R Textbook Companion for
Introduction To Time Series And Forecasting
by Peter J. Brockwell, Richard A. Davis[1]

Created by
Ayush Kumar Nayak
B.Tech.
Mechanical Engineering
National Institute of Technology, Rourkela
Cross-Checked by
R TBC Team

February 19, 2025

Book Description

Title: Introduction To Time Series And Forecasting

Author: Peter J. Brockwell, Richard A. Davis

Publisher: Springer-verlag, New York, Usa

Edition: 3

Year: 2016

ISBN: ISBN 0-387-95351-5

R numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means an R code whose theory is explained in Section 2.3 of the book.

Contents

List of R Codes

Chapter 1

Introduction

R code Exa 1.1.1 Australian wine sales

```
1  # Page No. 2
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  wine_data <- read.delim("WINE.TSM", header = FALSE)
5  colnames(wine_data)[1]<- "Sales"
6  ggplot(wine_data, aes(x = seq(as.Date("1980-01-01"),
       as.Date("1991-10-01"), by = "month"), y = Sales)
       ) +
7    geom_point() +
8    geom_line() +
9    labs(title = "Monthly Wine Sales (Jan 1980 - Oct
       1991)",
10         x = "Months",
11         y = "Sales") +
12   theme_minimal()
```

R code Exa 1.1.3 Accidental deaths

```
1  # Page No. 2
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  deaths= read.csv("DEATHS.TSM", header = FALSE)
5  colnames(deaths)[1]<- "deaths"
6  ggplot(deaths, aes(x = seq(as.Date("1973-01-01"), as
       .Date("1978-12-01"), by = "month"), y = deaths))
        +
7    geom_point(shape = 15, size = 1) +
8    geom_line() +
9    labs(title = "Deaths (Jan 1973 - Nov 1978)",
10         x = "Months",
11         y = "Deaths") +
12   theme_minimal()
```

R code Exa 1.1.4 Signal Detection Problem

```
1  # Page No. 3
2  set.seed(123)
3  t <- 1:200
4  N <- rnorm(200, mean = 0, sd = 0.5)
5  X <- cos(t/10)
6  plot(t, X, type = "l", col = "blue", xlab = "t",
       ylab = "X", main = "Signal plot",lwd=2)
7  points(t, N, pch = 16, col = "black", bg = "black",
       cex = 0.5)
```

R code Exa 1.1.5 Population of the USA

```
1  # Page No. 4
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
```

```
 3 library(ggplot2)
 4 uspop= read.csv("USPOP.TSM")
 5 names(uspop)[names(uspop) == "X3929214"] <- "
     population"
 6 start_year=1790
 7 num_repeated=20
 8 interval=10
 9 ggplot(uspop, aes(x=seq_len(num_repeated) * interval
     + start_year, y = population)) +
10   geom_point() +
11   geom_line() +
12   labs(title = "Population",
13        x = "Years",
14        y = "US population") +
15   theme_minimal()
```

R code Exa 1.1.6 Strikes in USA

```
 1 # Page No. 4
 2 # Downloading link: https://storage.googleapis.com/
     springer-extras/zip/2002/978-0-387-21657-7.zip
 3 library(ggplot2)
 4 strike <- read.delim("STRIKES.TSM", header = FALSE)
 5 colnames(strike)[1] <- "Strikes"
 6 start_year=1951
 7 end_year=1980
 8 ggplot(strike, aes(x=seq(start_year,end_year), y =
     Strikes)) +
 9   geom_point() +
10   geom_line() +
11   labs(title = "Strikes in US",
12        x = "Years",
13        y = "Strikes") +
14   theme_minimal()
```

R code Exa 1.3.3 Random walk

```r
1 # Page no. 7
2 set.seed(123)
3 t <- 200
4 steps <- rnorm(t)
5 random_walk <- cumsum(steps)
6 plot(0:t, c(0, random_walk), type = "l", col = "blue
      ",
7        xlab = "Time", ylab = "Value", main = "Simple
           Random Walk")
8 points(0:t, c(0, random_walk), col = "red", pch = 1)
```

R code Exa 1.3.4 Regression on population data

```r
1 # Page No. 8
2 # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3 library(ggplot2)
4 uspop= read.delim("USPOP.TSM", header = FALSE)
5 colnames(uspop)[1]<- "population"
6 start_year=1790
7 num_repeated=21
8 interval=10
9 uspop$years <- seq_len(num_repeated) * interval+
      start_year
10 fit<-lm(population ~ poly(years,2,raw = TRUE), data
      = uspop)
11 ggplot(uspop, aes(x=years, y=population)) +
12   geom_point() +
13   geom_smooth(method = "lm", formula = y ~poly(x,2,
        raw=TRUE), se = FALSE) +
```

```
14    labs(title = "US Population",
15          x = "Years",
16          y = "Population") +
17    theme_minimal()
```

R code Exa 1.3.5 Level of Lake Huron

```
1  # Page No. 9
2  # Downloading link: https://storage.googleapis.com/
       springer−extras/zip/2002/978−0−387−21657−7.zip
3  library(ggplot2)
4  hudson= read.csv("LAKE.TSM", header = FALSE)
5  colnames(hudson)[1] <- "level"
6  start_year=1875
7  end_year=1972
8  hudson$years <-(seq(start_year,end_year))
9  fit<-lm(level~years,data = hudson)
10 residuals <- resid(fit)
11 residual_df <- data.frame(years = hudson$years,
       residuals = residuals)
12 par(mfrow=c(1,2))
13 # Figure 1−9
14 plot(hudson$years, hudson$level, type = "o",
15        main = "Lake Hudson", xlab = "Years", ylab = "
            Water levels", pch = 19)
16 abline(fit, col = "blue",lw=2)
17 # Figure 1−10
18 plot(residual_df$years,residual_df$residuals, type =
       "o",pch = 19,
19        xlab = "Years", ylab = "Residuals", main = "
            Residuals plot")
20 abline(h = 0, col = "blue", lw = 2)
21 print(coef(fit))
```

