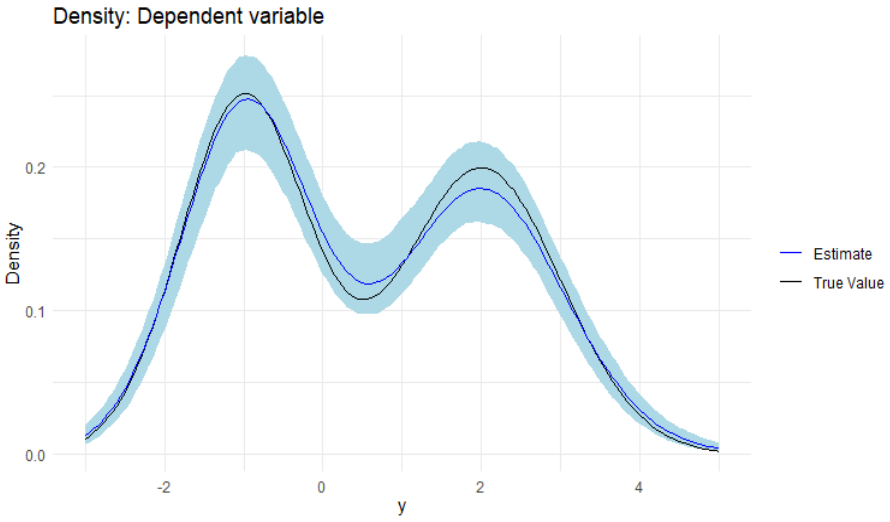

FIGURE 11.4
Density from Dirichlet process mixture.

ification:

$$y_i = \boldsymbol{z}_i^\top \boldsymbol{\gamma} + f(Age_i) + \mu_i,$$

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?

Answer:

The following code illustrates this exercise. We use the function *bicreg* from the package *BMA* to perform variable selection. Initially, we construct the design matrix, excluding the B-splines that cause perfect multicollinearity, and then select the variables with a posterior inclusion probability greater than 50%. This also corresponds to the model with the highest posterior probability.

After this, we proceed with inference using a linear regression with conjugate prior distributions. Figure 11.5 shows the fit, with the mean (blue line) and 95% credible intervals (shaded blue). This curve is simpler than the one using all the B-splines: there appears to be a hook-shaped pattern at the beginning, where teenagers consume less marijuana than individuals in their 20s to 50s. After age 55, consumption decreases significantly. Given this pattern, assuming a second-degree polynomial appears to be a reasonable choice.

R code. Simulation: Semi-parametric Dirichlet process mixture

```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)))
# Function to check if a column is constant
is_constant <- function(col) {
  return(length(unique(col)) == 1)
}
constant_columns <- apply(BS, 2, is_constant)
X <- cbind(BS[,-c(which(constant_columns == TRUE))], Z)
BMAglm <- BMA::bicreg(X, y, strict = FALSE, OR = 50)
summary(BMAglm)
idReg <- c(2, 23, 24, 27:29) # Relevant regressors, PIP >
    0.5
Xnew <- cbind(1, X[, idReg])
N <- dim(Xnew)[1]; k <- dim(Xnew)[2]
# 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~Xnew-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 <- 675 # Pick a particular individual
idfit <- sample(1:N, H)
BSfit <- bs(xfit, knots = knots, degree = 3, Boundary.knots
    = range(x), intercept = FALSE)
Xfit <- cbind(1, BSfit[,c(2,23)], Z[rep(i, H),c(1,4:6)]) #
    Relevant regressors, PIP > 0.5
Fit <- matrix(NA, mcmc, H)
for(s in 1:mcmc){
  Fit[s,] <- Xfit%*%posterior[s,1:7]
}
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()
```

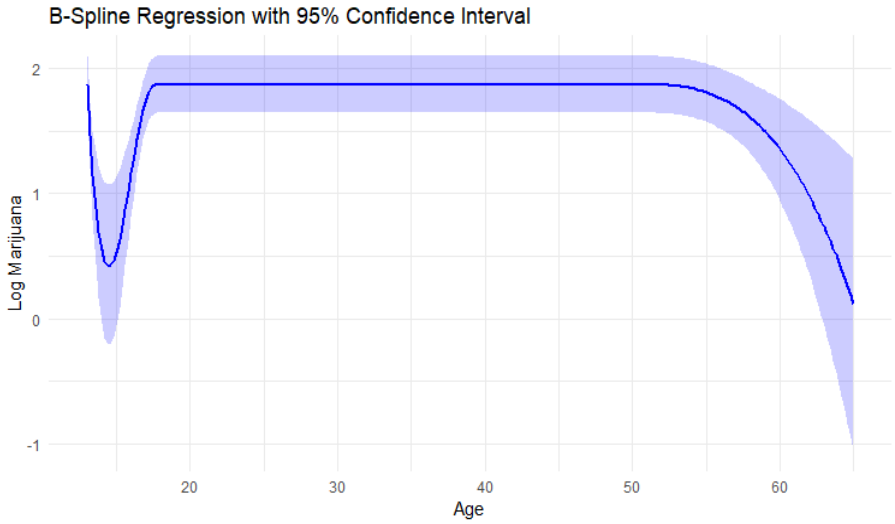

FIGURE 11.5
Spline: Marijuana monthly consumption vs age.

12

Bayesian machine learning

Solutions of 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.

   Answer

   The following code illustrates how to implement the Gibbs sampler for the Bayesian LASSO. Overall, the results are very similar to those obtained using the *monomvn* package (see Figure 12.1).

R code. Simulation: The Bayesian LASSO

```
rm(list = ls()); set.seed(10101)
library(monomvn)
# 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 monomvn ###
library(monomvn)
library(statmod)
# Fit model using Bayesian LASSO
fit <- monomvn::blasso(X, y, T = 5000, verb = 0)
# burn-in removal
burnin <- 1000
beta_samples <- fit$beta[(burnin + 1):5000, ]  # exclude
    burn-in
intercept_samples <- fit$mu[(burnin + 1):5000] # intercept
    posterior draws
# Posterior mean of coefficients
beta_post_mean <- colMeans(beta_samples)
# Intercept summary
mean(intercept_samples)
quantile(intercept_samples, c(0.025, 0.975))
# Local shrinkage parameters
colMeans(is.na(fit$tau2i))  # proportion of NA values per
    coefficient

#### Gibss sampler ####
# Hyperparameters
r0 <- 1; d0 <- 1
# MCMC parameters
mcmc <- 5000; burnin <- 5000
tot <- mcmc + burnin; thin <- 1
# Posterior distributions programming the Gibbs sampling
# Standardized variables
W <- scale(X)
yh <- y - mean(y)
# Gibbs sampling functions
PostBeta <- function(sig2, tau2){
  Bn <- solve(diag(1/tau2) + t(W)%*%W)
  bn <- Bn%*%t(W)%*%yh
  Beta <- MASS::mvrnorm(1, bn, sig2*Bn)
  return(Beta)
}
```

R code. Simulation: The Bayesian LASSO

```
PostSig2 <- function(Beta, tau2){
  an <- n - 1 + k
  dn <- t(yh - W%*%Beta)%*%(yh - W%*%Beta) + t(Beta)%*%diag
    (1/tau2)%*%Beta
  sig2 <- invgamma::rinvgamma(1, shape = an/2, rate = dn/2)
  return(sig2)
}
PostTau2 <- function(sig2, Beta, lambda, p){
  mukn <- (lambda^2*sig2/Beta[p]^2)^0.5
  lambdan <- lambda^2
  tauk2k <- rinvgauss(1, mean = mukn, shape = lambdan)
  return(tauk2k)
}
PostLambda2 <- function(tau2){
  sh <- k + r0
  rat <- 0.5 * sum(tau2) + d0
  lambda2 <- rgamma(1, shape = sh, rate = rat)
  return(lambda2^0.5)
}
PostBeta0 <- function(sig2){
  beta0 <- rnorm(1, mean = mean(y), sd = (sig2/n)^0.5)
  return(beta0)
}
PostBetas <- matrix(0, mcmc+burnin, k)
PostSigma2 <- rep(0, mcmc+burnin)
PostTaus2 <- matrix(0, mcmc+burnin, k)
PostLambda <- rep(0, mcmc+burnin)
PostBetas0 <- rep(0, mcmc+burnin)
Beta <- rep(0, k); tau2 <- rep(1, k); lambda <- 1
# create progress bar in case that you want to see
    iterations progress
pb <- txtProgressBar(min = 0, max = tot, style = 3)
for(s in 1:tot){
  sig2 <- PostSig2(Beta = Beta, tau2 = tau2)
  PostSigma2[s] <- sig2
  Beta <- PostBeta(sig2 = sig2, tau2 = tau2)
  PostBetas[s,] <- Beta
  tau2 <- sapply(1:k, function(i){PostTau2(sig2 = sig2, Beta
     = Beta, lambda = lambda, p = i)})
  PostTaus2[s,] <- tau2
  lambda <- PostLambda2(tau2 = tau2)
  PostLambda[s] <- lambda
  beta0 <- PostBeta0(sig2 = sig2)
  PostBetas0[s] <- beta0
  setTxtProgressBar(pb, s)
}
close(pb)
keep <- seq((burnin+1), tot, thin)
PosteriorBetas <- PostBetas[keep,]
summary(coda::mcmc(PosteriorBetas))
PosteriorBeta0 <- PostBetas0[keep]
summary(coda::mcmc(PosteriorBeta0))
PosteriorSigma2 <- PostSigma2[keep]
summary(coda::mcmc(PosteriorSigma2))
PosteriorTau2s <- PostTaus2[keep,]
summary(coda::mcmc(PosteriorTau2s))
PosteriorLambda <- PostLambda[keep]
summary(coda::mcmc(PosteriorLambda))
```

R code. Simulation: The Bayesian LASSO

```
# Posterior mean of coefficients
beta_post_meanNew <- colMeans(PosteriorBetas)
# Plot true vs estimated betas
plot(beta_true, beta_post_mean, pch = 19, col = "steelblue",
xlab = "True beta", ylab = "Posterior mean of beta",
main = "Bayesian LASSO (blasso): Shrinkage Effect")
abline(0, 1, col = "red", lty = 2)
# Add another set of posterior means
points(beta_true, beta_post_meanNew, pch = 17, col = "
    darkorange")
# Add legend
legend("topleft", legend = c("blasso", "Our Gibbs"),
col = c("steelblue", "darkorange"),
pch = c(19, 17), bty = "n")

```

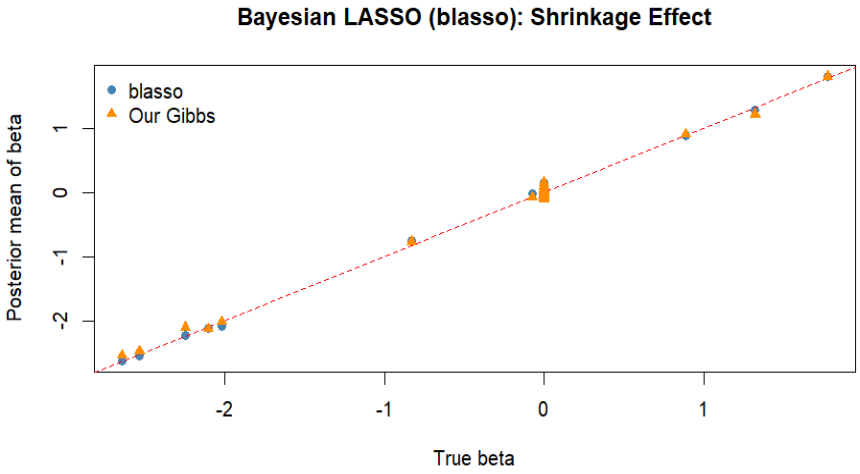

FIGURE 12.1
Comparison of LASSO performance: Package vs Gibbs from scratch.

2. [19] employ stochastic search variable selection (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.xlsx*, 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.

Answer

The following code illustrates how to implement SSVS in this application, based on [19], using 15,000 iterations and the default settings for the hyperparameters. We find that infant mortality rate, population, peace, and political rights are the most relevant drivers of the incidence of civil conflict. *Peace* exhibits a negative 95% credible interval, while the other variables have positive credible intervals. The model with the highest posterior probability (30%) includes the first three regressors.

R code. Application: Drivers of civil conflict in the post-Cold War era using a linear model

```
rm(list = ls())
set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/
    BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/
    Conflict.csv", sep = ",", header = TRUE, quote = "")

# Scale regressors
W <- as.matrix(scale(Data[, -c(1, 2)]))
niter <- 15000
y <- unlist(Data[, 2])

# Linear model
SSBoomLinear <- lm.spike(y ~ W, niter = niter)
Models <- SSBoomLinear$beta != 0
PIP <- colMeans(SSBoomLinear$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))
colnames(W)
# Get indices of best model
for(l in 1:5) {
  print(which(row_counts[l,] == 1))
  print(row_counts[l, dim(row_counts)[2]]/niter)
}
# Coefficients
SummarySS <- summary(coda::mcmc(SSBoomLinear$beta))
SummarySS
```

3. [46] proposes an SSVS approach for binary response models. Use the dataset *Conflict.xlsx*, 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.

Answer

The following code illustrates how to implement SSVS in this application, based on [19], using 15,000 iterations and the default settings for the hyperparameters with a logit model.

We find that the results using a logit model are very similar to those ob-

tained with a linear model. Once again, infant mortality rate, population, peace, and political rights emerge as the most relevant drivers of the incidence of civil conflict. *Peace* exhibits a negative 95% credible interval, while the other variables have positive credible intervals. In this case, the model with the highest posterior probability (35%) includes infant mortality rate and peace. However, the model with the second-highest posterior probability (17%) includes the first three drivers.

R code. Application: Drivers of civil conflict in the post-Cold War era using a logit model

```
rm(list = ls())
set.seed(010101)
Data <- read.csv("https://raw.githubusercontent.com/
    BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/
    Conflict.csv", sep = ",", header = TRUE, quote = "")

# Scale regressors
W <- as.matrix(scale(Data[, -c(1, 2)]))
niter <- 15000
y <- unlist(Data[, 2])

# Logit model
SSBoomLogit <- logit.spike(y ~ W, niter = niter)
ModelsLogit <- SSBoomLogit$beta != 0
PIPLogit <- colMeans(SSBoomLogit$beta != 0)
# Convert the logical matrix to a data frame and then to a
    tibble
df <- as.data.frame(ModelsLogit); df_tbl <- as_tibble(df)
# Count identical rows
row_counts <- df_tbl %>% count(across(everything()), name =
    "frequency") %>% arrange(desc(frequency))
colnames(W)
# Get indices of best model
for(l in 1:5) {
  print(which(row_counts[l,] == 1))
  print(row_counts[l, dim(row_counts)[2]]/niter)
}
# Coefficients
SummarySSlogit <- summary(coda::mcmc(SSBoomLogit$beta))
SummarySSlogit

```

4. **Example: Simulation exercise $K > N$**

   Use the simulation setting from the Bayesian LASSO and SSVS examples, but now assume there are 600 features. This setup implies that the number

of features 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.

Answer

The following code shows how to perform this exercise using the *bayesreg* and *BoomSpikeSlab* packages. We observe that both the Bayesian LASSO and SSVS perform well (see Figure 12.2) and allow for valid inference.

R code. Simulation: Performance of the Bayesian LASSO and SSVS when $K > N$

```r
rm(list = ls()); set.seed(10101)
library(BoomSpikeSlab)
library(dplyr)
library(tibble)
library(bayesreg)
# Parameters
n <- 500  # sample size
k <- 600  # 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)
niter <- 5000
######Estimate model########
# Least squared
Reg <- lm(y ~ X)
summary(Reg)
## BLASSO ##
# Fit the model
fit <- bayesreg::bayesreg(y ~ X, data = df, model = "gaussian", prior = "lasso",
n.samples = niter, burnin = 1000)
# Check summary
summary(fit)
# Extract posterior means of beta
beta_post_meanLASSO <- rowMeans(fit$beta)

## SSVS ##
SSBoomNew <- lm.spike(yh ~ W - 1, niter = niter)
# Coefficients
SummarySS <- summary(coda::mcmc(SSBoomNew$beta))
# Extract posterior means of beta
beta_post_meanSSVS <- SummarySS$statistics[, 1]

# Compare true vs estimated
plot(beta_true, beta_post_meanLASSO, pch = 19, col = "steelblue",
xlab = "True beta", ylab = "Posterior mean of beta",
main = "Bayesian LASSO and SSVS")
abline(0, 1, col = "red", lty = 2)
# Add another set of posterior means
points(beta_true, beta_post_meanSSVS, pch = 17, col = "darkorange")
# Add legend
legend("topleft", legend = c("BLASSO", "SSVS"),
col = c("steelblue", "darkorange"),
pch = c(19, 17), bty = "n")
```

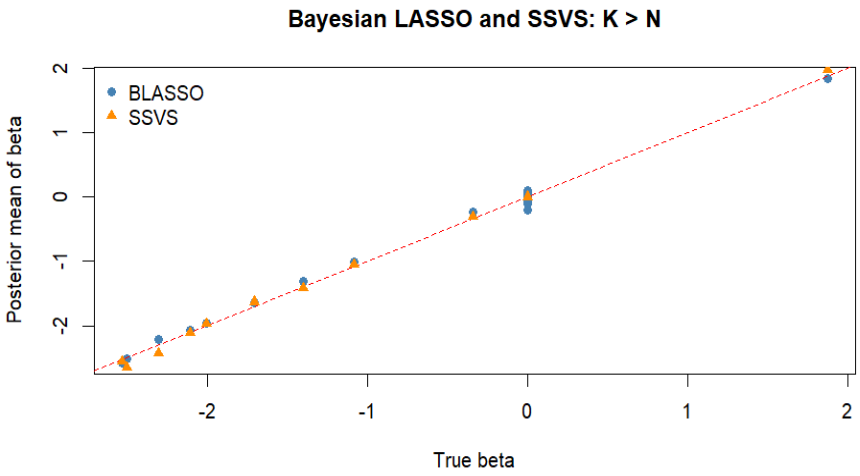

FIGURE 12.2
Performance of the Bayesian LASSO and SSVS: $K > N$.

5. **Simulation exercise: the BART model continues**

   Compute Friedman's partial dependence functions [9] for all variables in the BART model simulation example, and plot the posterior mean along with the 95% credible intervals.

   Answer

   The following code illustrates how to compute the posterior distribution of Friedman's partial dependence functions. Figures 12.3 and 12.4 display the results for variables 1 and 10. From the simulation setup, we know that variable 1 is relevant, whereas variable 10 is irrelevant. This is reflected in the partial dependence functions: the function for variable 10 is flat, indicating no effect, while the function for variable 1 shows a positive relationship with the outcome variable.