R code Exa 1.3.6 Harmonic regression on accidental deaths

```
1  # Page No. 11
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  deaths <- read.csv("DEATHS.TSM", header = FALSE)
5  colnames(deaths)[1] <- "deaths"
6  n <- length(deaths$deaths)
7  time <- 1:n
8  f1 <- n / 12
9  f2 <- n / 6
10 fit <- lm(deaths$deaths ~ sin(2 * pi * time / f1) +
       cos(2 * pi * time / f1) +
11               sin(2 * pi * time / f2) + cos(2 * pi *
                   time / f2))
12 fitted_values <- predict(fit)
13 plot(time, deaths$deaths, type = "p", col = "black",
       pch = 15, xlab = "Time", ylab = "Value",
14       main = "Harmonic Fit")
15 lines(time, fitted_values, col = "blue", lw =2)
```

R code Exa 1.4.6 Random noise

```
1  # Page No. 16
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  set.seed(123)
5  noise <- rnorm(200, mean = 0, sd = 1)
6  df <- data.frame(Index = 1:200, Noise = noise)
7  ggplot(df, aes(x = Index, y = Noise)) +
```

```
 8    geom_point()+
 9    geom_line() +
10    labs(x = "Index", y = "Noise", title = "Simulated
          N(0,1) Noise")+
11    theme_minimal()
12 acf_result <- acf(noise, plot = FALSE)
13 n <- length(noise)
14 bounds <- 1.96 / sqrt(n)
15 acf_df <- data.frame(Lag = acf_result$lag, ACF = acf
      _result$acf)
16 ggplot(acf_df, aes(x = Lag, y = ACF)) +
17    geom_hline(yintercept = c(-bounds, bounds)) +
18    geom_hline(yintercept = 0) +
19    geom_segment(aes(xend = Lag, yend = 0)) +
20    labs(x = "Lag", y = "ACF", title = "Sample
          Autocorrelation Function (ACF)") +
21    ylim(-1, 1)+
22    theme_minimal()
```

R code Exa 1.5.1 Moving average of strikes

```
 1 # Page No. 22
 2 # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
 3 library(ggplot2)
 4 library(zoo)
 5 strike<- read.csv("STRIKES.TSM", header =FALSE)
 6 colnames(strike)[1] <- "Strikes"
 7 start_year=1951
 8 end_year=1980
 9 window_size <- 5
10 strike$Moving_Avg <- rollmean(strike$Strikes, k =
      window_size, fill = NA)
11 strike$residuals <- strike$Strikes-strike$Moving_Avg
12 # Figure 1-18
```

```
13  ggplot()+
14    geom_line(data=strike, aes(x = seq(start_year,end_
         year),y=Moving_Avg))+
15    geom_point(data=strike, aes(x = seq(start_year,end
         _year),y=strike$Strikes))+
16    labs(x = "Year", y = "Strikes", title = "Strikes
         Data with Moving Average")+
17    theme_minimal()
18  # Figure 1-19
19  ggplot(data=strike, aes(x = seq(start_year,end_year)
       ,y=residuals))+
20    geom_line()+
21    geom_point()+
22    labs(x = "Year", y = "Strikes", title = "Strikes
         Data residuals")+
23    theme_minimal()
```

R code Exa 1.5.2 Smooth exponential and low pass filter

```
1  # Page No. 24
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  strike<- read.csv("STRIKES.TSM", header = FALSE)
5  colnames(strike)[1] <- "Strikes"
6  # Figure 1-21
7  plot(smooth.exp(ts(strike$Strikes),0.4))
8  lines(smooth.exp(ts(strike$Strikes),0.4))
9  # Figure 1-22
10 plot(smooth.fft(ts(strike$Strikes),0.4))
11 lines(smooth.fft(ts(strike$Strikes),0.4))
```

R code Exa 1.5.3 Differenced series

```
1  # Page No. 11
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  library(pracma)
5  library(dplyr)
6  uspop= read.delim("USPOP.TSM", header = FALSE)
7  colnames(uspop)[1] <- "population"
8  start_year=1790
9  num_repeated=21
10 interval=10
11 uspop$years <- seq_len(num_repeated) * interval+
       start_year
12 diff2 <- diff(diff(uspop$population))
13 uspop <- slice(uspop,-(1:2))
14 uspop$diff2 <- diff2
15 ggplot(uspop, aes(x = years, y = diff2)) +
16   geom_point()+
17   geom_line() +
18   labs(title = "Second-Order Differences of
        Population Data",
19       x = "Years", y = "Second-Order Differences")+
20   theme_minimal()
```

R code Exa 1.5.4 Deseasonalization and seasonal component

```
1  # Page No. 28
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  library(pracma)
5  deaths<-read.delim("DEATHS.TSM", header =FALSE)
6  deaths$years<- seq(as.Date("1973-01-01"), as.Date("
       1978-12-01"), by = "month")
7  period <- 12
```

```
8  colnames(deaths)[1] <- "deaths"
9  decomposition <- decompose(ts(deaths$deaths,
       frequency = period))
10 seasonal_component <- decomposition$seasonal
11 deseasonalized_data <- deaths$deaths - seasonal_
       component
12 deseasonalized_df <- data.frame(years = deaths$years
       , deseasonalized_deaths = deseasonalized_data)
13 seasonal_component_df <- data.frame(years = deaths$
       years, seasonal_component = seasonal_component)
14 # Figure 1-24
15 ggplot(deseasonalized_df, aes(x = years, y =
       deseasonalized_deaths)) +
16   geom_line(color = "blue") +
17   geom_point()+
18   labs(x = "Years", y = "Deseasonalized Deaths",
         title = "Deseasonalized Deaths") +
19   theme_minimal()
20 # Figure 1-25
21 ggplot(seasonal_component_df, aes(x = years, y =
       seasonal_component)) +
22   geom_line(color = "red") +
23   geom_point()+
24   labs(x = "Years", y = "Seasonal Component", title
         = "Seasonal Component") +
25   theme_minimal()
```