R code. Simulation: Partial dependence functions from BART models

```r
rm(list = ls()); set.seed(10101)
library(BART); library(ggplot2); library(dplyr); library(
    tidyr)
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
# Hyperparameters
alpha <- 0.95; beta <- 2; k <- 2
v <- 3; q <- 0.9; J <- 200
# MCMC parameters
MCMCiter <- 1000; burnin <- 100; thinning <- 1
# Partial dependence function
BARTfit <- wbart(x.train = X, y.train = y, base = alpha,
power = beta, k = k, sigdf = v, sigquant = q, ntree = J,
ndpost = MCMCiter, nskip = burnin, keepevery = thinning)
k <- 1; L <- 20
x <- seq(min(X[, k]), max(X[, k]), length.out = L)
X.test <- X
X.test[,k] <- x[1]
for(j in 2:L){
  X.k <- X
  X.k[,k] <- x[j]
  X.test <- rbind(X.test, X.k)
}
pred <- predict(BARTfit, X.test)
partial <- matrix(nrow = MCMCiter, ncol = L)
for(j in 1:L) {
  h <- (j - 1) * N + 1:N
  partial[, j] <- apply(pred[, h], 1, mean)
}
partial_df <- as.data.frame(partial)
partial_df$draw <- 1:nrow(partial_df)
long_df <- pivot_longer(partial_df, cols = -draw, names_to =
     "x", values_to = "value")
long_df$x <- as.numeric(gsub("V", "", long_df$x))
summary_df <- long_df %>% group_by(x) %>%
summarise(mean = mean(value), lower = quantile(value, 0.025)
    , upper = quantile(value, 0.975)
)
ggplot(summary_df, aes(x = x, y = mean)) + geom_line(color =
     "blue") + geom_ribbon(aes(ymin = lower, ymax = upper),
    fill = "lightblue", alpha = 0.4) + labs(x = "Grid point"
    , y = "Partial effect", title = "Partial Dependence
    Function with 95% Intervals") + theme_minimal()
```

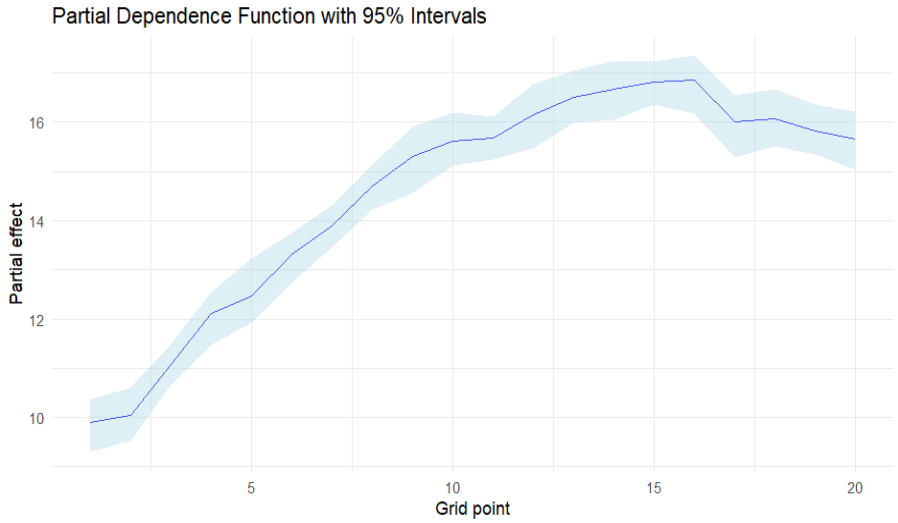

FIGURE 12.3
Partial dependence function: Variable 1.

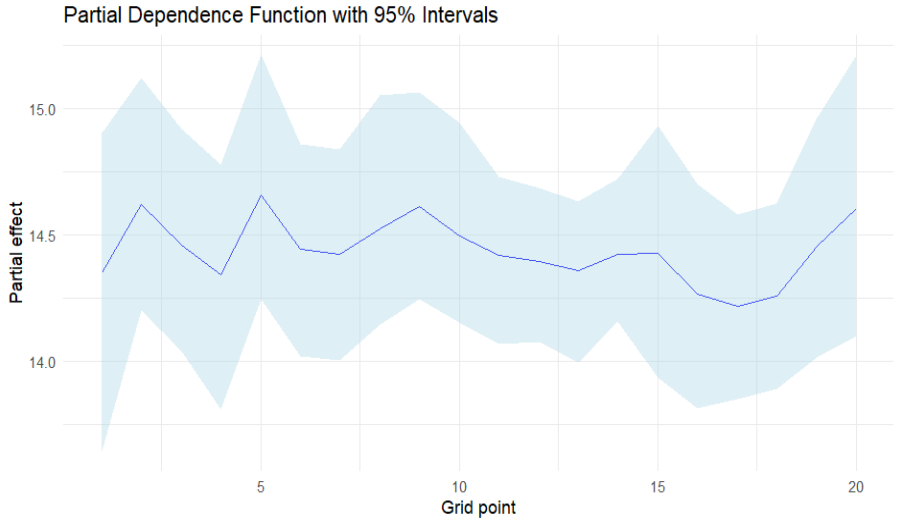

FIGURE 12.4
Partial dependence function: Variable 10.

6. [46] 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.

Answer

The following code demonstrates how to perform probit BART for binary classification. According to the results obtained using 10 trees, the most relevant regressors are *infant mortality rate*, *peace*, *political rights*, and *population*. Note that this agrees with previous exercises using this dataset.

R code. Application: Probit BART for binary classification

```r
rm(list = ls()); set.seed(10101)
library(BART); library(tidyr)
library(pROC); library(caret)
# Load and prepare data
Data <- read.csv("https://raw.githubusercontent.com/
    BEsmarter-consultancy/BSTApp/refs/heads/master/DataApp/
    Conflict.csv", sep = ",", header = TRUE, quote = "")
X <- as.matrix(Data[, -c(1, 2)])
y <- unlist(Data[, 2])
# MCMC parameters
niter <- 1000; burnin <- 200; J <- 200
# Create 5-fold cross-validation indices
folds <- createFolds(y, k = 5, list = TRUE, returnTrain =
    FALSE)
thresholds <- seq(0.01, 0.99, by = 0.01)
scores <- matrix(0, nrow = 5, ncol = length(thresholds))
for (i in 1:5) {
  test_idx <- folds[[i]]; train_idx <- setdiff(1:length(y),
    test_idx)
  X.train <- X[train_idx, ];  y.train <- y[train_idx]
  X.test <- X[test_idx, ]; y.test <- y[test_idx]
  pf <- pbart(x.train = X.train, y = y.train, x.test = X.
    train, ntree = J, ndpost = niter, nskip = burnin)
  prob_train <- pf$prob.test.mean
  for (j in 1:length(thresholds)) {
    thresh <- thresholds[j]
    preds <- ifelse(prob_train > thresh, 1, 0)
    TP <- sum(preds == 1 & y.train == 1)
    TN <- sum(preds == 0 & y.train == 0)
    FP <- sum(preds == 1 & y.train == 0)
    FN <- sum(preds == 0 & y.train == 1)
    TPR <- TP / (TP + FN + 1e-8)
    TNR <- TN / (TN + FP + 1e-8)
    scores[i, j] <- TPR + TNR
  }
}
mean_scores <- colMeans(scores)
best_thresh <- thresholds[which.max(mean_scores)]
cat("Optimal threshold by CV:", best_thresh, "\n")
# Train/test split to evaluate final performance
N <- nrow(X); K <- ncol(X)
N.train <- floor(0.8 * N); X.train <- X[1:N.train, ]; y.
    train <- y[1:N.train]
X.test <- X[(N.train + 1):N, ]; y.test <- y[(N.train + 1):N]
pf_final <- pbart(x.train = X.train, y = y.train, x.test = X
    .test, ntree = 200, ndpost = 1000, nskip = 200)
prob_test <- pf_final$prob.test.mean
y_pred_test <- ifelse(prob_test > best_thresh, 1, 0)
conf_mat <- table(Predicted = y_pred_test, Actual = y.test)
accuracy <- mean(y_pred_test == y.test)
print(conf_mat)
cat("Test set accuracy:", round(accuracy, 3), "\n")
```

R code. Application: Probit BART for binary classification

```r
# Relevant regressors
Js <- c(10, 20, 50, 100, 200)
VarImportance <- matrix(0, length(Js), K)
l <- 1
for (j in Js){
  BARTfit <- BART::pbart(x.train = X.train, y = y.train, x.
    test = X.test, ntree = j, ndpost = niter, nskip = burnin
    )
  VarImportance[l, ] <- BARTfit[["varcount.mean"]]/j
  l <- l + 1
}
# Assign row and column names for clarity
rownames(VarImportance) <- as.character(Js)
colnames(VarImportance) <- as.character(1:K)
# Convert to long format
importance_df <- as.data.frame(VarImportance) %>%
mutate(trees = rownames(.)) %>%
pivot_longer(
cols = -trees,
names_to = "variable",
values_to = "percent_used"
)
# Make variable numeric for plotting
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:K) + labs(x = "variable", y = "percent used", color =
     "#trees", linetype = "#trees") + theme_minimal()

data.frame(Nanes = colnames(X), Frequency = VarImportance
    [1,])
```

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.

Answer

We use the Cholesky decomposition to optimize time calculating the inverse of the covariance matrix. In particular, let $\mathbf{K} \in \mathbb{R}^{N \times N}$ be a symmetric positive definite matrix. Then, it admits a Cholesky decomposition:

$$\mathbf{K} = \mathbf{L}\mathbf{L}^\top,$$

where $\mathbf{L}$ is a lower triangular matrix with positive diagonal entries.

The determinant of $\mathbf{K}$ can be computed using the Cholesky factor:

$$\det(\mathbf{K}) = \det(\mathbf{L}\mathbf{L}^\top) = \det(\mathbf{L})^2.$$

Since $\mathbf{L}$ is lower triangular, its determinant is the product of its diagonal elements:

$$\det(\mathbf{K}) = \left(\prod_{i=1}^{N} L_{ii} \right)^2.$$

Equivalently, the log-determinant is:

$$\log |\mathbf{K}| = 2 \sum_{i=1}^{N} \log L_{ii}.$$

The inverse of $\mathbf{K}$ can also be computed using the Cholesky factor:

$$\mathbf{K}^{-1} = (\mathbf{L}\mathbf{L}^\top)^{-1} = (\mathbf{L}^\top)^{-1}\mathbf{L}^{-1}.$$

This is useful computationally because triangular systems can be solved efficiently via forward and backward substitution:

(a) Solve $\mathbf{L}\mathbf{z} = \mathbf{b}$ for $\mathbf{z}$ (forward solve),

(b) Solve $\mathbf{L}^\top \mathbf{x} = \mathbf{z}$ for $\mathbf{x}$ (backward solve),

then, $\mathbf{x} = (\mathbf{L}^\top)^{-1}\mathbf{z} = (\mathbf{L}^\top)^{-1}\mathbf{L}^{-1}\mathbf{b} = (\mathbf{L}\mathbf{L}^\top)^{-1}\mathbf{b} = \mathbf{K}^{-1}\mathbf{b}$. Thus, we avoid computing the matrix inverse, which requires $O(N^3)$ operations. Instead, it is more efficient and numerically stable to solve triangular systems of equations, which arise naturally from the Cholesky decomposition.

The following code shows how to perform this exercise, and Figure 12.5 displays the 3D plot.

R code. Simulation: $f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2}) + \mu_i$ ***and Gaussian process***

```
rm(list = ls()); set.seed(10101)
library(fields); library(DiceKriging); library(ggplot2)
# Simulate training data
n_train <- 500
x1 <- runif(n_train); x2 <- runif(n_train)
X_train <- data.frame(x1 = x1, x2 = x2)
sig <- 0.1; u <- rnorm(n_train, mean = 0, sd = sig)
# 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)
y_train <- scale(f_train + u)
# Empirical Bayes: Optimize the log marginal likelihood
log_marginal_likelihood <- function(par, X, y) {
  sigma2_f <- par[1]; l2 <- par[2]; sigma2_n <- par[3]; n <-
     nrow(X)
  # Ensure parameters are positive
  if (sigma2_f <= 0 | l2 <= 0 | sigma2_n <= 0) return(Inf)
  # Squared exponential kernel
  dists <- rdist(X)
  K <- sigma2_f * exp(-0.5 * (dists^2) / l2)
  # Add noise variance + jitter for numerical stability
  K <- K + (sigma2_n + 1e-8) * diag(n)
  # Cholesky decomposition
  L <- tryCatch(chol(K), error = function(e) return(NULL))
  if (is.null(L)) return(Inf)
  alpha <- backsolve(t(L), forwardsolve(L, y)) #--> K^{-1}y
  log_det_K <- 2 * sum(log(diag(L)))
  lml <- -0.5 * t(y) %*% alpha - 0.5 * log_det_K - 0.5 * n *
     log(2 * pi)
  return(-as.numeric(lml))  # Negative log-marginal
    likelihood
}
par0 <- rep(1, 3)
ResOpt <- optim(par = par0, fn = log_marginal_likelihood,
    method = "L-BFGS-B", lower = c(1e-5, 1e-5), X = X_train,
     y = y_train)
theta_hat <- ResOpt$par
sigma2_f_hat <- theta_hat[1]; l2_hat <- theta_hat[2]; sigma2
    _n_hat <- theta_hat[3]
# Grid for prediction
grid_points <- 20
x1_seq <- seq(0, 1, length.out = grid_points)
x2_seq <- seq(0, 1, length.out = grid_points)
X_new <- expand.grid(x1 = x1_seq, x2 = x2_seq)
# Fit Gaussian Process
fit_km <- km(design = X_train, response = y_train, covtype =
     "gauss", noise.var = rep(sigma2_n_hat, n_train))
# Predict GP surface
pred <- predict(fit_km, newdata = X_new, type = "UK")
z_pred <- matrix(pred$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 = y_train, col =
    "red", size = 8)
```

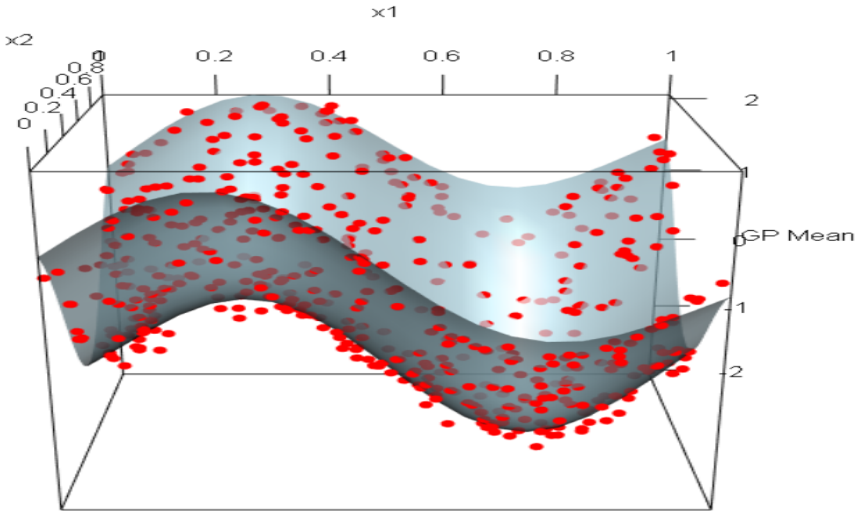

FIGURE 12.5
Simulation: $f_i = \sin(2\pi x_{i1}) + \cos(2\pi x_{i2}) + \sin(x_{i1}x_{i2}) + \mu_i$ and Gaussian process

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}.

   Answer

   The following code illustrates how to implement the Stochastic Gradient Langevin Dynamics (SGLD) algorithm with a control variate. Figure 12.6 shows the posterior distributions obtained from the SGLD algorithm with a control variate and the Metropolis-Hastings algorithm. We observe that the former is well centered, but exhibits overdispersion; however, the control variate helps reduce the variability compared to the naive SGLD algorithm (see Figure 12.10 in the book).

R code. Stochastic Gradient Langevin Dynamic control variate (SGLDcv): Logit regression

```
rm(list = ls()); set.seed(10101)
library(mvtnorm); library(MCMCpack)
library(ggplot2); library(dplyr)
#--- 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)
}
#--- One SGD step
SGD_step <- function(beta, y, X, lr, 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 + beta / prior_var * 1 / N

  beta_new <- beta - lr * grad
  return(beta_new)
}
#--- Parameters
K <- 10; N <- 100000
beta_true <- rep(0.5, K); batch_prop <- 0.01
batch_size <- round(N*batch_prop); n_iter <- 1000
n_iterSGD <- 1500; kappa <- 0.55
stepsize <- 1e-4; prior_var <- 10
k_target <- 5  # beta5

sim_data <- simulate_logit_data(K, N, beta_true)
y <- sim_data$y
X <- scale(sim_data$X)
```

R code. Stochastic Gradient Langevin Dynamic control variate (SGLDcv): Logit regression

```
beta_mat <- matrix(0, n_iterSGD, K)
beta_mat[1, ] <- rep(0, K)
for (s in 2:n_iterSGD) {
  beta_mat[s, ] <- SGD_step(beta_mat[s - 1, ], y = y, X = X,
     lr = s^(-kappa), batch_size = batch_size, prior_var =
    prior_var)
}
matplot(beta_mat, type = "l"); betahat <- beta_mat[s, ]
gradhat <- rep(0, K)
for (i in 1:N) {
  xi <- X[i, ]
  etahat <- sum(xi * betahat)
  pihat <- 1 / (1 + exp(-etahat))
  grad_ihat <- -(y[i] - pihat) * xi
  gradhat <- gradhat + grad_ihat
}
#--- One SGLD control variate step
SGLDcv_step <- function(beta, betahat, gradhat, y, X,
    stepsize, batch_size, prior_var = 10) {
  N <- nrow(X)
  K <- length(beta)
  ids <- sample(1:N, size = batch_size, replace = FALSE)
  grad_dif <- rep(0, K)
  for (i in ids) {
    xi <- X[i, ]
    eta <- sum(xi * beta)
    pi <- 1 / (1 + exp(-eta))
    etahat <- sum(xi * betahat)
    pihat <- 1 / (1 + exp(-etahat))
    grad_dif_i <- (pi - pihat) * xi
    grad_dif <- grad_dif + grad_dif_i
  }
  grad <- gradhat + grad_dif / 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_SGLDcv <- function(y, X, stepsize, batch_prop, n_iter,
    beta_init = NULL, prior_var = prior_var) {
  N <- nrow(X)
  K <- ncol(X)
  batch_size <- round(batch_prop * N)
  beta_mat <- matrix(0, n_iter, K)
  beta_mat[1, ] <- if (is.null(beta_init)) rep(0, K) else
    beta_init
  for (s in 2:n_iter) {
    beta_mat[s, ] <- SGLDcv_step(beta_mat[s - 1, ], betahat
    = betahat, gradhat = gradhat, y, X, stepsize, batch_size
    )
  }
  return(beta_mat)
}
#--- Run SGLD cv
posterior_sgldcv <- run_SGLDcv(y, X, stepsize, batch_prop, n
    _iter)
```

R code. Stochastic Gradient Langevin Dynamic control variate (SGLDcv): Logit regression

```
#--- 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 =
    0.1,
burnin = n_iterSGD, mcmc = n_iter)
#--- Compare densities for beta5
df_plot <- data.frame(
value = c(posterior_sgldcv[, k_target], posterior_mh[, k_
    target]),
method = rep(c("SGLDcv", "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_target], linetype = "dashed", color = "
    black") + labs(title = expression(paste("Posterior
    density of ", beta[k_target])), x = expression(beta[k_
    target]), y = "Density") + theme_minimal() + xlim(0.45,
    0.55)
```

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 $\alpha_0 = \delta_0 = 0.01$.