R code Exa 1.5.5 Estimation of seasonal component

```
1 # Page No. 28
2 # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3 library(ggplot2)
4 library(dplyr)
5 deaths= read.delim("DEATHS.TSM", header = FALSE)
```

```
 6  colnames (deaths)[1] <- "deaths"
 7  deaths$months=seq(as.Date("1973-01-01"), as.Date("
       1978-12-01"),by='month')
 8  diff1 <- diff(deaths$deaths, lag = 12)
 9  deaths <- slice(deaths,-(1:12))
10  deaths$diff1 <- diff1
11  # Figure 1-26
12  ggplot(deaths, aes(x = months, y = diff1)) +
13    geom_point()+
14    geom_line() +
15    labs(title = "First-Order Differences of deaths
         Data",
16        x = "months", y = "First-Order Differences")+
17    theme_minimal()
18  # Figure 1-27
19  diff2 <- diff(deaths$diff1)
20  deaths <- slice(deaths,-1)
21  deaths$diff2 <- diff2
22  ggplot(deaths, aes(x = months, y = diff2)) +
23    geom_point()+
24    geom_line() +
25    labs(title = "Second-Order Differences of deaths
         Data",
26        x = "months", y = "Second-Order Differences")
                +
27    theme_minimal()
```

R code Exa 1.6.1 ACF on signal data

```
1  # Page No. 33
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  signal<- read.delim("SIGNAL.TSM", header = FALSE)
4  colnames(signal)[1] <- "signals"
5  acf_values <- acf(signal$signals, plot = FALSE)$acf
```

```r
6  n <- length(signal$signals)
7  conf_bound <- 1.96 / sqrt(n)
8  plot(acf_values, ylim = c(-conf_bound, conf_bound),
9        main = "Sample Autocorrelation Function (ACF)",
10       ylab = "ACF", xlab = "Lag", type = "h")
11 abline(h = c(-conf_bound, conf_bound), col = "red",
      lty = 2)
12 abline(h = 0, lty = 2)
```

Chapter 2

Stationary Processes

R code Exa 2.4.3 MA1 Process

```r
# Page No. 53
n <- 200
set.seed(123)
Z <- rnorm(n)
X <- numeric(n)
X[1] <- Z[1]
for (i in 2:n) {
  X[i] <- Z[i] - 0.8 * Z[i-1]
}
acf_values <- acf(X, plot = FALSE)$acf
plot(0:40, acf_values[1:41], type = "h", ylim = c(-1, 1),
      xlab = "Lag", ylab = "ACF", main = "Sample
          Autocorrelation Function for MA(1)")
abline(h = c(-1.96/sqrt(n), 1.96/sqrt(n)), col = "red", lty = 2)
abline(h = 0, col = "blue", lty = 1)
```

R code Exa 2.4.4 AR1 Process

```
1  #..
2  # Page No. 54
3  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
4  hudson= read.csv("LAKE.TSM")
5  names(hudson)[names(hudson) == "X10.38"] <- "level"
6  start_year=1876
7  end_year=1972
8  hudson$years <- seq(start_year,end_year)
9  fit<-lm(level~years,data = hudson)
10 residuals <- resid(fit)
11 residuals_df <- data.frame(years = hudson$years,
       residuals = residuals)
12 n <- nrow(residuals_df)
13 phi <- 0.791
14 model_acf <- function(i) {
15   phi^i
16 }
17 confidence_bounds <- function(i) {
18   1.96 * (n^(-0.5)) * sqrt(((1 - (phi^(2*i))) * (1 +
         (phi^2))) / (1 - (phi^2)))
19 }
20 acf_values <- acf(residuals_df$residuals, plot =
       FALSE)$acf
21 upper_conf_bounds <- sapply(1:40, function(i) {
22   confidence_bounds(i) + (phi^i)
23 })
24 lower_conf_bounds <- sapply(1:40, function(i) {
25   (phi^i) - confidence_bounds(i)
26 })
27 plot(0:40, acf_values[1:41], type = "h", ylim = c
       (-1, 1),
28      xlab = "Lag", ylab = "ACF", main = "Sample
           Autocorrelation Function of Residuals (AR(1)
           )")
29 lines(1:40, upper_conf_bounds, col = "red", lty = 2)
30 lines(1:40, lower_conf_bounds, col = "red", lty = 2)
31
```

```
32  # Plot the model ACF
33  points(1:40, sapply(1:40, model_acf), type = "b",
        col = "blue")
```

R code Exa 2.5.5 Durbin Levinson and innovations algorithm

```
1   # Page no. 64
2   compute_autocovariance <- function(phi) {
3     gamma_0 <- 1 + phi^2
4     gamma_1 <- -phi
5     return(list(gamma_0 = gamma_0, gamma_1 = gamma_1))
6   }
7   innovation_algorithm <- function(gamma) {
8     theta_11 <- -gamma$gamma_1 / gamma$gamma_0
9     return(list(theta_11 = theta_11))
10  }
11  durbin_levinson_algorithm <- function(gamma) {
12    phi_11 <- gamma$gamma_1 / gamma$gamma_0
13    sigma_1_squared <- gamma$gamma_0 * (1 - phi_11^2)
14    return(list(phi_11 = phi_11, sigma_1_squared =
        sigma_1_squared))
15  }
16  phi <- 0.9
17  gamma <- compute_autocovariance(phi)
18  theta <- innovation_algorithm(gamma)
19  phi_result <- durbin_levinson_algorithm(gamma)
20  cat(paste0("theta_11 = ", theta$theta_11, "\n"))
21  cat(paste0("phi_11 = ", phi_result$phi_11, "\n"))
```