   Answer

   Take into account that SGLD was originally developed for parameter spaces in $\mathbb{R}^K$. In this exercise, we use SGLD to perform inference on $\boldsymbol{\beta}$, while obtaining draws from the conditional posterior distribution of $\sigma^2 \sim IG(\alpha_n/2, \delta_n/2)$, where $\alpha_n = N + \alpha_0$ and $\delta_n = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \delta_0$.

   Additionally, recall the gradients involved in the SGLD updates:

$$\nabla_{\boldsymbol{\beta}} \log p(\mathbf{y} \mid \boldsymbol{\beta}, \sigma^2) = \frac{1}{\sigma^2} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}),$$
$$\nabla_{\boldsymbol{\beta}} \log \pi(\boldsymbol{\beta}) = -\boldsymbol{\beta}.$$

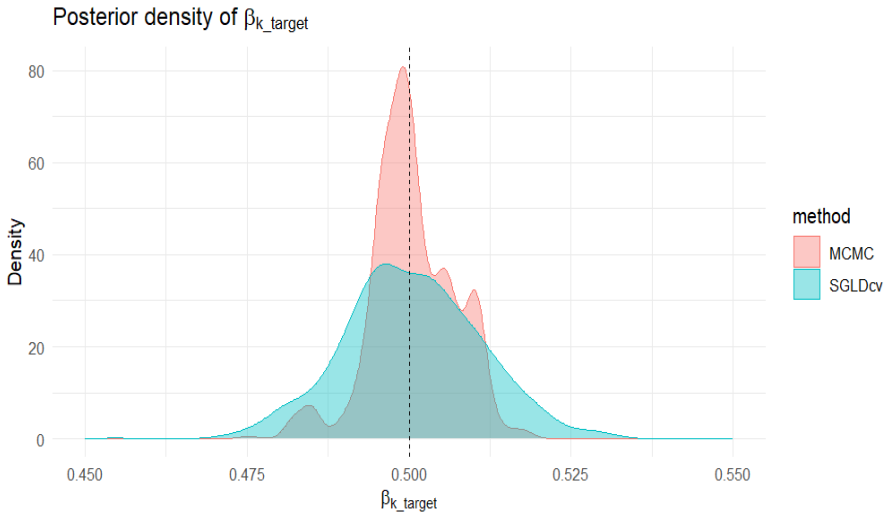

FIGURE 12.6
Simulation: Posterior distributions from logit simulation exercise

The following code illustrates how to implement this procedure. We note that the results for the location parameters $\boldsymbol{\beta}$ are satisfactory; however, the algorithm appears to exhibit a positive bias in the estimation of σ^2.

R code. Stochastic Gradient MCMC (SG-MCMC): Linear regression

```
rm(list = ls()); set.seed(10101)
# Simulated data
N <- 100000; K <- 3
X <- cbind(1, matrix(rnorm(N * (K - 1)), N, K - 1))  # design matrix
beta_true <- c(1, -2, 0.5); sigma2_true <- 1
y <- X %*% beta_true + rnorm(N, 0, sqrt(sigma2_true))
X[, 2:3] <- scale(X[, 2:3])
# SGLD settings
iterations <- 5000
batch_size <- 1000
initial_stepsize <- 1e-4
decay_rate <- 0.55
schedule <- 1000
# Priors
a0 <- 0.01; b0 <- 0.01
# Storage
trace_beta <- matrix(NA, nrow = iterations, ncol = K)
trace_sigma2 <- rep(NA, iterations)
# Initialization
beta <- rep(0, K)
sigma2 <- 0.5
# Gradient of log-posterior w.r.t. beta (scaled)
grad_log_post <- function(beta, X_batch, y_batch, N, sigma2) {
  grad_loglik <- t(X_batch) %*% (y_batch - X_batch %*% beta) / sigma2
  grad_logprior <- -beta  # derivative of log N(0, I)
  return((N / nrow(X_batch)) * grad_loglik + grad_logprior)
}
```

R code. Stochastic Gradient MCMC (SG-MCMC): Linear regression

```r
# SGLD algorithm
for (t in 1:iterations) {
  # Step size
  stepsize_t <- initial_stepsize / (1 + t / schedule)^decay_
    rate
  # Sample mini-batch
  idx <- sample(1:N, batch_size)
  X_batch <- X[idx, ]
  y_batch <- y[idx]
  # Beta update via SGLD
  grad <- grad_log_post(beta, X_batch, y_batch, N, sigma2)
  noise <- rnorm(K, 0, sqrt(stepsize_t))
  beta <- beta + 0.5 * stepsize_t * grad + noise
  # Sigma2 update via inverse-gamma (full conditional)
  resid <- y - X %*% beta
  shape <- (a0 + N) / 2
  rate <- (b0 + sum(resid^2)) / 2
  sigma2 <- 1 / rgamma(1, shape = shape, rate = rate)
  # Store draws
  trace_beta[t, ] <- beta
  trace_sigma2[t] <- sigma2
}
# Plot trace of beta[2]
plot(trace_beta[4001:5000, 2], type = "l", col = "blue", lwd
     = 2,
main = expression("SGLD: Evolution of " * beta[2]),
xlab = "Iteration", ylab = expression(beta[2]))
abline(h = beta_true[2], col = "red", lty = 2)
# Plot trace of sigma2
plot(trace_sigma2[4001:5000], type = "l", col = "darkgreen",
     lwd = 2,
main = expression("Trace plot of " * sigma^2),
xlab = "Iteration", ylab = expression(sigma^2))
abline(h = sigma2_true, col = "red", lty = 2)
```

13

Causal inference

Solutions of 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

$$\bar{y}_1 - \bar{y}_0.$$

Answer

Consider the simple linear regression model

$$Y_i = \beta_0 + \tau D_i + \mu_i, \qquad i = 1, \ldots, N,$$

where $D_i \in \{0, 1\}$ is a treatment indicator. In matrix notation:

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\mu}, \qquad X = \begin{pmatrix} 1 & D_1 \\ \vdots & \vdots \\ 1 & D_n \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \tau \end{pmatrix}.$$

Under the usual Gaussian likelihood and a non-informative prior $\pi(\boldsymbol{\beta}, \sigma^2) \propto \sigma^{-2}$, the posterior mean of $\boldsymbol{\beta}$ coincides with the maximum likelihood estimator (MLE), which is the ordinary least squares (OLS) estimator:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Compute $\mathbf{X}^\top \mathbf{X}$ and $\mathbf{X}^\top \mathbf{y}$:

$$\mathbf{X}^\top \mathbf{X} = \begin{pmatrix} N & N_1 \\ N_1 & N_1 \end{pmatrix}, \qquad \mathbf{X}^\top \mathbf{y} = \begin{pmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i:D_i=1} y_i \end{pmatrix},$$

where $N_1 = \sum_i D_i$ and $N_0 = N - N_1$. Let $\bar{y}_1 = \frac{1}{N_1} \sum_{i:D_i=1} y_i$ and $\bar{y}_0 = \frac{1}{N_0} \sum_{i:D_i=0} y_i$.

Compute the inverse:

$$(\mathbf{X}^\top \mathbf{X})^{-1} = \frac{1}{N_0 N_1} \begin{pmatrix} N_1 & -N_1 \\ -N_1 & N \end{pmatrix}.$$

Thus:

$$\widehat{\boldsymbol{\beta}} = \frac{1}{N_0 N_1} \begin{pmatrix} N_1 & -N_1 \\ -N_1 & N \end{pmatrix} \begin{pmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i:D_i=1} y_i \end{pmatrix} = \begin{pmatrix} \bar{y}_0 \\ \bar{y}_1 - \bar{y}_0 \end{pmatrix}.$$

Therefore, the intercept equals the mean outcome for the control group ($\bar{y}_0$), and the slope equals the difference in means between treated and control groups:

$$\widehat{\tau} = \bar{y}_1 - \bar{y}_0.$$

Since under a non-informative prior the posterior mean equals the ML estimator, the posterior mean of τ coincides with the sample difference in means:

$$\text{ATE} = \mathbb{E}[\tau \mid \mathbf{y}] = \bar{y}_1 - \bar{y}_0.$$

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 [8].

   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.

   Answer

   We want to maximize $H(f)$ subject to the constraints:

$$\int_{-\infty}^{\infty} f(y)\, dy = 1, \qquad \int_{-\infty}^{\infty} y f(y)\, dy = \mu, \qquad \int_{-\infty}^{\infty} (y-\mu)^2 f(y)\, dy = \sigma^2.$$

   Form the Lagrangian:

$$\mathcal{L}(f) = -\int f(y) \log f(y)\, dy + \lambda_0 \left(\int f(y)\, dy - 1 \right) + \lambda_1 \left(\int y f(y)\, dy - \mu \right) + \lambda_2 \left(\int (y-\mu)^2 f(y)\, dy - \sigma^2 \right).$$

Taking the functional derivative with respect to $f(y)$:

$$\frac{\delta \mathcal{L}}{\delta f} = -(\log f(y) + 1) + \lambda_0 + \lambda_1 y + \lambda_2 (y - \mu)^2 = 0.$$

Thus:

$$\log f(y) = \lambda_0' + \lambda_1 y + \lambda_2 (y - \mu)^2,$$

where λ_0' absorbs constants. Exponentiating:

$$f(y) \propto \exp\left(\lambda_1 y + \lambda_2 (y - \mu)^2\right).$$

For integrability, $\lambda_2 < 0$. Completing the square:

$$f(y) \propto \exp\left(-\frac{(y - \mu)^2}{2\sigma^2}\right),$$

which is the kernel of a normal density. Normalizing:

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mu)^2}{2\sigma^2}\right).$$

Therefore, the maximum entropy distribution for a given mean μ and variance σ^2 is

$$Y \sim N(\mu, \sigma^2).$$

3. Use the package *dagity* to construct the DAG in Figure 13.5, verify that it is acyclic, and check whether the causal effect of D on Y is identifiable by controlling for $\mathbf{X}$.

   Answer

R code. Simple exercise in dagity package: Conditional independence assumption

```
library(dagitty)
library(ggdag)

Gd = dagitty('dag{
  X [pos="-1,1"]
  D [exposure, pos="0,0"]
  Y [outcome, pos="1,1"]
  X -> D
  X -> Y
  D -> Y
}')

ggdag(Gd) +  theme_dag()

isAcyclic(Gd)

for( n in names(Gd) ){
  for( m in children(Gd,n) ){
    a <- adjustmentSets(Gd, n, m, effect = c("direct"), type
     = c("minimal"))
    if( length(a) > 0 ){
      cat("The effect ",n,"->",m,
      " is identifiable by controlling for:\n",sep="")
      print( a, prefix=" * " )
    }
  }
}
```

4. Use the package *dagitty* to construct the DAG in Figure 13.8, verify that it is acyclic, and check that the causal effect of D on Y is identifiable by controlling for $\mathbf{X}$ but not for C.

   Answer

R code. Simple exercise in dagity package: Collider bias

```
#### DAG: Collider bias  ####
library(dagitty)
library(ggdag)

Gd = dagitty('dag{
  X [pos="-1,1"]
  D [exposure, pos="0,0"]
  Y [outcome, pos="1,1"]
  C [pos="0,0.5"]
  X -> D
  X -> Y
  X -> C
  D -> Y
  D -> C
}')

ggdag(Gd) +  theme_dag()

isAcyclic(Gd)

adjustmentSets(Gd, exposure = "D", outcome = "Y")

for( n in names(Gd) ){
  for( m in children(Gd,n) ){
    a <- adjustmentSets(Gd, n, m, effect = c("direct"), type
     = c("minimal"))
    if( length(a) > 0 ){
      cat("The effect ",n,"->",m,
      " is identifiable by controlling for:\n",sep="")
      print( a, prefix=" * " )
    }
  }
}
```

5. Use the package *dagitty* to construct the DAG in Figure 13.11, taking into account that **U** is unobserved (latent). Verify that it is acyclic, check that Z is a valid instrument, and determine whether the causal effect of D on Y is identifiable.

   Answer

R code. Simple exercise in dagity package: Instrumental variable

```
#### DAG: Instruments  ####
library(dagitty)
library(ggdag)

Gd = dagitty('dag{
  U [latent, pos="-1,1"]
  D [exposure, pos="0,0"]
  Y [outcome, pos="1,1"]
  Z [pos="0,0.5"]
  U -> D
  U -> Y
  D -> Y
  Z -> D
}')

ggdag(Gd) +  theme_dag()

isAcyclic(Gd)

instrumentalVariables(Gd)

adjustmentSets(Gd, exposure = "D", outcome = "Y")
```

6. **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.

   Answer

   Taking into account that

$$\begin{aligned}\tau_{LATE} &= \mathbb{E}[Y_i(1) - Y_i(0) \mid D(1) = 1, D(0) = 0] \\ &= \frac{\mathbb{E}[Y_i \mid Z_i = 1] - \mathbb{E}[Y_i \mid Z_i = 0]}{\mathbb{E}[D_i \mid Z_i = 1] - \mathbb{E}[D_i \mid Z_i = 0]},\end{aligned}$$

   we can recover the ITT effect by multiplying the LATE by the effect of eligibility on participation. The following code illustrates this procedure and plots the ITT. The posterior distribution of the ITT is shown in Figure 13.1, which closely resembles the posterior distribution of the effect of eligibility reported in the main text.

R code. Treatment effect: 401(k) participation on net financial assets

```r
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

```
ITT <- Bs[,1]*Gs[,1]
summary(coda::mcmc(ITT))
# Convert to data frame for ggplot
df_ITT <- data.frame(ITT = as.vector(ITT))

# Plot posterior distribution of treatment effect
ggplot(df_ITT, aes(x = ITT)) + geom_density(fill = "
    steelblue", alpha = 0.6) +
geom_vline(xintercept = mean(ITT), color = "red", linetype =
     "dashed", linewidth = 1) + labs(title = "Posterior
    Distribution of 401(k) ITT", x = expression(beta["ITT"])
    , y = "Density") + theme_minimal(base_size = 14)
```

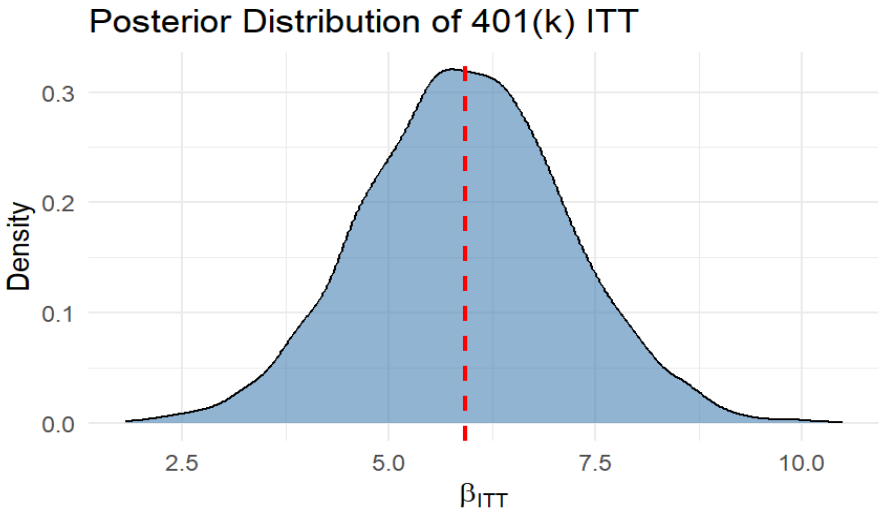

FIGURE 13.1
Posterior distribution: Intention-to-treat effect 401k participation on net financial assets.

7. **401(k) participation on net financial assets continues II**

   Perform inference in this example assuming that the stochastic errors follow a Dirichlet process, using the function *rivDP* from the *bayesm* package to

analyze 401(k) participation and its effect on net financial assets under the same specification as in the main text, and plot the LATE.

Answer

The following code implements the exercise, and Figure 13.2 display the posterior distribution of the LATE. The results are very similar to the ones in the main text assuming a normal distribution; however, the degree of dispersion of higher using a Dirichlet process for the stochastic errors.

R code. Treatment effect: 401(k) participation on net financial assets using Dirichlet process

```r
rm(list = ls()); set.seed(10101)
library(ggplot2); library(bayesm)
# Load data
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(age, inc, fsize, educ, marr, twoearn,
    db, pira, hown))  # Exogenous regressors with intercept
z <- as.matrix(e401)  # Instrument: eligibility (NO
    intercept here)

# specify data input and mcmc parameters
Data = list();
Data$z <- z
Data$x <- x
Data$y <- y
Data$w <- w

Mcmc = list()
Mcmc$maxuniq <- 100
Mcmc$R <- 5000
keep <- seq((1000+1), Mcmc$R)

out <- rivDP(Data=Data, Mcmc=Mcmc)

# Plot posterior distribution of treatment effect
df_ITT <- data.frame(ITT = out[["betadraw"]][kepp])
ggplot(df_ITT, aes(x = ITT)) + geom_density(fill = "
    steelblue", alpha = 0.6) + geom_vline(xintercept = mean(
    ITT), color = "red", linetype = "dashed", linewidth = 1)
     + labs(title = "Posterior Distribution of 401(k) ITT:
    Dirichlet process", x = expression(beta["ITT"]), y = "
    Density") + theme_minimal(base_size = 14)
```

8. **Difference-in-Differences simulation continues**

   Perform the simulation of the DiD example, and perform inference using the specification:

$$Y_{it} = \alpha + \alpha_i + \phi_t + \tau_2 \left[D_i \cdot \mathbb{1}(t = 2) \right] + \epsilon_{it}.$$

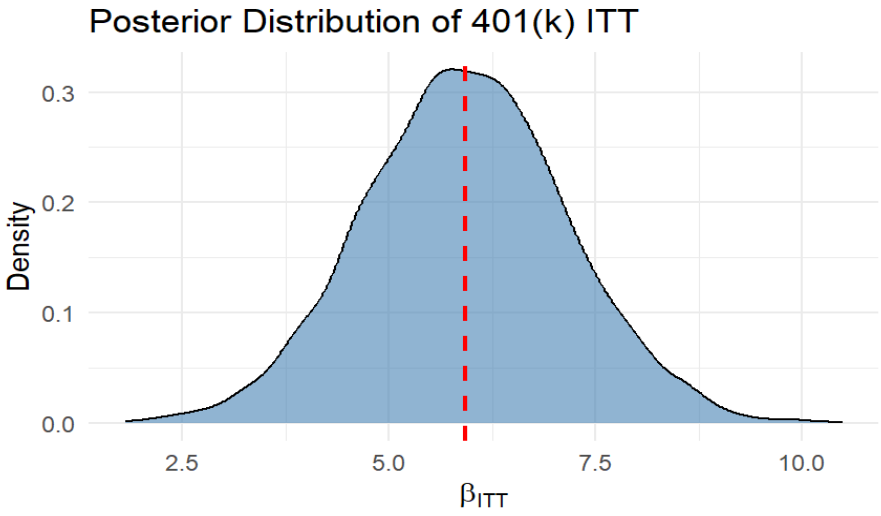

FIGURE 13.2
Posterior distribution: Local average treatment effect 401k participation on net financial assets using a Dirichlet process.

Answer

You can check that we get same results as in example in the main book.

R code. Difference-in-Differences: Simulation exercise