Chapter 3

ARMA Models

R code Exa 3.1.1 ARMA 1 1

```
1  # Page no.76
2  ar_params <- c(0.5)
3  ma_params <- c(0.4)
4  is_invertible <- function(ma_params) {
5    roots <- polyroot(c(1, ma_params))
6    all(abs(roots) > 1)
7  }
8
9  invertibility_status <- is_invertible(ma_params)
10 invertibility_status
```

R code Exa 3.1.2 AR2 Process

```
1  # Page no.76
2  # Coefficients of AR(2) model
3  phi1 <- 0.7
4  phi2 <- -0.1
5  poly_coefs <- c(1, -phi1, -phi2)
```

```
6 roots <- polyroot(poly_coefs)
7 cat("Roots of the characteristic polynomial (zeros
     of the AR(2) process):\n")
8 cat(roots, "\n")
```

R code Exa 3.1.3 ARMA 2 1

```
1 # Page no. 77
2 ar_params <- c(-0.75, 0.5625)
3 ma_params <- c(1.25)
4 is_invertible <- function(ma_params) {
5   roots <- polyroot(c(1, ma_params))
6   all(abs(roots) > 1)
7 }
8 invertibility_status <- is_invertible(ma_params)
9 invertibility_status
```

R code Exa 3.2.4 General AR2 process

```
1 # Page No. 80
2 # Figure 3-1
3 library(stats)
4 xi1 <- 2
5 xi2 <- 5
6 phi1 <- 1/xi1 + 1/xi2
7 phi2 <- -(1/xi1) * (1/xi2)
8 set.seed(123)
9 n <- 1000
10 ar_process <- arima.sim(model = list(ar = c(phi1,
     phi2)), n = n)
11 acf(ar_process, main = "Sample ACF of AR(2) Process"
     )
12 # Figure 3-2
```

23

```
13  xi1 <- 10/9
14  xi2 <- 2
15  phi1 <- 1/xi1 + 1/xi2
16  phi2 <- -(1/xi1) * (1/xi2)
17  ar_process <- arima.sim(model = list(ar = c(phi1,
       phi2)), n = n)
18  acf(ar_process, main = "Sample ACF of AR(2) Process"
       )
19  # Figure 3-3
20  xi1 <- -10/9
21  xi2 <- 2
22  phi1 <- 1/xi1 + 1/xi2
23  phi2 <- -(1/xi1) * (1/xi2)
24  ar_process <- arima.sim(model = list(ar = c(phi1,
       phi2)), n = n)
25  acf(ar_process, main = "Sample ACF of AR(2) Process"
       )
26
27  # Figure 3-4
28  xi1 <- complex(real = 2/3, imaginary = 2*sqrt(3)/3)
29  xi2 <- complex(real = 2/3, imaginary = -2*sqrt(3)/3)
30  phi1 <- Re(1/xi1 + 1/xi2)
31  phi2 <- Re(-(1/xi1) * (1/xi2))
32  ar_process <- arima.sim(model = list(ar = c(phi1,
       phi2)), n = n)
33  acf(ar_process, main = "Sample ACF of AR(2) Process"
       )
```

R code Exa 3.2.8 Overshorts series

```
1  # Page No. 84
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  oshorts<- read.csv("OSHORTS.TSM", header =FALSE)
4  colnames(oshorts)[1] <- "overshorts"
```

```
5  oshorts$days <- seq(1,nrow(oshorts))
6  # Figure 3-5
7  plot(oshorts$days,oshorts$overshorts, xlab = "Days",
       ylab = "Overshorts",
8       type = 'o', col = "blue")
9  abline(h=0)
10 # Figure 3-6
11 acf_result <- acf(oshorts$overshorts, plot = FALSE)
12 n <- length(oshorts)
13 bounds <- 1.96 * ((1 + 2 * acf_result$acf[2]^2)^(1/
       2)) / sqrt(n)
14 plot(acf_result, main = "Sample ACF with Bounds")
15 print(mean(oshorts$overshorts))
16 acvf<-acf(oshorts$overshorts, plot= FALSE, type = '
       covariance')
17 print(acvf$acf[1])
18 print(acvf$acf[2])
```

R code Exa 3.2.9 The sunspot numbers

```
1  # Page No. 86
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  spots<- read.csv("SUNSPOTS.TSM",header = FALSE)
5  colnames(spots)[1] <- "sunspots"
6  pacf_result <- pacf(spots, plot = FALSE)
7  bounds <- 1.96 / sqrt(100)
8  plot(pacf_result, main = "Sample PACF")
9  print(pacf_result)
10 acvf<-acf(spots$sunspots, plot= FALSE, type = '
       covariance')
11 print(acvf$acf[1])
12 print(acvf$acf[2])
13 print(acvf$acf[3])
```