```r
rm(list = ls()); set.seed(10101)
library(ggplot2); library(dplyr); library(fastDummies)
# Parameters
N_per_group <- 200          # units per group
T_periods    <- 2           # keep 2x2 for clarity
tau_true     <- 1         # ATT
sigma_eps    <- 0.5         # noise SD
# Panel index
id  <- rep(1:(2*N_per_group), each = T_periods)
t   <- rep(1:T_periods, times = 2*N_per_group)
# Group: treated (D=1) vs control (D=0)
D   <- rep(c(rep(0, N_per_group), rep(1, N_per_group)), each
    = T_periods)
# Post indicator (t=2 is post)
post <- as.integer(t == 2)
# Unit fixed effects (random heterogeneity)
alpha_i <- rnorm(2*N_per_group, 0, 0.8)
alpha   <- alpha_i[id]
# Time effects
phi_t <- c(0, -1.8)  # common decline from t=1 to t=2
phi   <- phi_t[t]
treat_effect <- tau_true * (D * post)
# Outcome
eps <- rnorm(length(id), 0, sigma_eps)
Y   <- alpha + phi + treat_effect + eps
did <- data.frame(id, t, D, post, Y)
plot_data <- did %>%
group_by(D, t) %>%
summarise(meanY = mean(Y), .groups = "drop") %>%
mutate(Group = ifelse(D == 1, "Treated", "Control"))
ggp <- ggplot(plot_data, aes(x = t, y = meanY, group = Group
    , linetype = Group)) +
geom_line(linewidth = 1) + geom_point() + scale_x_continuous
    (breaks = c(1, 2), labels = c("t = 1 (pre)", "t = 2 (
    post)")) + labs(x = "Time", y = "Mean outcome", title =
    "Synthetic DiD: Parallel Trends & No Anticipation") +
    theme_minimal(base_size = 12)
print(ggp)
# Bayesian inference: Model with interaction treatment x
    post period
post_fit1 <- MCMCpack::MCMCregress(Y ~ 1 + factor(id) +
    factor(t) + I(D * post), data = did, burnin = 100, mcmc
    = 1000)
tau_draws1 <- post_fit1[, "I(D * post)"]
quantile(tau_draws1, c(.025,.5,.975))
```

9. **Difference-in-Differences simulation continues II**

Note that another strategy to perform inference on the ATT is to estimate the saturated model

$$Y_{it} = \sum_{t,d} \mu_{tl} \big[D_{il} \cdot \mathbb{1}(t = t) \big] + \epsilon_{it}, \quad t = 1, 2, \ d = 1, 0,$$

and then use the posterior draws to compute

$$\tau_2 = (\mu_{21} - \mu_{11}) - (\mu_{20} - \mu_{10}).$$

Explain why inference on τ_2 using this approach in the simulation setting shows that the posterior mean is similar to that from the previous approaches, but the level of uncertainty is higher.

Answer

There is more uncertainty because performing inference on τ_2 uses four parameters in this setting, whereas in the previous we use only one parameter.

R code. Difference-in-Differences: Simulation exercise

```
rm(list = ls()); set.seed(10101)
library(ggplot2); library(dplyr); library(fastDummies)
# Parameters
N_per_group <- 200          # units per group
T_periods    <- 2           # keep 2x2 for clarity
tau_true     <- 1           # ATT
sigma_eps    <- 0.5         # noise SD
# Panel index
id  <- rep(1:(2*N_per_group), each = T_periods)
t   <- rep(1:T_periods, times = 2*N_per_group)
# Group: treated (D=1) vs control (D=0)
D   <- rep(c(rep(0, N_per_group), rep(1, N_per_group)), each
    = T_periods)
# Post indicator (t=2 is post)
post <- as.integer(t == 2)
# Unit fixed effects (random heterogeneity)
alpha_i <- rnorm(2*N_per_group, 0, 0.8)
alpha   <- alpha_i[id]
# Time effects
phi_t <- c(0, -1.8)  # common decline from t=1 to t=2
phi   <- phi_t[t]
# No anticipation: effect is zero in pre, equals tau in post
    *only for treated*
treat_effect <- tau_true * (D * post)
# Outcome: Y_it(0) = alpha_i + phi_t + linear trend if you
    want; here just FE + shock
eps <- rnorm(length(id), 0, sigma_eps)
Y   <- alpha + phi + treat_effect + eps
did <- data.frame(id, t, D, post, Y)
# Calculating the means at each level, and then ATT
# ---- Build 2x2 cell indicators (no intercept model) ----
did$c_pre  <- as.integer(did$D == 0 & did$t == 1)
did$c_post <- as.integer(did$D == 0 & did$t == 2)
did$t_pre  <- as.integer(did$D == 1 & did$t == 1)
did$t_post <- as.integer(did$D == 1 & did$t == 2)
# Sanity check: each row must belong to exactly one cell
stopifnot(all(did$c_pre + did$c_post + did$t_pre + did$t_
    post == 1))
# ---- Bayesian saturated cell-mean model (compatible priors
    ) ----
# No intercept: each coefficient IS a cell mean.
fit_cells <- MCMCpack::MCMCregress(
Y ~ 0 + c_pre + c_post + t_pre + t_post, data  = did, burnin
    = 1000, mcmc = 10000, thin = 5)
draws_cells <- as.matrix(fit_cells)
colnames(draws_cells)  # should be c("c_pre","c_post","t_pre
    ","t_post", "sigma2")
# Posterior of the four cell means
mu_c_pre_draw  <- draws_cells[, "c_pre"]; mu_c_post_draw <-
    draws_cells[, "c_post"]
mu_t_pre_draw  <- draws_cells[, "t_pre"]; mu_t_post_draw <-
    draws_cells[, "t_post"]
# ATT = (mu_t,post - mu_t,pre) - (mu_c,post - mu_c,pre)
ATT_cells_draw <- (mu_t_post_draw - mu_t_pre_draw) - (mu_c_
    post_draw - mu_c_pre_draw)
quantile(ATT_cells_draw, c(.025, .5, .975))
```

10. Perform a simulation exercise to assess the ability of BETEL to identify the causal effect when an instrument is used to address the omission of relevant regressors. Illustrate the consequences of varying the dependence between the omitted regressor and the observed regressor.

Answer

We simulate the observed process $X_i = 0.5Z_i + e_i$, with the unobserved regressor defined as $W_i = H_i + \nu_i$, where

$$\begin{bmatrix} e_i \\ \nu_i \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right).$$

Thus, ρ defines the degree of correlation between X_i and W_i, and we perform two exercises with $\rho \in \{0.0, 0.7\}$. The variables Z_i and H_i are standard normal. The dependent variable is specified as

$$Y_i = 1 + 1.2X_i - 0.7W_i + \mu_i,$$

where μ_i follows a mixture of two normal distributions with means 0.5 and -0.5, and standard deviations 0.5 and 1.2. The sample size is 2,000, the burn-in is 1,000, and the number of MCMC draws retained after burn-in is 10,000. We adopt the default hyperparameter values provided in the *betel* package. For comparison, we also perform the analysis under the assumption of no endogeneity using the package *MCMCpack* using default values for the hyperparameters.

The following code shows the implementation, and Figure 13.3 displays the posterior distributions. When $\rho = 0$, implying no association between the regressors, there are no endogeneity issues, and assuming exogeneity yields more efficient posterior estimates. In contrast, when $\rho = 0.7$, endogeneity arises, and assuming exogeneity produces a biased posterior distribution. BETEL performs relatively well in both settings, although it is less efficient.

R code. BETEL: Omission of a relevant regressor

```
rm(list = ls()); set.seed(10101)
library(betel); library(ucminf)
# Simulate data
N <- 2000; d <- 2; k <- 2
gamma <- 0.5; beta <- c(1, 1.2, -0.7); rho <- 0.7
# Mixture
mum1 <- 1/2; mum2 <- -1/2
mu1 <- rnorm(N, mum1, 0.5); mu2 <- rnorm(N, mum2, 1.2)
mu <- sapply(1:N, function(i){sample(c(mu1[i], mu2[i]), 1,
    prob = c(0.5, 0.5))})
z <- rnorm(N) # Instrument
E <- MASS::mvrnorm(n = N, mu = c(0, 0), Sigma = matrix(c(1,
    rho,rho,1),2,2))
x <- gamma*z + E[,1] # Observed regressor
w <- rnorm(N) + E[,2] # Unobserved correlated regressor
X <- cbind(1, x, w)
y <- X%*%beta + mu
dat <- cbind(1, x, z) # Data
# Function g_i by row in BETEL
gfunc <- function(psi = psi, y = y, dat = dat) {
  X <- dat[,1:2]
  e <- y - X %*% psi
  E <- e %*% rep(1,d)
  Z <- dat[,c(1,3)]
  G <- E * Z;
  return(G)
}
nt <- round(N * 0.1, 0); # training sample size for prior
psi0 <- lm(y[1:nt]~x[1:nt])$coefficients # Starting value of
     psi = (theta, v), v is the slack parameter in CSS
    (2018)
names(psi0) <- c("alpha","beta")
psi0_ <- as.matrix(psi0) # Prior mean of psi
Psi0_ <- 5*rep(1,k) # Prior dispersions of psi
lam0 <- .5*rnorm(d) # Starting value of lambda
nu <- 2.5 # df of the prior student-t
nuprop <- 15 # df of the student-t proposal
n0 <- 1000 # burn-in
m <- 10000 # iterations beyond burn-in
# MCMC ESTIMATION BY THE CSS (2018) method
psim2 <- betel::bayesetel(gfunc = gfunc, y = y[-(1:nt)], dat
     = dat[-(1:nt),], psi0 = psi0, lam0 = lam0, psi0_ = psi0
    _, Psi0_ = Psi0_, nu = nu, nuprop = nuprop,
controlpsi = list(maxiterpsi = 50, mingrpsi = 1.0e-8), #
    list of parameters in maximizing likelihood over psi
controllam = list(maxiterlam = 50, # list of parameters in
    minimizing dual over lambda
mingrlam = 1.0e-7),
n0 = n0, m = m)
MCMCreg2 <- MCMCpack::MCMCregress(y~x)
```

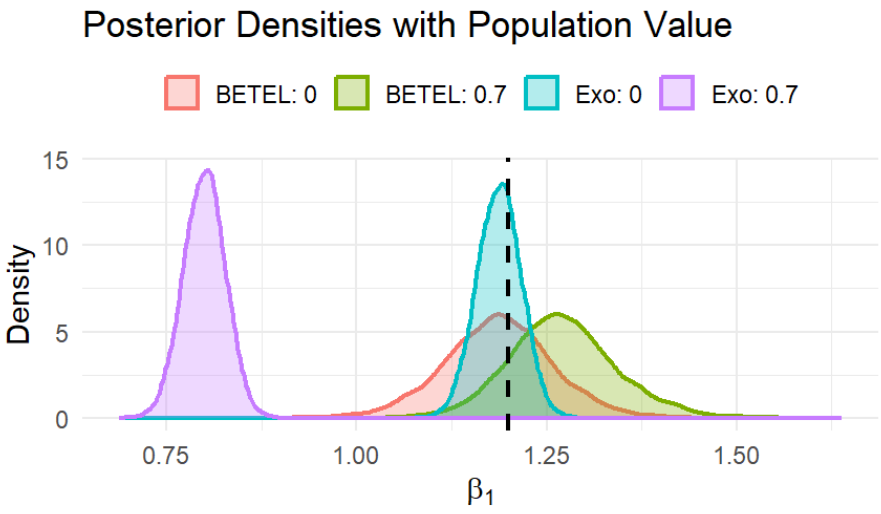

FIGURE 13.3
Posterior distribution slope parameter given omission of relevant regressor: betel and MCMCregress

11. **Simultaneous causality continues**

    Use the demand–supply simulation and the moment conditions to infer the causal effect of a 10% tax, implementing a BETEL algorithm from scratch.

    Answer

    The following code shows how to program a BETEL from scratch for the demand-supply simulation exercise. Figure 13.4 displays the posterior distribution of the causal effect. The 95% credible interval contains the population value, and the posterior mean lies close to it.

R code. BETEL: Simultaneous causality

```r
# Simulation
rm(list = ls()); set.seed(12345)
# Population parameters demand
B1 <- 5; B2 <- -0.5; B3 <- 0.8; B4 <- -0.4; B5 <- 0.7; SD <- 0.5
# Population parameters supply
A1 <- -2; A2 <- 0.5; A3 <- -0.4; SS <- 0.5
# Reduced form parameters
P0 <- (A1-B1)/(B2-A2); P2 <- -B3/(B2-A2); P3 <- -B4/(B2-A2); P1 <- A3/(B2-A2); P4 <- -B5/(B2-A2)
T0 <- B1+B2*P0; T2 <- B3+B2*P2; T3 <- B4+B2*P3; T1 <- B2*P1; T4 <- B5+B2*P4;
n <- 5000
ED <- rnorm(n, 0, SD); ES <- rnorm(n, 0, SS)
VP <- (ES-ED)/(B2-A2); UQ <- B2*VP+ED
y <- rnorm(n, 10, 1); pc <- rnorm(n, 5, 1); er <- rnorm(n, 15, 1); ps <- rnorm(n, 5, 1);
p <- P0+P1*er+P2*y+P3*pc+P4*ps+VP
q <- T0+T1*er+T2*y+T3*pc+T4*ps+UQ

### Supply ###
p_s  <- as.numeric(scale(p,  TRUE, TRUE))
er_s <- as.numeric(scale(er, TRUE, TRUE))

X <- cbind(1, p_s, er_s)  # k = 3  (STRUCTURAL: intercept, p, er)
Z <- cbind(1,
as.numeric(scale(y,  TRUE, TRUE)),
as.numeric(scale(pc, TRUE, TRUE)),
as.numeric(scale(ps, TRUE, TRUE)),
er_s)          # EXCLUDED instruments y,pc,ps, plus er

k <- ncol(X); d <- ncol(Z)

# Objective function
lambdafunc <- function(lambda, theta, q, X, Z){
  e <- as.numeric(q - X %*% theta); if (!all(is.finite(e))) return(1e300)
  G <- Z * e
  eta <- as.numeric(G %*% lambda); if (!all(is.finite(eta))) return(1e300)
  m <- max(eta); m + log(mean(exp(eta - m)))
}

# ----- Profile ETEL log-likelihood (use this in MH) -----
loglik_etel <- function(theta, q, X, Z){
  e <- as.numeric(q - X %*% theta); if (!all(is.finite(e))) return(-Inf)
  G <- Z * e
  op <- optim(par = rep(0, ncol(G)), fn = function(l){
    eta <- as.numeric(G %*% l); m <- max(eta)
    m + log(mean(exp(eta - m)))
  }, method="BFGS", control=list(maxit=2000))
  if (!is.finite(op$value)) return(-Inf)
  lam <- op$par
  eta <- as.numeric(G %*% lam)
  sum(eta) - length(e) * log(sum(exp(eta)))
}
```

R code. BETEL: Simultaneous causality

```
theta0 <- coef(lm(q ~ p_s + er_s))
# ----- MH using profile ETEL -----
b0 <- rep(0, k); B0 <- 1000*diag(k)
S <- 10000; burnin <- 2000; thin <- 5; tot <- S + burnin
BETA <- matrix(NA, tot, k); accept <- logical(tot)
step <- c(0.05, 0.02, 0.02)                    #
    preconditioned steps
BETA[1,] <- theta0
LL <- loglik_etel(BETA[1,], q, X, Z)
pb <- txtProgressBar(min=0, max=tot, style=3)
for(s in 2:tot){
  cand <- BETA[s-1,] + rnorm(k, 0, step)
  LLc  <- loglik_etel(cand, q, X, Z)
  priorRat <- mvtnorm::dmvnorm(cand, b0, B0, log=TRUE) -
  mvtnorm::dmvnorm(BETA[s-1,], b0, B0, log=TRUE)
  loga <- (LLc - LL) + priorRat
  if (is.finite(loga) && log(runif(1)) <= loga) {
    BETA[s,] <- cand; LL <- LLc; accept[s] <- TRUE
  } else {
    BETA[s,] <- BETA[s-1,]; accept[s] <- FALSE
  }
  if (s %% 200 == 0) {          # gentle adaptation
    acc <- mean(accept[(s-199):s])
    if (acc > 0.4) step <- step * 1.25
    if (acc < 0.15) step <- step / 1.25
  }
  setTxtProgressBar(pb, s)
}
close(pb)
cat("\nAcceptance rate:", mean(accept), "\n")
keep <- seq(burnin, tot, by = thin)
post_s <- BETA[keep, , drop=FALSE]   # posterior draws in
    scaled space
## ---------- Back-transform each draw to original scale
    ----------
p_mu <- mean(p);   p_sd <- sd(p); er_mu <- mean(er); er_sd
    <- sd(er)
alpha_draws <- cbind(
alpha0  = post_s[,1] - post_s[,2] * (p_mu/p_sd) - post_s[,3]
     * (er_mu/er_sd),
alphap  = post_s[,2] / p_sd,
alphaer = post_s[,3] / er_sd
)
## ---------- Summaries ----------
summ <- function(x) c(mean=mean(x), sd=sd(x), quantile(x, c
    (.025,.25,.5,.75,.975)))
cat("\nPosterior (original scale):\n")
print(rbind(
alpha0  = summ(alpha_draws[,"alpha0"]), alphap  = summ(alpha
    _draws[,"alphap"]), alphaer = summ(alpha_draws[,"alphaer
    "])))
cat("\nTrue (original scale):\n")
print(c(alpha0=A1, alphap=A2, alphaer=A3))
ElastSupPrice <- alpha_draws[,2]
#### Demand ####
y_s <- as.numeric(scale(y)); pc_s <- as.numeric(scale(pc))
ps_s <- as.numeric(scale(ps)); Xdem <- cbind(1, p_s, y_s, pc
    _s, ps_s)
k <- ncol(Xdem); d <- ncol(Z); theta0 <- coef(lm(q ~ p_s + y
    _s + pc_s + ps_s))
```

R code. BETEL: Simultaneous causality