R code Exa 3.3.4 Numerical prediction of ARMA 2 3

```r
1  # Page no. 90
2  # Answer may vary due to randomization in simulation
3  library(forecast)
4  ar_params <- c(1,-0.24)
5  ma_params <- c(0.4, 0.2, 0.1)
6  set.seed(46)
7  n <- 10
8  arma_process <- arima.sim(model = list(ar = ar_
      params, ma = ma_params), n = n)
9  print(arma_process)
10 acf_values <- acf(arma_process, type="covariance",
      plot=FALSE)$acf
11 gamma_0 <- acf_values[1]
12 gamma_1 <- acf_values[2]
13 gamma_2 <- acf_values[3]
14 cat("gamma_0 =", gamma_0, "\n")
15 cat("gamma_1 =", gamma_1, "\n")
16 cat("gamma_2 =", gamma_2, "\n")
17 innovations_algorithm <- function(arma_process, n_
      steps) {
18   n <- length(arma_process)
19   predictions <- numeric(n_steps)
20   e <- numeric(n + n_steps)
21   phi <- numeric(n + n_steps)
22   theta <- numeric(n + n_steps)
23   for (i in 1:n_steps) {
24     predictions[i] <- sum(ar_params * arma_process[(
          n-i+1):(n-i+2)])
25     + sum(ma_params * e[(n-i+1):(n-i+3)])
26     e[n+i] <- arma_process[i] - predictions[i]
27   }
28   return(predictions)
```

```
29 }
30 predictions <- innovations_algorithm(arma_process,
      10)
31 print(predictions)
```

R code Exa 3.3.5 h step prediction of ARMA

```
1 # Page no. 91
2 # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3 library(forecast)
4 ar_params <- c(1,-0.24)
5 ma_params <- c(0.4, 0.2, 0.1)
6 E334 <- read.delim("E334.TSM", header = FALSE)
7 colnames(E334)[1] <- "E"
8 Ets <- ts(E334$E)
9 arma_model <- Arima(Ets, order=c(2, 0, 3))
10 forecasts <- forecast(arma_model, h=10)
11 cat("\nForecasted values for the next 10 steps:\n")
12 print(forecasts$fitted)
```

Chapter 4

Spectral Analysis

R code Exa 4.1.2 Linear combination of sinusoids

```r
1  # Page  no.  101
2  # Answer  may  vary  due  to  randomization
3  library(ggplot2)
4  k <- 2
5  omega <- seq(pi/4, pi/6, length.out = k)
6  sigma2 <- 9
7  t <- 1:100
8  set.seed(123)
9  A <- rnorm(k, mean = 0, sd = sqrt(sigma2))
10 B <- rnorm(k, mean = 0, sd = sqrt(sigma2))
11 X_t <- sapply(t, function(ti) {
12   sum(A * cos(omega * ti) + B * sin(omega * ti))
13 })
14 df <- data.frame(Time = t, Value = X_t)
15 ggplot(df, aes(x = Time, y = Value)) +
16   geom_line() +
17   geom_point()+
18   ggtitle("Sample  Path") +
19   xlab("Time") +
20   ylab("X(t)") +
21   theme_minimal()
```

```
22  F_lambda <- function(lambda, omega, sigma2) {
23    sapply(lambda, function(l) {
24      sum(sigma2 * (0.5 * (l >= -omega & l < omega) +
         1.0 * (l >= omega)))
25    })
26  }
27  lambda <- seq(-pi, pi, length.out = 1000)
28  F_values <- F_lambda(lambda, omega, sigma2)
29  df_F <- data.frame(Lambda = lambda, F_Lambda = F_
       values)
30  ggplot(df_F, aes(x = Lambda, y = F_Lambda)) +
31    geom_step() +
32    ggtitle("Spectral Distribution Function F(  )") +
33    xlab("  ") +
34    ylab("F(  )") +
35    theme_minimal()
```

R code Exa 4.1.4 Spectral density of AR 1

```
1   # Page no. 103
2   library(ggplot2)
3   library(stats)
4   set.seed(123)
5   n <- 1000
6   # Figure 4-3
7   phi <- 0.7
8   sigma2 <- 1
9   density <- function(lambda, phi, sigma2) {
10    1 / (2 * pi) * sigma2 / (1 + phi^2 - 2 * phi * cos
        (lambda))
11  }
12  lambda <- seq(0, pi, length.out = 1000)
13  values <- density(lambda, phi, sigma2)
14  df_spectral <- data.frame(Lambda = lambda,
       SpectralDensity = values)
```

```
15  ggplot(df_spectral, aes(x = Lambda, y =
       SpectralDensity)) +
16    geom_line() +
17    ggtitle("Spectral Density") +
18    xlab("  ") +
19    ylab("Spectral Density") +
20    theme_minimal()
21  # Figure 4−4
22  phi <- -0.7
23  sigma2 <- 1
24  density <- function(lambda, phi, sigma2) {
25    1 / (2 * pi) * sigma2 / (1 + phi^2 - 2 * phi * cos
         (lambda))
26  }
27  lambda <- seq(0, pi, length.out = 1000)
28  values <- density(lambda, phi, sigma2)
29  df_spectral <- data.frame(Lambda = lambda,
       SpectralDensity = values)
30  ggplot(df_spectral, aes(x = Lambda, y =
       SpectralDensity)) +
31    geom_line() +
32    ggtitle("Spectral Density") +
33    xlab("  ") +
34    ylab("Spectral Density") +
35    theme_minimal()
36  # Figure 4−5
37  phi <- 0.7
38  ar_process <- arima.sim(model = list(ar = c(phi)), n
       = n)
39  acf(ar_process, main = "ACF of AR(1) Process")
40  # Figure 4−6
41  phi <- -0.7
42  ar_process <- arima.sim(model = list(ar = c(phi)), n
       = n)
43  acf(ar_process, main = "ACF of AR(1) Process")
```

R code Exa 4.1.5 Spectral density of MA 1

```r
# Page no. 105
library(ggplot2)
theta <- 0.9
sigma2 <- 1
density <- function(lambda, theta, sigma2) {
  sigma2 / (2 * pi) * (1 + theta^2 + 2 * theta * cos
      (lambda))
}
lambda <- seq(0, pi, length.out = 1000)
values <- density(lambda, theta, sigma2)
df_spectral <- data.frame(Lambda = lambda,
    SpectralDensity = values)
# Figure 4-7
ggplot(df_spectral, aes(x = Lambda, y =
    SpectralDensity)) +
  geom_line() +
  ggtitle("Spectral Density of MA(1) Process") +
  xlab(expression(lambda)) +
  ylab(expression(f(lambda))) +
  theme_minimal()
# Figure 4-8
theta <- -0.9
sigma2 <- 1
density <- function(lambda, theta, sigma2) {
  sigma2 / (2 * pi) * (1 + theta^2 + 2 * theta * cos
      (lambda))
}
lambda <- seq(0, pi, length.out = 1000)
values <- density(lambda, theta, sigma2)
df_spectral <- data.frame(Lambda = lambda,
    SpectralDensity = values)
ggplot(df_spectral, aes(x = Lambda, y =
```

```
      SpectralDensity)) +
28    geom_line() +
29    ggtitle("Spectral Density of MA(1) Process") +
30    xlab(expression(lambda)) +
31    ylab(expression(f(lambda))) +
32    theme_minimal()
```

R code Exa 4.2.2 Sunspot numbers spectral density

```
1  # Page No. 110
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  library(TSA)
5  library(stats)
6  library(itsmr)
7  spots= read.csv("SUNSPOTS.TSM", header =FALSE)
8  colnames(spots)[1]<- "sunspots"
9  periodogram <- spec.pgram(spots, log = "no", plot =
       FALSE)
10 freq <- periodogram$freq
11 spec <- periodogram$spec
12 weights <- rep(1/3, 3)
13 freq <- freq * (2 * pi)
14 smoothed_spec <- stats::filter(spec, filter=weights,
       sides=2)
15 # Figure 4-9
16 p <- periodogram(ts(spots$sunspots), q = 1, opt = 0)
17 plot(p$freq,(p$spec)/(2*pi), type = "o-", pch=19,
       xlab = "frequency", ylab = "spectral density")
18 # Figure 4-10
19 df <- data.frame(freq = freq, smoothed_spec =
       smoothed_spec)
20 ggplot(df, aes(x = freq, y = smoothed_spec)) +
21   geom_line() +
```

```
22    scale_x_continuous(limits = c(0, pi)) +
23    labs(
24      x = expression(lambda),
25      y = expression(hat(f)(lambda)),
26      title = "Spectral Density Estimate"
27    ) +
28    theme_minimal()
29  # Figure 4-11
30  weights <- c(1/15, 2/15, 3/15, 3/15, 3/15, 2/15, 1/
        15)
31  smoothed_spec <- stats::filter(spec, filter=weights,
        sides=2)
32  df <- data.frame(freq = freq, smoothed_spec =
        smoothed_spec)
33  ggplot(df, aes(x = freq, y = smoothed_spec)) +
34    geom_line() +
35    scale_x_continuous(limits = c(0, pi)) +
36    labs(
37      x = expression(lambda),
38      y = expression(hat(f)(lambda)),
39      title = "Spectral Density Estimate"
40    ) +
41    theme_minimal()
```

R code Exa 4.4.1 Spectral density of AR 2

```
1  # Page 112
2  library(ggplot2)
3  D_q <- function(lambda, q) {
4    if (lambda == 0) {
5      return(1)
6    } else {
7      return(sin((q + 0.5) * lambda) / ((2 * q + 1) *
          sin(lambda / 2)))
8    }
```

```r
 9  }
10  q <- 10
11  lambda <- seq(0, pi, length.out = 1000)
12  D_10 <- sapply(lambda, D_q, q = q)
13  df <- data.frame(lambda = lambda, D_10 = D_10)
14  ggplot(df, aes(x = lambda, y = D_10)) +
15    geom_line() +
16    labs(
17      x = expression(lambda),
18      y = expression(D[10](lambda)),
19      title = "Transfer Function D[10](lambda) for
            Simple Moving-Average Filter"
20    ) +
21    theme_minimal()
22  # Figure 4-13
23  ideal_low_pass <- function(lambda, wc) {
24    ifelse(abs(lambda) <= wc, 1, 0)
25  }
26  wc <- pi / 4
27  q_values <- c(2, 10)
28  ideal_values <- ideal_low_pass(lambda, wc)
29  D_2_values <- sapply(lambda, D_q, q = 2)
30  D_10_values <- sapply(lambda, D_q, q = 10)
31  df <- data.frame(
32    lambda = rep(lambda, 3),
33    value = c(ideal_values, D_2_values, D_10_values),
34    type = factor(rep(c("Ideal", "q = 2", "q = 10"),
        each = length(lambda)))
35  )
36  ggplot(df, aes(x = lambda, y = value, color = type))
      +
37    geom_line() +
38    labs(
39      x = expression(lambda),
40      y = "Transfer Function",
41      title = "Transfer Functions: Ideal Low-Pass
            Filter and Truncated Fourier Approximations"
42    ) +
```

```
43    scale_color_manual(values = c("Ideal" = "black", "
          q = 2" = "blue", "q = 10" = "red")) +
44    theme_minimal() +
45    theme(legend.title = element_blank())
```