```r
# ----- MH using profile ETEL -----
b0 <- rep(0, k); B0 <- 1000*diag(k)
S <- 10000; burnin <- 2000; thin <- 5; tot <- S + burnin
BETAdem <- matrix(NA, tot, k); accept <- logical(tot)
step <- c(0.05, 0.02, 0.02); BETAdem[1,] <- theta0
LL <- loglik_etel(BETAdem[1,], q, X = Xdem, Z)
pb <- txtProgressBar(min=0, max=tot, style=3)
for(s in 2:tot){
  cand <- BETAdem[s-1,] + rnorm(k, 0, step)
  LLc  <- loglik_etel(cand, q, X = Xdem, Z)
  priorRat <- mvtnorm::dmvnorm(cand, b0, B0, log=TRUE) -
  mvtnorm::dmvnorm(BETAdem[s-1,], b0, B0, log=TRUE)
  loga <- (LLc - LL) + priorRat
  if (is.finite(loga) && log(runif(1)) <= loga) {
    BETAdem[s,] <- cand; LL <- LLc; accept[s] <- TRUE
  } else {
    BETAdem[s,] <- BETAdem[s-1,]; accept[s] <- FALSE
  }
  if (s %% 200 == 0) {   # gentle adaptation
    acc <- mean(accept[(s-199):s])
    if (acc > 0.4) step <- step * 1.25
    if (acc < 0.15) step <- step / 1.25
  }
  setTxtProgressBar(pb, s)
}
close(pb)
cat("\nAcceptance rate:", mean(accept), "\n")
keep <- seq(burnin, tot, by = thin)
postDem_s <- BETAdem[keep, , drop=FALSE]
p_mu <- mean(p);   p_sd <- sd(p); y_mu <- mean(y); y_sd <-
    sd(y)
pc_mu <- mean(pc); pc_sd <- sd(pc); ps_mu <- mean(ps); ps_sd
     <- sd(ps)
alphaDem_draws <- cbind(
alpha0  = postDem_s[,1] - postDem_s[,2] * (p_mu/p_sd) -
    postDem_s[,3] * (y_mu/y_sd)
- postDem_s[,4] * (pc_mu/pc_sd) - postDem_s[,5] * (pc_mu/pc_
    sd),
alphap  = postDem_s[,2] / p_sd, alphay = postDem_s[,3] / y_
    sd, alphapc = postDem_s[,4] / pc_sd, alphaps = postDem_s
    [,5] / ps_sd)
summ <- function(x) c(mean=mean(x), sd=sd(x), quantile(x, c
    (.025,.25,.5,.75,.975)))
cat("\nPosterior (original scale):\n")
print(rbind(
alpha0  = summ(alphaDem_draws[,"alpha0"]), alphap  = summ(
    alphaDem_draws[,"alphap"]), alphay = summ(alphaDem_draws
    [,"alphay"]),
alphapc = summ(alphaDem_draws[,"alphapc"]), alphaps = summ(
    alphaDem_draws[,"alphaps"])))
cat("\nTrue (original scale):\n")
print(c(alpha0=B1, alphap=B2, alphay=B3, alphpc=B4, alphaps=
    B5))
ElastDemPrice <- alphaDem_draws[,2]; tax <- 0.1
CausalEffect <- (ElastSupPrice*ElastDemPrice)*log(1+tax)/(
    ElastSupPrice-ElastDemPrice)
popCauEff <- (A2 * B2)/(A2 - B2) * log(1 + tax)
```

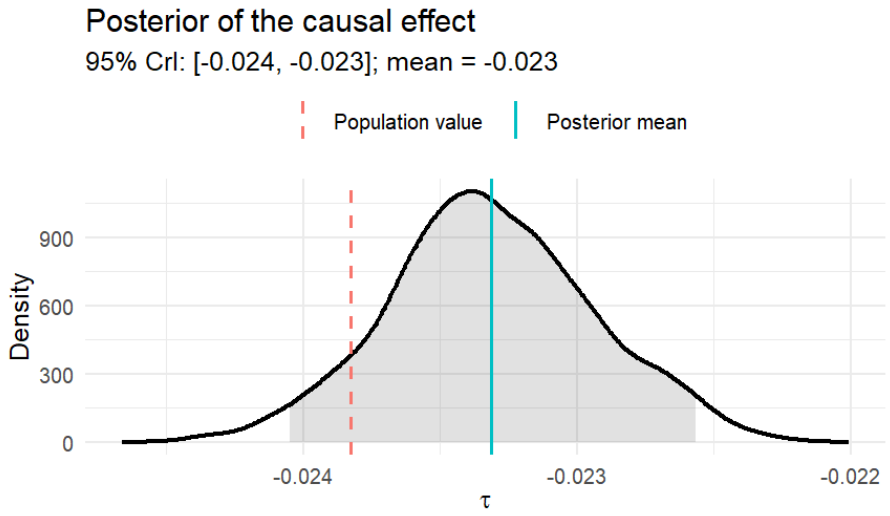

FIGURE 13.4
Posterior distribution causal effect: BETEL

12. Perform a simulation of the instrumental variable quantile regression, and then perform inference using the general Bayes posterior framework. Use the Laplace asymmetric distribution to simulate the stochastic error, such that

$$\int_{-\infty}^{0} f_\tau(\mu_i)\, d\mu_i = \tau,$$

which implies that the τ-th quantile of μ_i is 0. In addition, let

$$q_\tau(Y_i \mid X_i, D_i) = \beta_0 + \beta_1 X_i + \beta_2 D_i$$

denote the τ-th quantile regression function of Y_i given X_i and D_i, with $0 < \tau < 1$. Specifically, consider the model

$$Y_i = \beta_{0\tau} + \beta_{1\tau} X_i + \beta_{2\tau} D_i + \mu_i,$$

together with

$$D_i = \gamma Z_i + \delta \mu_i, \quad Z_i \sim N(0,1).$$

Answer:

The following code shows how to perform inference in this simulation. The 95% credible intervals encompass the population values, and the posterior means are also close to these values.

R code. Instrumental Variable Quantile Regression: General Bayes posteriors

```r
rm(list = ls()); set.seed(10101); library(quantreg)
### Simulation ####
# Asymmetric Laplace
rALD <- function(n, tau) {
  stopifnot(tau > 0, tau < 1)
  theta <- (1 - 2*tau) / (tau * (1 - tau))
  psi   <- 2 / (tau * (1 - tau))
  v  <- rexp(n, rate = 1)
  z  <- rnorm(n)
  mu <- theta * v + sqrt(psi * v) * z
  return(mu)
}
## --- Simulator for a given tau ---
simulate_qr <- function(n = 5000, tau = 0.5,
beta0 = 0.7,
beta1_tau = 1.5,
beta2_tau = 1,
gamma = 1,
delta = 0.5) {
  # Nonnegative regressor stabilizes intercept & preserves
    tau-ordering of slopes
  x <- runif(n, 0, 1)
  # optional: anchor a portion at 0 to pin intercept
  x[sample.int(n, size = round(0.1*n))] <- 0
  mu <- rALD(n, tau = tau) # ALD error
  z <- rnorm(n) # Instrument
  d <- gamma * z + delta * mu
  y <- beta0 + beta1_tau * x + beta2_tau * d + mu
  df <- data.frame(y, x, d, z)
}
tau <- 0.75
df <- simulate_qr(tau = tau, beta0 = 1, beta1_tau = 1,
beta2_tau = 1, gamma = 1, delta = 1, n = 5000)
Reg <- MCMCpack::MCMCquantreg(y~x+d, data = df, tau = tau)
summary(Reg)
summary(rq(y ~ x + d, tau = tau, data = df))
LossFunct <- function(par, tau, y, z, x, d){
  n <- length(y)
  X <- cbind(1, x, d)
  Z <- cbind(1, x, z)
  Ind <- as.numeric(y <= X%*%par)
  gn <- colMeans((tau - Ind) * Z)
  Wni <- lapply(1:n, function(i) {Z[i,] %*% t(Z[i,])})
  Wn <- 1 / (tau * (1 - tau)) * solve(Reduce("+", Wni)/n)
  Ln <- - 0.5 * n * t(gn) %*% Wn %*% gn
  return(Ln)
}
par0 <- rnorm(3)
LossFunct(par = par0, tau = tau, y = df$y, z = df$z, x = df$
    x, d = df$d)
```

R code. Instrumental Variable Quantile Regression: General Bayes posteriors

```
# ----- MH using Ln -----
k <- 3
b0 <- rep(0, k); B0 <- 1000*diag(k)
S <- 5000; burnin <- 6000; thin <- 5; tot <- S + burnin
BETA <- matrix(NA, tot, k); accept <- logical(tot)
step <- 0.05
BETA[1,] <- par0
LL <- LossFunct(par = BETA[1,], tau = tau, y = df$y, z = df$
    z, x = df$x, d = df$d)
pb <- txtProgressBar(min=0, max=tot, style=3)
for(s in 2:tot){
  cand <- BETA[s-1,] + rnorm(k, 0, step)
  LLc  <- LossFunct(par = cand, tau = tau, y = df$y, z = df$
    z, x = df$x, d = df$d)
  priorRat <- mvtnorm::dmvnorm(cand, b0, B0, log=TRUE) -
  mvtnorm::dmvnorm(BETA[s-1,], b0, B0, log=TRUE)
  loga <- (LLc - LL) + priorRat
  if (is.finite(loga) && log(runif(1)) <= loga) {
    BETA[s,] <- cand; LL <- LLc; accept[s] <- TRUE
  } else {
    BETA[s,] <- BETA[s-1,]; accept[s] <- FALSE
  }
  if (s <= burnin && s %% 200 == 0) {
    acc <- mean(accept[(s-199):s])
    if (acc > 0.4) step <- step * 1.25
    if (acc < 0.15) step <- step / 1.25
  }
  setTxtProgressBar(pb, s)
}
close(pb)
cat("\nAcceptance rate:", mean(accept), "\n")
keep <- seq(burnin, tot, by = thin)
post <- BETA[keep, , drop=FALSE]   # posterior draws in
    scaled space
colnames(post) <- c("beta0","beta_x","beta_d")
# Posterior summaries
post_mean <- colMeans(post)
post_ci   <- apply(post, 2, quantile, c(0.025, 0.975))
round(post_mean, 3); round(post_ci, 3)
```

13. **Example: Doubly robust Bayesian inference continues**

    Perform a simulation exercise to assess the consequences of misspecifying the propensity score function and the outcome regression in doubly robust Bayesian inference.

Answer:

Let us simulate the process

$$\eta_i^{\pi} = -0.3 + 0.8X_{i1} - 0.5X_{i2} + 0.7X_{i3} + 0.7X_{i1}X_{i3} + 0.5sin(X_{i1}),$$

$$\eta_i^{m} = -0.2 + 0.6D_i + 0.5X_{i1} - 0.4X_{i2} + 0.3X_{i3} + 0.6X_{i1}^2 - 0.5X_{i1}X_{i2},$$

where $X_1 \sim N(0,1)$, $X_2 \sim$ Bernoulli(0.5), and $X_3 \sim U(-1,1)$, and the sample size is 2,000.

The following code implements the algorithm.

R code. Doubly robust Bayesian inference: Misspecified simulation

```r
rm(list = ls()); set.seed(123); library(glmnet)

## simulate covariates
n  <- 2000
X1 <- rnorm(n)                  # continuous
X2 <- rbinom(n, 1, 0.5)         # binary
X3 <- runif(n, -1, 1)           # continuous
Z <- cbind(X1, X2, X3)
# True propensity and outcome
logit <- function(t) 1/(1+exp(-t))
true_pi  <- function(X){ logit(-0.3 + 0.8*X[,1] -0.5*X[,2] + 0.7*X[,3] + 0.7*X[,1]*X[,3] + 0.5*sin(X[,1])) }
true_eta <- function(d,X){ -0.2 + 0.6*d + 0.5*X[,1] -0.4*X[,2] + 0.3*X[,3] + 0.6*(X[,1]^2) - 0.5*X[,1]*X[,2] }
true_p   <- function(d,X){ logit(true_eta(d,X)) }
ATE  <- mean(true_p(1,Z) - true_p(0,Z))
pi <- true_pi(Z)
D  <- rbinom(n,1,pi)
P  <- true_p(D,Z)
Y  <- rbinom(n,1,P)
dat <- data.frame(Y, D, X1, X2 = factor(X2), X3)
######### Doubly Robust Bayesian: Implementation #########
# OK specification propensity score
Z <- cbind(X1, X2, X3, X1*X3, sin(X1)); p <- dim(Z)[2]
covars <- c("X1","X2","X3", "X1X3", "sinX1")
# # OK specification outcome regression
# Z <- cbind(X1, X2, X3, X1*X2, X1^2); p <- dim(Z)[2]
# covars <- c("X1","X2","X3", "X1X2", "X12")
cvfit <- cv.glmnet(Z, D, family = "binomial", alpha=1, nfolds = 10) # lasso,min
ps.coef <- coef(cvfit, s = "lambda.min")[0:p+1]
ps_hat <- predict(cvfit, newx = Z, s = "lambda.min",type = "response")
ps_hat <- pmin(pmax(ps_hat, 1e-6), 1 - 1e-6)
# Riesz representer
riesz_fun <- function(d, ps) d/ps - (1 - d)/(1 - ps)
# GP model
make_gp <- function(X_train, y_bin, use_correction = FALSE, gamma_scaled = NULL,
approx_method = gplite::approx_laplace(),
init_lscale = 0.3) {
  stopifnot(length(y_bin) == nrow(X_train))
  cf_se <- gplite::cf_sexp(vars = colnames(X_train), lscale = init_lscale, magn = 1, normalize = TRUE)
  cfs <- list(cf_se)
  if (use_correction) {
    stopifnot(!is.null(gamma_scaled))
    cf_lin <- gplite::cf_lin(vars = "gamma", magn = 1, normalize = FALSE)
    cfs <- list(cf_se, cf_lin)
  }
  gp <- gplite::gp_init(cfs = cfs, lik = gplite::lik_bernoulli(), approx = approx_method)
  gp <- gplite::gp_optim(gp, x = X_train, y = y_bin, maxiter = 1000, restarts = 3, tol = 1e-05)
  return(gp)
}
```

R code. Doubly robust Bayesian inference: Simulation exercise

```
posterior_draws_prob <- function(gp, X_test, ndraws = 5000) {
  out <- gplite::gp_draw(gp, xnew = X_test, draws = ndraws, transform = TRUE, target = FALSE, jitter = 1e-6)
  return(t(out))
}
# Main function
run_trim <- function(t_trim = 0.10, N_post = 5000, h_r = 1/2, cor_size = 1) {
  # Trim by estimated PS
  keep <- which(ps_hat >= t_trim & ps_hat <= (1 - t_trim))
  n_eff <- length(keep)
  if (n_eff < 50) stop("Too few observations after trimming.")
  Yk  <- Y[keep]
  Dk  <- D[keep]
  Zk  <- Z[keep, , drop = FALSE]
  psk <- ps_hat[keep]
  # Riesz representer and sigma_n choice
  gamma_k <- riesz_fun(Dk, psk)
  sigma_n <- cor_size * (log(n_eff) / ( (n_eff)^(h_r) * mean(abs(gamma_k)) ))
  gamma_scaled <- sigma_n * gamma_k
  # Training inputs for GP: [Z, D] and, if corrected, an extra column 'gamma'
  X_train_nc <- data.frame(Zk)
  X_train_nc$D <- Dk
  colnames(X_train_nc) <- c(covars, "D")
  X_train_c <- X_train_nc
  X_train_c$gamma <- gamma_scaled
  # Test inputs: for each i, (Z_i, d=0) and (Z_i, d=1)
  X_t0 <- X_train_nc; X_t0$D <- 0
  X_t1 <- X_train_nc; X_t1$D <- 1
  X_t0c <- X_t0; X_t1c <- X_t1
  X_t0c$gamma <- sigma_n * riesz_fun(0, psk)
  X_t1c$gamma <- sigma_n * riesz_fun(1, psk)
  # GP WITHOUT prior correction #
  gp_nc <- make_gp(X_train = X_train_nc, y_bin = Yk, use_correction = FALSE)
  # joint draws for [m(Z,0); m(Z,1)]
  M_nc_0 <- posterior_draws_prob(gp_nc, X_t0, ndraws = N_post)  # N_post x n_eff
  M_nc_1 <- posterior_draws_prob(gp_nc, X_t1, ndraws = N_post)
  # observed arm probabilities
  M_nc_obs <- ifelse(matrix(Dk, nrow = N_post, ncol = n_eff, byrow = TRUE) == 1, M_nc_1, M_nc_0)
  # GP WITH prior correction #
  gp_c <- make_gp(X_train = X_train_c, y_bin = Yk, use_correction = TRUE, gamma_scaled = gamma_scaled)
  M_c_0 <- posterior_draws_prob(gp_c, X_t0c, ndraws = N_post)
  M_c_1 <- posterior_draws_prob(gp_c, X_t1c, ndraws = N_post)
  M_c_obs <- ifelse(matrix(Dk, nrow = N_post, ncol = n_eff, byrow = TRUE) == 1, M_c_1, M_c_0)
```

R code. Doubly robust Bayesian inference: Simulation exercise