Chapter 5

Modeling and Forecasting with ARMA Processes

R code Exa 5.1.1 The Dow Jones Utilities Index

```
1  # Page No. 126
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(forecast)
4  library(tseries)
5  dow<- read.csv("DOWJ.TSM", header = FALSE)
6  colnames(dow)[1]<- "jones"
7  dowjones <- ts(dow$jones)
8  dowjones_diff <- diff(dowjones, lag = 1)
9  ar_model <- ar(dowjones_diff, order.max = 1, method
      = "yule-walker")
10 sample_autocovariance <- acf(dowjones_diff, plot =
      FALSE, type = 'covariance')
11 ar_coefficient <- ar_model$ar
12 par(mfrow = c(1, 2))
13 acf(dowjones_diff, main = "ACF of Differenced Series
      ")
14 pacf(dowjones_diff, main = "PACF of Differenced
      Series")
```

```
15 print(sample_autocovariance)
16 print(ar_coefficient)
```

R code Exa 5.1.2 MA 1 model forecasting

```
1  # Page No. 128
2  library(forecast)
3  library(tseries)
4  oshorts<- read.csv("OSHORTS.TSM", header = FALSE)
5  colnames(oshorts)[1]<- "overshorts"
6  ots <- ts(oshorts$overshorts)
7  rho_1 <- acf(ots, plot=FALSE)$acf[2]
8  gamma <- acf(ots, plot = FALSE, type = 'covariance')
       $acf[1]
9
10 if (abs(rho_1) > 0.5) {
11    theta_hat <- rho_1/abs(rho_1)
12 } else {
13    theta_hat <- (rho_1) * sqrt(4 * rho_1^2 - 4 * rho_
         1) / (2 * abs(rho_1))
14 }
15 sigma2_hat <- gamma / (1 + theta_hat^2)
16 cat("Estimated theta_hat:", theta_hat, "\n")
17 cat("Estimated sigma2_hat:", sigma2_hat, "\n")
```

R code Exa 5.1.3 Dow jones utilities index using burg model

```
1  # Page No. 131
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(tseries)
4  library(itsmr)
5  dow<- read.csv("DOWJ.TSM", header = FALSE)
```

```
 6  colnames(dow)[1] <- "jones"
 7  time_series <- ts(dow$jones)
 8  Y_t <- diff(time_series, lag=1)
 9  ar_order <- 1
10  burg_model <- burg(Y_t, ar_order)
11  ar_param <- burg_model$phi
12  stderror <- (burg_model$se.phi)
13  aicc <- burg_model$aicc
14  cat("AR(1) model parameter:", ar_param, "\n")
15  cat("AICC:", aicc, "\n")
16  find_conf <- function(param, stderr){
17    low <- param - (stderr*1.96)
18    high <- param + (stderr*1.96)
19    x <- c(low, high)
20    return (x)
21  }
22  confs <- find_conf(ar_param,stderror)
23  cat("95% Confidence Bounds: ",confs)
```

R code Exa 5.1.4 Modeling on Lake data

```
 1  # Page No. 131
 2  # Downloading link: https://storage.googleapis.com/
        springer-extras/zip/2002/978-0-387-21657-7.zip
 3  library(tseries)
 4  library(itsmr)
 5  huron<- read.csv("LAKE.TSM", header=FALSE)
 6  colnames(huron)[1] <- 'water'
 7  time_series <- ts(huron$water)
 8  Y_t <- time_series
 9  X_t <- Y_t - 9.0041
10  par(mfrow = c(1, 2))
11  # Figure 5-3
12  acf(X_t, main = "ACF")
13  # Figure 5-4
```

```
14  pacf(X_t, main = "PACF")
15  ar_order <-2
16
17  # Burg model
18  burg_model <- burg(X_t, ar_order)
19  arb_param <- burg_model$phi
20  stderr <- (burg_model$se.phi)
21  aicc <- burg_model$aicc
22  conf_lower <- arb_param - (stderr*1.96)
23  conf_upper <- arb_param + (stderr*1.96)
24  print(" For burg model: ")
25  cat("AR(1) model parameter:", arb_param, "\n")
26  cat("AICC:", aicc, "\n")
27  cat("95% Confidence Bounds: (", conf_lower, ", ",
       conf_upper, ")\n")
28
29  # Yule walker model
30  yw_model <- yw(X_t, ar_order)
31  ary_param <- yw_model$phi
32  stderr <- (yw_model$se.phi)
33  aicc <- yw_model$aicc
34  conf_lower <- ary_param - (stderr*1.96)
35  conf_upper <- ary_param + (stderr*1.96)
36  print(" For yule walker model: ")
37  cat("AR(1) model parameter:", ary_param, "\n")
38  cat("AICC:", aicc, "\n")
39  cat("95% Confidence Bounds: (", conf_lower, ", ",
       conf_upper, ")\n")
```

R code Exa 5.1.5 Estimations on Dow jones utilities index

```
1  # Page No.  134
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(tseries)
```

```
4  library(itsmr)
5  dow<- read.csv("DOWJ.TSM", header = FALSE)
6  colnames(dow)[1] <- "jones"
7  time_series <- ts(dow$jones)
8  Y_t <- diff(time_series, lag=1)
9  ma_order <- 2
10 inno_model <- ia(Y_t, ma_order, m = 17)
11 ma_param <- inno_model$theta
12 stderr <- (inno_model$se.theta)
13 aicc <- inno_model$aicc
14 stddev_1 <- ma_param[1]/(1.96*stderr[1])
15 stddev_2 <- ma_param[2]/(1.96*stderr[2])
16 wnvar <- inno_model$sigma2
17 cat("MA(2) model parameter:", ma_param, "\n")
18 cat("AICC:", aicc, "\n")
19 print("Standard deviations for first two MA
       parameters:")
20 print(stddev_1);print(stddev_2)
21 cat("White noise variance: ", wnvar)
```