```
  # Bayesian bootstrap weights #
  W <- matrix(rexp(N_post * n_eff, rate = 1), nrow = N_post)
  W <- W / rowSums(W)
  # ATEs
  # (1) Unadjusted Bayes
  Ate_nc_draws <- rowSums((M_nc_1 - M_nc_0) * W)
  # (2) Prior-adjusted Bayes
  Ate_c_draws  <- rowSums((M_c_1 - M_c_0) * W)
  # (3) DR Bayes (posterior recentering)
  # Pre-centering using UNcorrected GP posterior means
  mu_nc_diff <- colMeans(M_nc_1 - M_nc_0)
  mu_nc_obs  <- colMeans(M_nc_obs)
  ATE_dr_pre <- mean(mu_nc_diff + gamma_k * (Yk - mu_nc_obs)
    )
  # Recenter draws using corrected GP draws
  DR_rec_1 <- (matrix(gamma_k, nrow = N_post, ncol = n_eff,
    byrow = TRUE)) * (matrix(Yk, nrow = N_post, ncol = n_eff
    , byrow = TRUE) - M_c_obs)
  # The first component is \tau_{\eta}^s in Algorithm 1
  # The second component is \hat{m} a scalar. This has
    positive sign because
  # minus x minus, the bias is with minus, and then, this
    component enters with minus in the bias term
  # The third component is \gamma_k x (y-m(d,x)), m(d,x) is
    with prior correction
  # The fourth component is m(1,x)-m(0,x) also with
    correction in the prior
  Ate_drb_draws <- rowSums((M_c_1 - M_c_0) * W) + ATE_dr_pre
     - rowSums(DR_rec_1) / n_eff - rowSums(M_c_1 - M_c_0) /
    n_eff

  qfun <- function(x) {
    qs <- quantile(x, c(0.025, 0.975))
    c(mean   = mean(x),   median = median(x), lo = qs[1], hi
     = qs[2], len = diff(qs)
    )
  }
  vals <- list(
  Bayes      = qfun(Ate_nc_draws),
  `PA Bayes` = qfun(Ate_c_draws),
  `DR Bayes` = qfun(Ate_drb_draws)
  )
  out_df <- as.data.frame(do.call(rbind, vals), check.names
    = FALSE)
  list(t_trim = t_trim, results = out_df)
}
N_post <- 5000
h_r    <- 1/2
cor_size <- 1
trim <- 0.05
res_list <- run_trim(t_trim = trim, N_post = N_post, h_r = h
    _r, cor_size = cor_size)
print(res_list$results)
```

14

Approximate Bayesian methods

Solutions of Exercises

1. **g-and-k distribution for financial returns continues I**

   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.[1]

   - 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.

   Answer:

   The following code implements our ABC-AR algorithm, following the steps outlined in the ABC-AR Algorithm of the book. Users can observe that a linear regression adjustment is applied; however, due to multicollinearity issues, one summary statistic is omitted in the prediction step (see the *abc* package for alternative regression adjustments).

   Figure 14.1 presents the posterior distributions of θ_1, g, and k using both our ABC implementations and the *EasyABC* package. We observe that our algorithms provide less information about θ_1 compared to the *EasyABC* implementation. Conversely, our algorithms offer more information about g and k than the *EasyABC* function. Additionally, regression adjustment helps center the posterior distributions of g and k around the population parameters. But, this is no the case regarding θ_1. Moreover, applying regression adjustment introduces a degree of smoothness to the distributions.

[1] Note that this setting does not satisfy the asymptotic requirements for Bayesian consistency. However, it serves as a pedagogical exercise.

Overall, in all implementations, the 95% credible intervals encompass the population values.

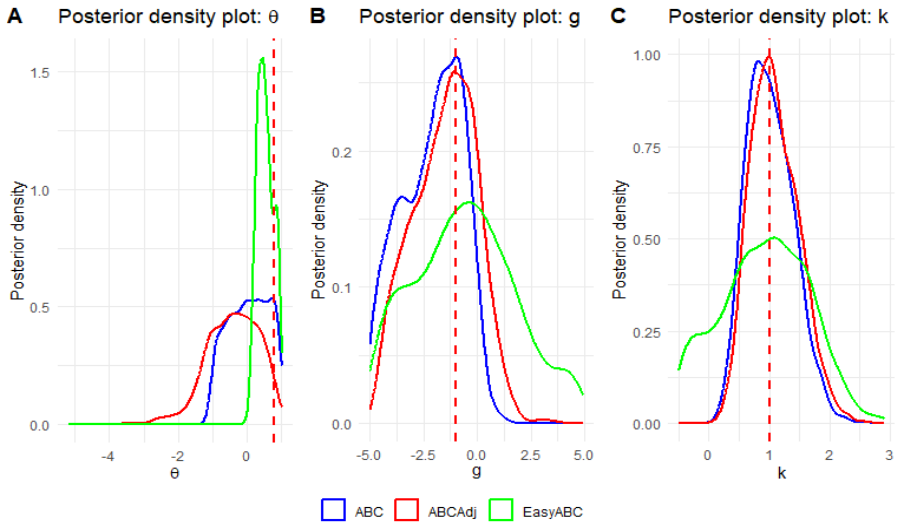

FIGURE 14.1
Posterior distributions using approximate Bayesian computation accept/reject: g-and-k distribution.

R code. Approximate Bayesian computation accept/reject algorithm: g-and-k distribution simulation

```
rm(list = ls()); set.seed(010101)
# 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)
}
# Population parameters
theta1 <- 0.8; a <- 1; b <- 0.5; g <- -1; k <- 1
parpop <- c(theta1, a, b, g, k)
n <- 500
y <- RGKnew(par = parpop)
##### ABC Function#####
ABC <- function(S, a, y) {
  prior <- cbind(runif(S,-1,1), runif(S,0,5), runif(S,0,5),
    runif(S,-5,5), runif(S,-0.5,5))
  Z <- apply(prior, 1, RGKnew)
  EtasZ <- apply(Z, 2, SumSt)
  Etay <- SumSt(y)
  Dist <- sapply(1:S, function(l) {
    dist(rbind(Etay, EtasZ[, l]))
  })
  OrdPrior <- prior[order(Dist), ]
  SelPrior <- OrdPrior[1:round(S * a), ]
  SelSumSt <- t(EtasZ)[1:round(S * a), ]
  return(list(SelPrior = SelPrior, SelSumSt = SelSumSt))
}
S <- 1000000
a <- 0.001
tick <- Sys.time()
ResABC <- ABC(S = S, a = 0.001, y = y)
tock <- Sys.time()
tock - tick
PostABC_ARown <- ResABC[["SelPrior"]]
```

R code. Approximate Bayesian computation accept/reject algorithm: g-and-k distribution simulation

```
# Regression adjusted ABC
X <- ResABC[["SelSumSt"]]-matrix(SumSt(y), S*a, 12, byrow =
    TRUE)
PostABC_ARownRegAd <- PostABC_ARown
for(j in 1:5){
  Reg <- lm(PostABC_ARown[,j] ~ X)
  # Coefficient of regressor 9 is na.
  PostABC_ARownRegAd[,j] <- PostABC_ARown[,j] - X[,-9]%*%Reg
    $coefficients[-c(1,9)]
}
RGKnewSum <- function(par) {
  z <- NULL
  theta <- par[1]; a <- par[2]; b <- par[3]; g <- par[4]; k
    <- par[5]
  e <- rnorm(n + 1)
  for(t in 2:(n + 1)){
    zt <- e[t] + theta * e[t-1]
    z <- c(z, zt)
  }
  zs <- z / (1 + theta^2)^0.5
  x <- a + b * (1 + 0.8 * (1 - exp(-g * zs)) / (1 + exp(-g *
     zs))) * (1 + zs^2)^k * zs
  Etaz <- SumSt(x)
  return(Etaz)
}
sum_stat_obs <- SumSt(y)
toy_prior <- list(c("unif",-1,1), c("unif",0,5), c("unif",
    0,5), c("unif", -5,5), c("unif", -0.5,5))
library(EasyABC)
tick <- Sys.time()
ABC_AR <- ABC_rejection(model=RGKnewSum, prior=toy_prior,
summary_stat_target = sum_stat_obs, nb_simul=260000, tol =
    0.00385,
progress_bar = TRUE)
tock <- Sys.time()
tock - tick
PostABC_AR <- coda::mcmc(ABC_AR$param)
# Summary
summary(coda::mcmc(PostABC_ARown))
summary(coda::mcmc(PostABC_ARownRegAd))
summary(coda::mcmc(PostABC_AR))
```

R code. Approximate Bayesian computation accept/reject algorithm: g-and-k distribution simulation

```
library(ggplot2); library(latex2exp)
Sp <- 1000

df1 <- data.frame(Value = c(PostABC_AR[1:Sp,1], PostABC_
    ARown[1:Sp,1], PostABC_ARownRegAd[1:Sp,1]), Distribution
     = factor(c(rep("EasyABC", Sp), rep("ABC", Sp), rep("
    ABCAdj", Sp))))
dentheta <- ggplot(df1, aes(x = Value, color = Distribution)
    ) + geom_density(linewidth = 1) + geom_vline(xintercept
    = theta1, linetype = "dashed", color = "red", 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(PostABC_AR[1:Sp,4], PostABC_
    ARown[1:Sp,4], PostABC_ARownRegAd[1:Sp,4]), Distribution
     = factor(c(rep("EasyABC", Sp), rep("ABC", Sp), rep("
    ABCAdj", Sp))))
deng <- ggplot(df2, aes(x = Value, color = Distribution)) +
      geom_density(linewidth = 1) + geom_vline(xintercept =
    g, linetype = "dashed", color = "red", linewidth = 1) +
    labs(title = TeX("Posterior density plot: g"), x = TeX("
    $g$"), y = "Posterior density") +
scale_color_manual(values = c("blue", "red", "green")) +
    theme_minimal() + theme(legend.title = element_blank())

df3 <- data.frame(Value = c(PostABC_AR[1:Sp,5], PostABC_
    ARown[1:Sp,5], PostABC_ARownRegAd[1:Sp,5]), Distribution
     = factor(c(rep("EasyABC", Sp), rep("ABC", Sp), rep("
    ABCAdj", Sp))))
denk <- ggplot(df3, aes(x = Value, color = Distribution)) +
      geom_density(linewidth = 1) + geom_vline(xintercept =
    k, linetype = "dashed", color = "red", linewidth = 1) +
labs(title = TeX("Posterior density plot: k"), x = TeX("$k$"
    ), 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)
```

2. **Simulation: g-and-k distribution continues**

   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.

   Answer:

   The following code shows how to perform inference in this exercise. The results using the semi-parametric and misspecified settings in mean are very different compared to the other methods. In general, the performance of these methods are not good in this particular exercise and setting.

   Figure 14.2 shows the posterior plots of the vanilla, unbiased and misspecified in variance settings. All the 95% credible intervals encompass the population variables; the posterior distributions of a, b and k are really good.

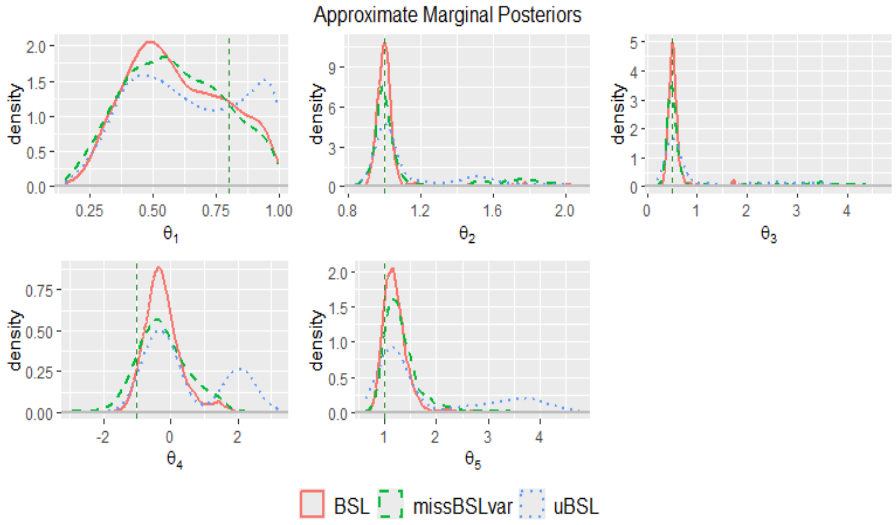

FIGURE 14.2
Posterior distributions using Bayesian synthetic likelihood: Simulation exercise g-and-k distribution.

R code. Bayesian synthetic likelihood: Simulation exercise g-and-k distribution.

```
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[4], 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 <- 1100 # Number of MCMC iterations
burnin <- 100 # Burn in iterations
thin <- 2 # 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)
keep <- seq(burnin + 1, S, thin)
tune <- 0.1 # Tuning parameter RW MH
simgk <- simulation(Modelgk, n = M, theta = par0, seed = 10)
par(mfrow = c(2, 3))
```

R code. Bayesian synthetic likelihood: Simulation exercise g-and-k distribution.

```
# Check if the summary statistics are roughly normal
for (i in 1:6){
  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")
}
Lims <- matrix(c(-1, 0, 0, -5, -0.5, 1, rep(5, 4)), 5, 2)
tick <- Sys.time()
Resultsgk <- bsl(y = y, n = M, M = S, model = Modelgk,
    covRandWalk = tune*diag(5),
method = "BSL", thetaNames = expression(theta, a, b, g, k),
logitTransformBound = Lims, plotOnTheFly = TRUE)
tock <- Sys.time()
tock - tick
PostChain <- coda::mcmc(Resultsgk@theta[keep,])
CovarRWnew <- var(PostChain)
M <- 500 # Number of iterations to calculate mu and sigma
S <- 6000 # Number of MCMC iterations
burnin <- 1000 # Burn in iterations
thin <- 5 # Thining parameter
keep <- seq(burnin + 1, S, thin)
tune <- 1 # Tuning parameter RW MH
ModelgkNew <- newModel(fnSim = RGKnew, fnSum = SumSt, theta0
     = par0, fnLogPrior = LogPrior, verbose = FALSE)
logitTransform <- function(par, a, b){
  logtrans <- log((par - a)/(b - par))
  return(logtrans)
}
ParTrans <- matrix(NA, dim(PostChain)[1], 5)
for(j in 1:5){
  ParTrans[,j] <- logitTransform(par = PostChain[,j], a =
    Lims[j,1], b = Lims[j,2])
}
CovarRW <- var(ParTrans)
# Vanilla BLS
tick <- Sys.time()
ResultsgkVanila <- bsl(y = y, n = M, M = S, model =
    ModelgkNew, covRandWalk = tune*CovarRW, method = "BSL",
    thetaNames = expression(theta, a, b, g, k),
    logitTransformBound = Lims, plotOnTheFly = TRUE)
tock <- Sys.time()
tock - tick
# Unbiased BLS
tick <- Sys.time()
ResultsgkuBLS <- bsl(y = y, n = M, M = S, model = ModelgkNew
    , covRandWalk = tune*CovarRW,
method = "uBSL", thetaNames = expression(theta, a, b, g, k),
logitTransformBound = Lims, plotOnTheFly = TRUE)
tock <- Sys.time()
tock - tick
```

R code. Bayesian synthetic likelihood: Simulation exercise g-and-k distribution.

```
# Semi-parametric BLS
tick <- Sys.time()
ResultsgksemiBLS <- bsl(y = y, n = M, M = S, model = ModelgkNew, covRandWalk = tune*CovarRW, method = "semiBSL", thetaNames = expression(theta, a, b, g, k), logitTransformBound = Lims, plotOnTheFly = TRUE)
tock <- Sys.time()
tock - tick
# Misspecified BLS in mean
tick <- Sys.time()
ResultsgkmisspecBLSmean <- bsl(y = y, n = M, M = S, model = ModelgkNew, covRandWalk = tune*CovarRW, method = "BSLmisspec", thetaNames = expression(theta, a, b, g, k), logitTransformBound = Lims, misspecType = "mean", tau = 0.5, plotOnTheFly = TRUE)
tock <- Sys.time()
tock - tick
# Misspecified BLS in variance
tick <- Sys.time()
ResultsgkmisspecBLSvar <- bsl(y = y, n = M, M = S, model = ModelgkNew, covRandWalk = tune*CovarRW, method = "BSLmisspec", thetaNames = expression(theta, a, b, g, k), logitTransformBound = Lims, misspecType = "variance", tau = 0.5, plotOnTheFly = TRUE)
tock <- Sys.time()
tock - tick
# Summary all models
gkResults <- list(ResultsgkVanila, ResultsgkuBLS, ResultsgksemiBLS, ResultsgkmisspecBLSmean, ResultsgkmisspecBLSvar)
names(gkResults) <- c("BSL", "uBSL", "semiBLS", "missBSLmean", "missBSLvar")
t(sapply(gkResults, summary))
combinePlotsBSL(gkResults, which = 2, thetaTrue = parpop, thin = thin, options.linetype = list(values = 1:8), options.size = list(values = rep(1, 8)), options.theme = list(plot.margin = grid::unit(rep(0.03, 4), "npc"), axis.title = ggplot2::element_text(size = 12), axis.text = ggplot2::element_text(size = 8), legend.text = ggplot2::element_text(size = 12)))
# semi BSL and miss BSL mean are totally diferent to other methods
gkResultsNew <- list(gkResults[["BSL"]], gkResults[["uBSL"]], gkResults[["missBSLvar"]])
names(gkResultsNew) <- c("BSL", "uBSL", "missBSLvar")
combinePlotsBSL(gkResultsNew, which = 2, thetaTrue = parpop, thin = thin, options.linetype = list(values = 1:8), options.size = list(values = rep(1, 8)), options.theme = list(plot.margin = grid::unit(rep(0.03, 4), "npc"), legend.text = ggplot2::element_text(size = 12)))
```

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 [41] for details).

   Answer:

R code. Multinomial model in INLA: Simulation exercise.

```
rm(list = ls()); set.seed(010101)
library(INLA)
beta1 <- -0.3; beta2 <- 1.2; gamma <- c(0.3, 0, 0.5)
param <- c(beta1, beta2, gamma); n <- 1000
# Alternative specific regressors
x11 <- rnorm(n); x12 <- rnorm(n); x13 <- rnorm(n)
x21 <- rnorm(n); x22 <- rnorm(n); x23 <- rnorm(n)
# Individual specific regressor
Z <- rnorm(n)
Multinom.sample = function(N){
  Y <- matrix(NA, ncol = 3, nrow = n)
  for(i in 1:n){
    V1 <- beta1*x11[i] + beta2*x21[i] + gamma[1]*Z[i]
    V2<- beta1*x12[i] + beta2*x22[i] + gamma[2]*Z[i]
    V3 <- beta1*x13[i] + beta2*x23[i] + gamma[3]*Z[i]
    probs <- c(V1, V2, V3)
    probs <- exp(probs)/sum(exp(probs))
    samp <- rmultinom(1, N, prob = probs)
    Y[i,] = as.vector(samp)
  }
  colnames(Y ) = c("Y1", "Y2", "Y3")
  return(Y)
}
Y <- Multinom.sample(1)
df <- data.frame(cbind(Y, x11, x12, x13, x21, x22, x23, Z))
Data.structure = function(df){
  Data = matrix(NA, ncol = 6, nrow = n*3)
  for(i in 1:n){
    # simulated variable
    Data[((i-1)*3+1):(i*3), 1] = c(df$Y1[i], df$Y2[i], df$Y3
    [i])
    # alternative specific with generic coeff
    Data[((i-1)*3+1):(i*3), 2] = c(df$x11[i], df$x12[i], df$
    x13[i])
    # alternative specific with generic coeff
    Data[((i-1)*3+1):(i*3), 3] = c(df$x21[i], df$x22[i], df$
    x23[i])
    # individual specific with alternative coeff
    Data[((i-1)*3+1):(i*3), 4] = rep(df$Z[i],3)
    # choice situation index
    Data[((i-1)*3+1):(i*3), 5] = rep(i,3)
    # choice alternative index
    Data[((i-1)*3+1):(i*3), 6] = c(1, 2, 3)
  }
  Data = data.frame(Data)
  names(Data) = c('Y', "X1", "X2", "Z",'phi','alt.idx')
  return(Data)
}
formula = Y ~ -1 + X1 + X2 + f(phi, initial = -10, fixed = T
    ) + f(alt.idx, Z, fixed = T, constr = T)
Data = Data.structure(df)
model = inla(formula, data = Data, family = 'Poisson')
result = rbind(model$summary.fixed[1:5], model$summary.
    random$alt.idx[2:6])
result = cbind(result, true = param)
```

R code. Multinomial model in INLA: Simulation exercise.

```
row.names(result) = c("beta1", "beta2", "gamma.A","gamma.B",
    "gamma.C" )
round(result,3)
diff.result = cbind("0.025quant"= diff(model$summary.random$
    alt.idx$`0.025quant`), "0.5quant" = diff(model$summary.
    random$alt.idx$`0.5quant`), "0.975quant" = diff(model$
    summary.random$alt.idx$`0.975quant`), "true" = diff(
    gamma))
row.names(diff.result) = c("gamma.B - gamma.A", "gamma.C -
    gamma.B")
round(diff.result,3)
```

4. Get the expression for the ELBO in the linear regression model with conjugate family.

   Answer:

$$
\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)] \\
&= \mathbb{E}_{\boldsymbol{\beta},\sigma^2}[\log p(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{X}) + \log \pi(\boldsymbol{\beta} \mid \sigma^2) + \log(\sigma^2)] \\
&- \mathbb{E}_{\boldsymbol{\beta},\sigma^2}[\log q(\boldsymbol{\beta}) + \log q(\sigma^2)] \\
&= \mathbb{E}_{\boldsymbol{\beta},\sigma^2}\left[-\frac{N}{2}\log(2\pi) - \frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\right. \\
&- \frac{K}{2}\log(2\pi) - \frac{K}{2}\log(\sigma^2) - \frac{1}{2}\log|\boldsymbol{B}_0| - \frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \boldsymbol{B}_0^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_0) \\
&- \left.(\alpha_0/2 + 1)\log\sigma^2 - \frac{\delta_0}{2\sigma^2} + \alpha_0/2\log(\delta_0/2) - \log\Gamma(\alpha_0/2)\right] \\
&- \left\{\mathbb{E}_{\boldsymbol{\beta},\sigma^2}\left[\frac{K}{2}\log\mathbb{E}_{\sigma^2}(1/\sigma^2) - \frac{1}{2}\log|\boldsymbol{B}_n| - \frac{1}{2}\mathbb{E}_{\sigma^2}(1/\sigma^2)(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)\right.\right. \\
&\left.\left. - \frac{K}{2}\log(2\pi) - (\alpha_n/2 + 1)\log\sigma^2 - \frac{\delta_n}{2\sigma^2} + \alpha_n/2\log(\delta_n/2) - \log\Gamma(\alpha_n/2)\right]\right\} \\
&= -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{B}_0| + \frac{1}{2}\log|\boldsymbol{B}_n| + \alpha_0/2\log(\delta_0/2) \\
&- \alpha_n/2\log(\delta_n/2) - \log\Gamma(\alpha_0/2) + \log\Gamma(\alpha_n/2) - \frac{K}{2}\log\mathbb{E}_{\sigma^2}(1/\sigma^2) \\
&+ \frac{1}{2}\mathbb{E}_{\sigma^2}(1/\sigma^2)\mathbb{E}_{\boldsymbol{\beta}}\left[(\boldsymbol{\beta} - \boldsymbol{\beta}_n)^\top \boldsymbol{B}_n^{-1}(\boldsymbol{\beta} - \boldsymbol{\beta}_n)\right] \\
&= -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{B}_0| + \frac{1}{2}\log|\boldsymbol{B}_n| + \alpha_0/2\log(\delta_0/2) \\
&- \alpha_n/2\log(\delta_n/2) - \log\Gamma(\alpha_0/2) + \log\Gamma(\alpha_n/2) - \frac{K}{2}\log(\alpha_n/\delta_n) \\
&+ 0.5tr\left\{Var(\boldsymbol{\beta})\boldsymbol{B}_n^{-1}\right\}
\end{aligned}
$$

We take into account in the fourth equality that the terms collecting $\log\sigma^2$ and $\frac{1}{2\sigma^2}$ cancel out by the definitions of α_n and δ_n, respectively.

5. **Linear regression continues**

   Perform inference in the linear regression example using stochastic variational inference via the automatic differentiation variational inference (ADVI) approach [23], 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 $\boldsymbol{\beta}_0 = \boldsymbol{0}_3$, $\boldsymbol{B}_0 = 1000\boldsymbol{I}_3$, and $\alpha_0 = \delta_0 = 0.01$. The sample size is one hundred thousand.

   Answer:

   The posterior mean values are very close to the population parameters, but the 95% credible intervals are too narrow and do not contain the true values.

R code. Stochastic variational inference: Simulation exercise linear regression.

```r
library(rstan)
# --- Simulated data ---
N <- 100000; K <- 2; B <- rep(1, K + 1); sig2 <- 1
X <- cbind(1, matrix(rnorm(N * K), N, K))
y <- X %*% B + rnorm(N, 0, sig2^0.5)
# --- Priors ---
b0 <- 0; B0 <- 1000; a0 <- 0.01; d0 <- 0.01
# --- Prepare data for Stan ---
data_list <- list(
N = N,
K = K + 1,  # K+1 because of the intercept
X = X,
y = as.vector(y),
b0 = b0,
B0 = B0,
a0 = a0,
d0 = d0
)
# --- Stan model compilation ---
stan_model_code <- "
data {
  int<lower=0> N;        // number of data points
  int<lower=0> K;        // number of predictors
  matrix[N, K] X;        // predictor matrix
  vector[N] y;           // response vector
  real b0;               // location parameter prior on beta
  real<lower=0> B0;      // variance parameter prior on beta
  real<lower=0> a0;      // shape parameter prior on sigma^2
  real<lower=0> d0;      // rate parameter prior on sigma^2
}
parameters {
  vector[K] beta;        // coefficients of the linear model
  real<lower=0> sigma2;  // variance parameter (sigma^2)
}
model {
  // Priors
  beta ~ normal(b0, B0);       // Normal prior on beta
  sigma2 ~ inv_gamma(a0/2, d0/2); // Inverse Gamma prior

  // Likelihood
  y ~ normal(X * beta, sqrt(sigma2)); // Normal likelihood
    with linear predictor
}
generated quantities {
  vector[N] y_pred;
  for (n in 1:N) {
    y_pred[n] = normal_rng(dot_product(X[n], beta), sqrt(
    sigma2));
  }
}
"
```

R code. Stochastic variational inference: Simulation exercise linear regression.

```
# Compile Stan model
stan_model <- stan_model(model_code = stan_model_code)
# --- Perform Stochastic Variational Inference (SVI) ---
fit <- vb(stan_model, data = data_list, algorithm = "
    meanfield", iter = 10000)
# --- Extract the results ---
mu_estimate <- extract(fit)$beta
summary(coda::mcmc(mu_estimate))
sigma2_estimate <- extract(fit)$sigma2
summary(coda::mcmc(as.vector(sigma2_estimate)))

```

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.

Answer:

The likelihood function in this model is $p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2) = \prod_{i=1}^N \prod_{h=1}^H \phi(y_i \mid \boldsymbol{x}_i^\top \boldsymbol{\beta}_h, \sigma_h^2)^{z_{ih}}$, and $\pi(\boldsymbol{z} \mid \boldsymbol{\lambda}) = \prod_{i=1}^N \prod_{h=1}^H \lambda_h^{z_{ih}}$, $\boldsymbol{\lambda} \sim Dir(\boldsymbol{a}_0)$, $\boldsymbol{\beta}_h \sim N(\boldsymbol{\beta}_{h0}, \boldsymbol{B}_{h0})$ and $\sigma_h^2 \sim IG(\alpha_{h0}/2, \delta_{h0}/2)$. In this notation, $\boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2$ and $\boldsymbol{\lambda}$ stacks all the latent indicators, all locations and scale parameters of each cluster, and all the probabilities, respectively.

Let's work with the mean-field variational family $q(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2) = q(\boldsymbol{z})q(\boldsymbol{\lambda}) \prod_{h=1}^H q(\boldsymbol{\beta}_h) \prod_{h=1}^H q(\sigma_h^2)$.

Then,

$$\begin{aligned}\log q(\boldsymbol{z}) =& \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{\beta},\boldsymbol{\sigma}^2}\left[\log p(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2)\right] + c_1 \\ =& \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{\beta},\boldsymbol{\sigma}^2}\left[\log p(\boldsymbol{y} \mid \boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2) + \log \pi(\boldsymbol{z} \mid \boldsymbol{\lambda})\right] + c_2 \\ =& \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{\beta},\boldsymbol{\sigma}^2}\left[\sum_{i=1}^{N}\sum_{h=1}^{H} z_{ih}\left\{-0.5\log 2\pi - 0.5\log\sigma_h^2 - \right.\right. \\ & \left.\left. -\frac{1}{2\sigma_h^2}(y_i - \boldsymbol{x}_i\boldsymbol{\beta}_h)^2 + \log\lambda_h\right\}\right] + c_2 \\ =& \sum_{i=1}^{N}\sum_{h=1}^{H}\psi_{ih}\log\rho_{ih} + c_2.\end{aligned}$$

Thus, $q(\boldsymbol{z}) = \prod_{i=1}^{N}\prod_{h=1}^{H} r_{ih}^{z_{ih}}$, such that $\mathbb{E}_{z_{ih}}[z_{ih}] = r_{ih}$, where $r_{ih} = \frac{\rho_{ih}}{\sum_{j=1}^{H}\rho_{ij}}$, and

$$\log\rho_{ih} = -0.5\log 2\pi - 0.5\mathbb{E}_{\sigma_h^2}\log\sigma_h^2 - 0.5\mathbb{E}_{\sigma_h^2}\left(\frac{1}{\sigma_h^2}\right)\mathbb{E}_{\boldsymbol{\beta}_h}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2 + \mathbb{E}_{\lambda_h}\log\lambda_h.$$

$$\begin{aligned}\log q(\boldsymbol{\lambda}) &= \mathbb{E}_{\boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2}\left[\log p(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2)\right] + c_1 \\ &= \mathbb{E}_{\boldsymbol{z}}\left[\log\pi(\boldsymbol{z} \mid \boldsymbol{\lambda}) + \log\pi(\boldsymbol{\lambda})\right] + c_2 \\ &= \mathbb{E}_{\boldsymbol{z}}\left[\sum_{i=1}^{N}\sum_{h=1}^{H} z_{ih}\log\lambda_h + \sum_{h=1}^{H}(a_{h0}-1)\log\lambda_h\right] + c_3 \\ &= \sum_{i=1}^{N}\sum_{h=1}^{H}\mathbb{E}_{z_{ih}}[z_{ih}]\log\lambda_h + \sum_{h=1}^{H}(a_{h0}-1)\log\lambda_h + c_3 \\ &= \sum_{i=1}^{N}\sum_{h=1}^{H} r_{ih}\log\lambda_h + \sum_{h=1}^{H}(a_{h0}-1)\log\lambda_h + c_3 \\ &= \sum_{h=1}^{H}\left(\sum_{i=1}^{N} r_{ih} + a_{h0} - 1\right)\log\lambda_h + c_3.\end{aligned}$$

This implies that $q(\boldsymbol{\lambda})$ is Dirichlet with parameters $a_{hn} = a_{h0} + \sum_{i=1}^{N} r_{ih}$.

$$\begin{aligned}\log q(\boldsymbol{\beta}_h) = \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{z},\boldsymbol{\sigma}^2}\left[\log p(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2)\right] + c_1 \\ =\mathbb{E}_{\boldsymbol{z},\boldsymbol{\sigma}^2}\left[\log p(\boldsymbol{y} \mid \boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2) + \log\pi(\boldsymbol{\beta}_h)\right] + c_2 \\ = \mathbb{E}_{\boldsymbol{z},\boldsymbol{\sigma}^2}\left[\sum_{i=1}^{N}\sum_{h=1}^{H} z_{ih}\left\{-\frac{1}{2\sigma_h^2}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2\right\}\right. \\ \left. -0.5(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{h0})^\top\boldsymbol{B}_{h0}^{-1}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{h0})\right] + c_3.\end{aligned}$$

Following same steps as those to obtain the posterior distribution,

and taking into account the expected values, we have that $q(\boldsymbol{\beta}_h)$ is $N(\boldsymbol{\beta}_{hn}, \boldsymbol{B}_{hn})$, where $\boldsymbol{B}_{hn} = \left(\boldsymbol{B}_{h0}^{-1} + \mathbb{E}_{\sigma_h^2}\left(\frac{1}{\sigma_h^2}\right)\sum_{i=1}^N r_{ih}\boldsymbol{x}_i\boldsymbol{x}_i^\top\right)^{-1}$ and $\boldsymbol{\beta}_{hn} = \boldsymbol{B}_{hn}\left(\boldsymbol{B}_{h0}^{-1}\boldsymbol{\beta}_{h0} + \mathbb{E}_{\sigma_h^2}\left(\frac{1}{\sigma_h^2}\right)\sum_{i=1}^N r_{ih}\boldsymbol{x}_i y_i\right)$.

$$
\begin{aligned}
\log q(\sigma_h^2) &= \mathbb{E}_{\boldsymbol{\lambda},\boldsymbol{z},\boldsymbol{\beta}}\left[\log p(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2)\right] + c_1 \\
&= \mathbb{E}_{\boldsymbol{z},\boldsymbol{\beta}}\left[\log p(\boldsymbol{y} \mid \boldsymbol{z},\boldsymbol{\beta},\boldsymbol{\sigma}^2) + \log \pi(\sigma_h^2)\right] + c_2 \\
&= \mathbb{E}_{\boldsymbol{z},\boldsymbol{\beta}}\left[\sum_{i=1}^N\sum_{h=1}^H z_{ih}\left\{-0.5\log\sigma_h^2 - \frac{1}{2\sigma_h^2}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2\right\}\right. \\
&\quad \left. -(\alpha_{h0}/2+1)\log\sigma_h^2 - \frac{\delta_{h0}}{2\sigma_h^2}\right] + c_1.
\end{aligned}
$$

Collecting terms we see that $q(\sigma_h^2)$ is $IG(\alpha_{hn}/2, \delta_{hn}/2)$, where $\alpha_{hn} = \sum_{i=1}^N r_{ih} + \alpha_{h0}$ and $\delta_{hn} = \sum_{i=1}^N r_{ih}\mathbb{E}_{\boldsymbol{\beta}_h}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2 + \delta_{h0}$.

Note that the previous results imply that $\mathbb{E}_{\sigma_h^2}\left(\frac{1}{\sigma_h^2}\right) = \alpha_{hn}/\delta_{hn}$, $\mathbb{E}_{\boldsymbol{\beta}_h}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2 = (y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_{hn})^2 + \boldsymbol{x}_i^\top\boldsymbol{B}_{hn}\boldsymbol{x}_i$, and

$$
\begin{aligned}
\log\rho_{ih} &= -0.5\log 2\pi - 0.5\mathbb{E}_{\sigma_h^2}\log\sigma_h^2 - 0.5\mathbb{E}_{\sigma_h^2}\left(\frac{1}{\sigma_h^2}\right)\mathbb{E}_{\boldsymbol{\beta}_h}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2 + \mathbb{E}_{\lambda_h}\log\lambda_h \\
&= -0.5\log 2\pi - 0.5(\log(\delta_{hn}/2) - \psi(\alpha_{hn}/2)) - 0.5(\alpha_{hn}/\delta_{hn})(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_{hn})^2 \\
&\quad + \boldsymbol{x}_i^\top\boldsymbol{B}_{hn}\boldsymbol{x}_i^\top + \psi(a_{hn}) - \psi\left(\sum_{h=1}^H a_{hn}\right),
\end{aligned}
$$

where ψ is the digamma function.

In addition, the evidence lower bound (ELBO) is given by

$$
\begin{aligned}
\text{ELBO}(q(\boldsymbol{\theta})) &= \mathbb{E}_q\left[\log p(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2)\right] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2)] \\
&= \mathbb{E}_q\left[\log p(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2) + \log \pi(\boldsymbol{z} \mid \boldsymbol{\lambda}) + \log \pi(\boldsymbol{\lambda}) + \sum_{h=1}^{H}(\log \pi(\boldsymbol{\beta}_h) + \log \pi(\sigma_h^2))\right] \\
&- \mathbb{E}_q\left[\log q(\boldsymbol{z}) + \log \pi(\boldsymbol{\lambda}) + \sum_{h=1}^{H}(\log q(\boldsymbol{\beta}_h) + \log \pi(\sigma_h^2))\right] \\
&= \mathbb{E}_q\left[\sum_{i=1}^{N}\sum_{h=1}^{H} z_{ih}\left\{-0.5\log(2\pi) - 0.5\log\sigma_h^2 - 0.5\frac{1}{\sigma_h^2}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_h)^2\right\}\right. \\
&+ \sum_{i=1}^{N}\sum_{h=1}^{H} z_{ih}\log\lambda_h + \sum_{h=1}^{H}\left\{-\frac{K}{2}\log(2\pi) - 0.5\log|\boldsymbol{B}_{h0}| - 0.5(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{h0})^\top\boldsymbol{B}_{h0}^{-1}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{h0})\right\} \\
&\left.+ \sum_{h=1}^{H}\left\{-(\alpha_{h0}/2+1)\log\sigma_h^2 - \frac{\delta_{h0}}{2\sigma_h^2} + (\alpha_{h0}/2)\log(\delta_{h0}/2) - \log\Gamma(\alpha_0/2)\right\}\right] \\
&- \mathbb{E}_q\left[\sum_{i=1}^{N}\sum_{h=1}^{H} z_{ih}\log r_{ih} - \log B(\boldsymbol{a}_n) + \sum_{h=1}^{H}(a_{hn}-1)\log\lambda_h\right. \\
&+ \sum_{h=1}^{H}\left\{-\frac{K}{2}\log(2\pi) - 0.5\log|\boldsymbol{B}_{hn}| - 0.5(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})^\top\boldsymbol{B}_{hn}^{-1}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})\right\} \\
&\left.+ \sum_{h=1}^{H}\left\{-(\alpha_{hn}/2+1)\log\sigma_h^2 - \frac{\delta_{hn}}{2\sigma_h^2} + (\alpha_{hn}/2)\log(\delta_{hn}/2) - \log\Gamma(\alpha_{hn}/2)\right\}\right].
\end{aligned}
$$

The first part of this expression, that is, $\mathbb{E}_q\left[\log p(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2)\right]$, is given by
$\sum_{i=1}^{N}\sum_{h=1}^{H} r_{ih}\left\{-0.5\log(2\pi) - 0.5(\log(\delta_{hn}/2) - \psi(\alpha_{hn}/2)) - \frac{\alpha_{hn}}{2\delta_{hn}}[(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}_{hn})^2 + \boldsymbol{x}_i^\top\boldsymbol{B}_{hn}\boldsymbol{x}_i]\right\}+$
$\sum_{i=1}^{N}\sum_{h=1}^{H} r_{ih}\log r_{it}+\sum_{h=1}^{H}\left\{-\frac{K}{2}\log(2\pi) - 0.5\log|\boldsymbol{B}_{h0}| - tr(\boldsymbol{B}_{hn}\boldsymbol{B}_{h0}^{-1}) - (\boldsymbol{\beta}_{hn} - \boldsymbol{\beta}_{h0})^\top(\boldsymbol{\beta}_{hn} - \boldsymbol{\beta}_{h0})\right\}+$
$\sum_{h=1}^{H}\left\{-(\alpha_{h0}/2+1)(\log(\delta_{hn}/2) - \psi(\alpha_{hn}/2)) - \frac{\delta_{h0}}{2}\frac{\alpha_{hn}}{\delta_{hn}} + (\alpha_{h0}/2)\log(\delta_{h0}/2) - \log\Gamma(\alpha_0/2)\right\}$.

The second part of the ELBO, that is, $\mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2)]$ is given by
$\sum_{i=1}^{N}\sum_{h=1}^{H} r_{ih}\log r_{ih} - \log B(\boldsymbol{a}_n) + \sum_{h=1}^{H}(a_{hn}-1)\left(\psi(a_{hn}) - \psi\left(\sum_{h=1}^{H} a_{hn}\right)\right)+$
$\sum_{h=1}^{H} -\frac{K}{2}(\log(2\pi)+1) - 0.5\log|\boldsymbol{B}_{hn}| + (\alpha_{hn}/2+1)\psi(\alpha_{hn}/2) - 0.5\alpha_{hn} -$
$\log(\delta_{hn}/2) - \log\Gamma(\alpha_{hn}/2)$. We use

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{\beta}_h}[(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})^\top\boldsymbol{B}_{hn}^{-1}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})] &= tr\left\{\mathbb{E}_{\boldsymbol{\beta}_h}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})^\top\boldsymbol{B}_{hn}^{-1}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})\right\} \\
&= \mathbb{E}_{\boldsymbol{\beta}_h}[tr\left\{(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})^\top\boldsymbol{B}_{hn}^{-1}(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})\right\}] \\
&= tr\left\{\mathbb{E}_{\boldsymbol{\beta}_h}[(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})(\boldsymbol{\beta}_h - \boldsymbol{\beta}_{hn})^\top\boldsymbol{B}_{hn}^{-1}]\right\} \\
&= tr\left\{Var(\boldsymbol{\beta}_h)\boldsymbol{B}_{hn}^{-1}\right\} \\
&= tr\left\{\boldsymbol{I}_K\right\} \\
&= K
\end{aligned}
$$

R code. Variational Bayes: Mixture Gaussian regression.

```
rm(list = ls())
set.seed(010101)
# Simulate data from a 2-component mixture model
n <- 1000
K <- 2; H <- 2
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))
X <- cbind(1,x)
# Iterations
S <- 10000
##### From scratch ######
# Hyperparameters
b0h <- rep(0, K); B0h <- diag(1000,K); B0hi <- solve(B0h)
a0h <- 0.01; d0h <- 0.01; a0 <- rep(1/H, H)
# Innitial values
dhn <- c(50, 50); ahn <- c(500, 500)
bhn <- matrix(c(2, 1.5, -1, 0.5), K, H)
Bhn <- array(c(1,0,0,1,1,0,0,1), c(K, K, H))
an <- c(500, 500)
VarParPsi <- function(dhn, ahn, bhn, Bhn, an){
  logrhoih <- matrix(NA, n, H)
  for(i in 1:n){
    for(h in 1:H){
      logrhoih[i,h] <- -0.5*log(2*pi) - 0.5*(log(dhn[h]/2) -
     digamma(ahn[h]/2)) -
      0.5*(ahn[h]/dhn[h])*((y[i] - t(X[i,])%*%bhn[,h])^2 + t
    (X[i,])%*%Bhn[,,h]%*%X[i,]) +
      digamma(an[h]) - digamma(sum(an))
    }
  }
  rhoih <- exp(logrhoih - apply(logrhoih, 1, max))
  rih <- rhoih / rowSums(rhoih)
  return(rih)
}
Rih <- VarParPsi(dhn = dhn, ahn = ahn, bhn = bhn, Bhn = Bhn,
     an = an)
VarParLambda <- function(rih){
  an <- a0 + colSums(rih)
  return(an)
}
VarParLambda(rih = Rih)
VarParBetah <- function(ahn, dhn, rh){
  rXX <- matrix(0, K, K)
  rXy <- matrix(0, K, 1)
  for(i in 1:n){
    rXXi <- rh[i]*X[i,]%*%t(X[i,])
    rXX <- rXX + rXXi
    rXyi <- rh[i]*X[i,]*y[i]
    rXy <- rXy + rXyi
  }
  Bhn <- solve(B0hi + ahn/dhn * rXX)
  bhn <- Bhn%*%(B0hi%*%b0h + ahn/dhn * rXy)
  return(list(bhn = bhn, Bhn = Bhn))
}
```

R code. Variational Bayes: Mixture Gaussian regression.

```
VarParSigma2h <- function(bhn, Bhn, rh){
  ahn <- a0h + sum(rh); dhnterm <- 0
  for(i in 1:n){
    dhntermi <- rh[i]*((y[i] - t(X[i,])%*%bhn)^2 + t(X[i,])%
    *%Bhn%*%X[i,])
    dhnterm <- dhnterm + dhntermi
  }
  dhn <- dhnterm + d0h; return(list(ahn = ahn, dhn = dhn))
}
multivariate_beta <- function(alpha) {
  prod_gammas <- sum(sapply(alpha, lgamma))
  sum_alpha <- sum(alpha)
  gamma_sum_alpha <- lgamma(sum_alpha)
  return(prod_gammas - gamma_sum_alpha)
}
ELBO <- function(rih, dhn, ahn, bhn, Bhn, an){
  LogLikh <- 0; LogPrioPsih <- 0
  LogPriorBeta <- 0; LogPriorSig2 <- 0
  for(h in 1:H){
    for(i in 1:n){
      LogLikih <- rih[i, h]*(-0.5*log(2*pi) - 0.5*(log(dhn[h
    ]/2) - digamma(ahn[h]/2)) - 0.5*(ahn[h]/dhn[h])*((y[i] -
     t(X[i,])%*%bhn[,h])^2 + t(X[i,])%*%Bhn[,,h]%*%X[i,]))
      LogLikh <- LogLikh + LogLikih
      LogPrioPsiih <- rih[i, h]*(digamma(an[h]) - digamma(
    sum(an)))
      LogPrioPsih <- LogPrioPsih + LogPrioPsiih
    }
    LogPriorBetah <- -0.5*K*log(2*pi) - 0.5*log(det(B0h)) -
    sum(diag(Bhn[,,h]%*%B0hi)) - t(bhn[,h] - b0h)%*%B0hi%*%(
    bhn[,h] - b0h)
    LogPriorBeta <- LogPriorBeta + LogPriorBetah
    LogPriorSig2h <- -(a0h/2+1)*(log(dhn[h]/2) - digamma(ahn
    [h]/2)) - 0.5*d0h*(ahn[h]/dhn[h]) + (a0h/2)*log(d0h/2) -
     lgamma(a0h/2)
    LogPriorSig2 <- LogPriorSig2 + LogPriorSig2h
  }
  LogJoint <- LogLikh + LogPrioPsih + LogPriorBeta +
    LogPriorSig2
  logqPsi <- 0; logqLambda <- 0; logqBeta <- 0;logqSig2 <- 0
  for(h in 1:H){
    for(i in 1:n){
      logqPsiih <- rih[i, h] * log(rih[i, h])
      logqPsi <- logqPsi + logqPsiih
    }
    logqLambdah <- (an[h] - 1)*(digamma(an[h]) - digamma(sum
    (an)))
    logqLambda <- logqLambda + logqLambdah
    logqBetah <- -0.5*K*log(2*pi + 1) - 0.5*log(det(Bhn[,,h
    ]))
    logqBeta <- logqBeta + logqBetah
    logqSig2h <- -log(dhn[h]/2) + (ahn[h]/2+1)*digamma(ahn[h
    ]/2) - ahn[h]/2 - lgamma(ahn[h]/2)
    logqSig2 <- logqSig2 + logqSig2h
  }
  logqLambda <- logqLambda - multivariate_beta(alpha = an)
  logq <- logqPsi + logqLambda + logqBeta + logqSig2
  ELBO <- LogJoint - logq
  return(ELBO)
}
```

R code. Variational Bayes: Mixture Gaussian regression.

```
# Initial values
fit_lm1 <- lm(y ~ x)
bhn[,1] <- coef(fit_lm1)
bhn[,2] <- coef(fit_lm1) + rnorm(K, 0, 0.5)
Bhn[,,1] <- diag(0.1, K); Bhn[,,2] <- diag(0.1, K)
ahn <- rep(10, H); dhn <- rep(10, H)
an <- c(300, 700)
ELBOs <- rep(NA, S)
ELBOs[1] <- ELBO(rih = Rih, dhn = dhn, ahn = ahn, bhn = bhn,
     Bhn = Bhn, an = an)
epsilon <- 1e-5
for(s in 2:S){
  Rih <- VarParPsi(dhn = dhn, ahn = ahn, bhn = bhn, Bhn =
    Bhn, an = an)
  an <- VarParLambda(rih = Rih)
  bhn <- matrix(NA, K, H)
  Bhn <- array(NA, c(K, K, H))
  for(h in 1:H){
    EnsVarParBetah <- VarParBetah(ahn = ahn[h], dhn = dhn[h
    ], rh = Rih[,h])
    bhn[,h] <- EnsVarParBetah[["bhn"]]
    Bhn[,,h] <- EnsVarParBetah[["Bhn"]]
  }
  ahn <- rep(NA, H)
  dhn <- rep(NA, H)
  for(h in 1:H){
    EnsVarParSig2h <- VarParSigma2h(bhn = bhn[,h], Bhn = Bhn
    [,,h], rh = Rih[,h])
    ahn[h] <- EnsVarParSig2h[["ahn"]]
    dhn[h] <- EnsVarParSig2h[["dhn"]]
  }
  ELBOs[s] <- ELBO(rih = Rih, dhn = dhn, ahn = ahn, bhn =
    bhn, Bhn = Bhn, an = an)
  # Check if lower bound decreases
  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")}
}
plot(ELBOs[1:s], type = "l", main = "ELBO over iterations",
     ylab = "ELBO", xlab = "Iteration")
Sig21 <- MCMCpack::rinvgamma(S, ahn[1]/2, dhn[1]/2)
summary(coda::mcmc(Sig21))
Sig22 <- MCMCpack::rinvgamma(S, ahn[2]/2, dhn[2]/2)
summary(coda::mcmc(Sig22))
Beta1 <- MASS::mvrnorm(S, mu = bhn[,1], Sigma = Bhn[,,1])
summary(coda::mcmc(Beta1))
Beta2 <- MASS::mvrnorm(S, mu = bhn[,2], Sigma = Bhn[,,2])
summary(coda::mcmc(Beta2))
Lambda <- MCMCpack::rdirichlet(S, alpha = an)
summary(coda::mcmc(Lambda))
```

Bibliography

[1] Jim Albert. *Bayesian Computation with R*. Use R! Springer, New York, NY, 2nd edition, 2009.

[2] R. Baath. *Package bayesboot*, 2018.

[3] 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.

[4] 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.

[5] Joshua Chan, Gary Koop, Dale J Poirier, and Justin L Tobias. *Bayesian econometric methods*, volume 7. Cambridge University Press, 2019.

[6] W. Chang. *Web Application Framework for R: Package shiny*. R Studio, 2018.

[7] T. Conley, C. Hansen, and P. Rossi. Plausibly exogenous. *The Review of Economics and Statistics*, 94(1):260–272, 2012.

[8] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2006.

[9] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[10] Andrew Gelman, John B Carlin, Hal S Stern, David Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2021.

[11] J. Geweke. *Bayesian Statistics*, chapter Evaluating the accuracy of sampling-based approaches to calculating posterior moments. Clarendon Press, Oxford, UK., 1992.

[12] John Geweke. Getting it right: Joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, 99(467):799–804, 2004.

[13] John Geweke. *Contemporary Bayesian econometrics and statistics*, volume 537. John Wiley & Sons, 2005.

[14] 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.

[15] Edward Greenberg. *Introduction to Bayesian econometrics*. Cambridge University Press, 2012.

[16] P. Heidelberger and P. D. Welch. Simulation run length control in the presence of an initial transient. *Operations Research*, 31(6):1109–1144, 1983.

[17] H. Jeffreys. *Theory of Probability*. Oxford University Press, London, 1961.

[18] M. Jetter and A. Ramírez Hassan. Want export diversification? Educate the kids first. *Economic Inquiry*, 53(4):1765–1782, 2015.

[19] 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.

[20] Valen E Johnson and David Rossell. On the use of non-local prior densities in bayesian hypothesis tests. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(2):143–170, 2010.

[21] 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.

[22] Gary M Koop. *Bayesian econometrics*. John Wiley & Sons Inc., 2003.

[23] 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.

[24] Tony Lancaster. *An introduction to modern Bayesian econometrics*. Blackwell Oxford, 2004.

[25] Alex Lenkoski, Anna Karl, and Andreas Neudecker. *Package ivbma*, 2013.

[26] Dennis V Lindley. A statistical paradox. *Biometrika*, 44(1/2):187–192, 1957.

[27] Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. *Package MCMCpack*, 2018.

[28] P. Müller, F. Quintana, A. Jara, and T. Hanson. *Bayesian Nonparametric Data Analysis*. Springer, New York, 2015.

[29] Martyn Plummer, Nicky Best, Kate Cowles, Karen Vines, Deepayan Sarkar, Douglas Bates, Russell Almond, and Arni Magnusson. *Output Analysis and Diagnostics for MCMC*, 2016.

[30] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[31] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023.

[32] Adrian Raftery, Jennifer Hoeting, Chris Volinsky, Ian Painter, and Ka Yee Yeung. *Package BMA*, 2012.

[33] 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.

[34] Andrés Ramírez-Hassan. Dynamic variable selection in dynamic logistic regression: an application to internet subscription. *Empirical Economics*, 59(2):909–932, 2020.

[35] Andrés Ramírez-Hassan and Daniela A Carvajal-Rendón. Specification uncertainty in modeling internet adoption: A developing city case analysis. *Utilities Policy*, 70:101218, 2021.

[36] 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.

[37] A. Ramírez-Hassan and M. Graciano-Londoño. A guided tour of Bayesian regression. Technical report, Universidad EAFIT, 2020.

[38] P. Rossi. *Package bayesm*, 2017.

[39] Peter E Rossi, Greg M Allenby, and Rob McCulloch. *Bayesian statistics and marketing*. John Wiley & Sons, 2012.

[40] Peter E. Rossi, Greg M. Allenby, and Robert McCulloch. *Bayesian Statistics and Marketing*. John Wiley and Sons, Hoboken, NJ, 2005.

[41] Francesco Serafini. Multinomial logit models with inla. *R-INLA Tutorial. https://inla.r-inla-download.org/r-inla.org/doc/vignettes/multinomial.pdf*, 2019.

[42] 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.

[43] A. F. M. Smith. A General Bayesian Linear Model. *Journal of the Royal Statistical Society. Series B (Methodological).*, 35(1):67–75, 1973.

[44] Stan Development Team. shinystan: Interactive visual and numerical diagnostics and posterior analysis for Bayesian models., 2017. R package version 2.3.0.

[45] Samuel Thomas and Wanzhu Tu. Learning hamiltonian monte carlo in r. *The American Statistician*, 75(4):403–413, 2021.

[46] Regina Tüchler. Bayesian variable selection for logistic models using auxiliary mixture sampling. *Journal of Computational and Graphical Statistics*, 17(1):76–94, 2008.

[47] P. Woodward. BugsXLA: Bayes for the common man. *Journal of Statistical Software*, 14(5):1–18, 2005.

[48] Jeffrey M. Wooldridge. *Introductory Econometrics: A Modern Approach.* Cengage Learning, Mason, Ohio: South-Western, fifth edition, 2012.

[49] A. Zellner. *Introduction to Bayesian inference in econometrics.* John Wiley & Sons Inc., 1996.

A

Appendix

TABLE A.1
Libraries and commands in BEsmarter GUI.

Univariate models			
Model	**Library**	**Command**	**Reference**
Normal	MCMCpack	MCMCregress	[27]
Logit	MCMCpack	MCMClogit	[27]
Probit	bayesm	rbprobitGibbs	[38]
Multinomial(Mixed) Probit	bayesm	rmnpGibbs	[38]
Multinomial(Mixed) Logit	bayesm	rmnlIndepMetrop	[38]
Ordered Probit	bayesm	rordprobitGibbs	[38]
Negative Binomial(Poisson)	bayesm	rnegbinRw	[38]
Tobit	MCMCpack	MCMCtobit	[27]
Quantile	MCMCpack	MCMCquantreg	[27]
Bayesian bootstrap	bayesboot	bayesboot	[2]
Multivariate models			
Model	**Library**	**Command**	**Reference**
Multivariate	bayesm	rmultireg	[38]
Seemingly Unrelated Regression	bayesm	rsurGibbs	[38]
Instrumental Variable	bayesm	rivGibbs	[38]
Bivariate Probit	bayesm	rmvpGibbs	[38]
Time series models			
Model	**Library**	**Command**	**Reference**
Normal dynamic linear model	dlm	dlmGibbsDIGs	[?]
ARMA	bayesforecast	stan_sarima	[?]
Stochastic volatility models	stochvol	svsample	[?]
VAR	bvartools	draw_posterior	[?]
Hierarchical longitudinal models			
Model	**Library**	**Command**	**Reference**
Normal	MCMCpack	MCMChregress	[27]
Logit	MCMCpack	MCMChlogit	[27]
Poisson	MCMCpack	MCMChpoisson	[27]
Bayesian model averaging			
Model	**Library**	**Command**	**Reference**
Normal (BIC)	BMA	bicreg	[32]
Normal (MC^3)	BMA	MC3.REG	[32]
Normal (instrumental variables)	ivbma	ivbma	[25]
Normal (Dynamic BMA)	dma	dma	[?]
Logit (BIC)	BMA	bic.glm	[32]
Gamma (BIC)	BMA	bic.glm	[32]
Poisson (BIC)	BMA	bic.glm	[32]
Diagnostics			
Diagnostic	**Library**	**Command**	**Reference**
Trace plot	coda	traceplot	[29]
Autocorrelation plot	coda	autocorr.plot	[29]
Geweke test	coda	geweke.diag	[29]
Raftery & Lewis test	coda	raftery.diag	[29]
Heidelberger & Welch test	coda	heidel.diag	[29]

TABLE A.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	110SimBootstrapmodel.csv	110SimBootstrapmodel.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
Time series models		
Model	**Data set file**	**Data set simulation**
Dynamic Linear Models	31SimDynamicLinearModel.csv	31SimDynamicLinearModel.R
ARMA	32SimARMAmodels.csv	32SimARMAmodels.R
Stochastic volatility Models	33SimStochasticVolatility.csv	33SimStochasticVolatility.R
VAR	34SimVARmodels.csv	34SimVARmodels.R
Hierarchical longitudinal models		
Model	**Data set file**	**Data set simulation**
Normal	41SimLogitudinalNormal.csv	41SimLogitudinalNormal.R
Logit	42SimLogitudinalLogit.csv	42SimLogitudinalLogit.R
Poisson	43SimLogitudinalPoisson.csv	43SimLogitudinalPoisson.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
Normal (Dynamic BMA)	514SimDynamicBMA.csv 514SimModels.csv	514SimDynamicBMA.R
Logit (BIC)	52SimLogitBMA.csv	52SimLogitBMA.R
Gamma (BIC)	53SimGammaBMA.csv	53SimGammaBMA.R
Poisson (BIC)	53SimPoissonBMA.csv	53SimPoissonBMA.R

TABLE A.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	$y = [\text{Hosp SHI}]'$
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	$\boldsymbol{y} = [\text{Hosp SHI}]^\top$
Time series models		
Model	**Data set file**	**Dependent variable**
Dynamic Linear Models	16INTDEF.csv	Δ(i3)
ARMA	16INTDEF.csv	Δ(i3)
Stochastic volatility models	17ExcRate.csv	USDEUR
VAR	18USAfiscal	$\boldsymbol{y}_t = [\Delta(ttr_t)\ \Delta(gs_t)\ \Delta(gdp_t)]^\top$
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