R code Exa 5.1.6 Estimations on Lake data

```
1  # Page No. 137
2  library(itsmr)
3  library(tseries)
4  huron<- read.csv("LAKE.TSM", header = FALSE)
5  colnames(huron)[1] <- 'water'
6  Y_t <- ts(huron$water)
7  X_t <- Y_t - mean(Y_t)
8  arma_model <- arma(X_t, p=1, q=1)
9  ma_param <- arma_model$theta
10 ar_param <- arma_model$phi
11 stderr_phi <- arma_model$se.phi
12 stderr_theta <- arma_model$se.theta
13 aicc <- arma_model$aicc
```

```
14  stddev_phi <- ar_param/(1.96*stderr_phi)
15  stddev_theta <- ma_param/(1.96*stderr_theta)
16  cat("Estimated AR coefficient: ", ar_param, "\n")
17  cat("Estimated MA coefficient: ", ma_param, "\n")
18  cat("AICC: ", aicc, "\n")
19  cat(" Standard deviations: ", stddev_phi, " ",
       stddev_theta)
20  find_conf <- function(param, stderr){
21    low <- param - (stderr*1.96)
22    high <- param + (stderr*1.96)
23    x <- c(low, high)
24    return (x)
25  }
26  conf_phi <- find_conf(ar_param, stderr_phi)
27  cat("95% Confidence Bounds for phi: ", conf_phi)
28  conf_theta <- find_conf(ma_param, stderr_theta)
29  cat("95% Confidence Bounds for theta: ", conf_theta)
```

R code Exa 5.1.7 Lake data analysis using Hannan algorithm

```
1  # Page No. 138
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  library(tseries)
5  huron<- read.csv("LAKE.TSM", header = FALSE)
6  colnames(huron)[1] <- 'water'
7  time_series <- ts(huron$water)
8  Y_t <- time_series
9  X_t <- Y_t - mean(Y_t)
10 p <- 1
11 q <- 1
12 h_model <- hannan(X_t, p, q)
13 ar_param <- h_model$phi
14 ma_param <- h_model$theta
```

```
15 aicc <- h_model$aicc
16 stderr_phi <- h_model$se.phi
17 stderr_theta <- h_model$se.theta
18 stddev_phi <- ar_param/(1.96*stderr_phi)
19 stddev_theta <- ma_param/(1.96*stderr_theta)
20 cat("Estimated AR coefficient: ", ar_param, "\n")
21 cat("Estimated MA coefficient: ", ma_param, "\n")
22 cat("AICC: ", aicc, "\n")
23 cat(" Standard deviations, phi and theta
       respectively: ", stddev_phi, stddev_theta)
24 find_conf <- function(param, stderr){
25    low <- param - (stderr*1.96)
26    high <- param + (stderr*1.96)
27    x <- c(low, high)
28    return (x)
29 }
30 confs_phi <- find_conf(ar_param,stderr_phi)
31 cat("95% Confidence Bounds for phi: ",confs_phi)
32 confs_theta <- find_conf(ma_param,stderr_theta)
33 cat("95% Confidence Bounds for theta: ",confs_theta)
```

R code Exa 5.2.4 Burg and yule walker model comparison

```
1 # Page No. 143
2 # Downloading link: https://storage.googleapis.com/
     springer-extras/zip/2002/978-0-387-21657-7.zip
3 library(itsmr)
4 library(tseries)
5 dow<- read.csv("DOWJ.TSM", header = FALSE)
6 colnames(dow)[1]<- "jones"
7 dowjones <- ts(dow$jones)
8 dowjones_diff <- diff(dowjones, lag = 1)
9 dow_mean_diff <- dowjones_diff - mean(dowjones_diff)
10 p <- 1; q <- 0; n <- length(dow_mean_diff)
11 ywmodel <- yw(dow_mean_diff, p)
```

```
12  bmodel <- burg(dow_mean_diff, p)
13  model <- autofit(dow_mean_diff, p=0:5, q=0:5)
14  aicc <- model$aicc
15  aicc_yw <- ywmodel$aicc
16  aicc_b <- bmodel$aicc
17  LL_yw <- aicc_yw - (2*(p+q+1)*n/(n-p-q-2))
18  LL_b <- aicc_b - (2*(p+q+1)*n/(n-p-q-2))
19  LL <- aicc - (2*(p+q+1)*n/(n-p-q-2))
20  b_param <- bmodel$phi
21  stderr <- model$se.phi
22  ar_param <- model$phi
23  find_conf <- function(param, stderr){
24    low <- param - (stderr*1.96)
25    high <- param + (stderr*1.96)
26    x <- c(low, high)
27    return (x)
28  }
29  confs <- find_conf(ar_param,stderr)
30
31  cat("Minimum AICC:",aicc,"\n")
32  cat("Standard error:",stderr,"\n")
33  cat("95% Confidence Bounds: ",confs)
34  cat("Log likelihood for autofit:",LL,"\n")
35  cat("Parameters in burg model:",b_param,"\n")
36  cat("Log likelihood for yule walker:",LL_yw,"\n")
37  cat("Log likelihood for burg:",LL_b,"\n")
```

R code Exa 5.2.5 Autofit on Lake data

```
1  # Page No. 144
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  library(tseries)
5  hudson <- read.csv("LAKE.TSM", header = FALSE)
```

```r
6  colnames(hudson)[1]<- "level"
7  Y_t <- ts(hudson$level)
8  X_t <- Y_t - mean(Y_t)
9  arma_model <- autofit(X_t, p=0:5, q=0:5)
10 aicc <- arma_model$aicc
11 ar_param <- arma_model$phi
12 ma_param <- arma_model$theta
13 stderr_phi <- arma_model$se.phi
14 stderr_theta <- arma_model$se.theta
15 stddev_phi <- ar_param/(1.96*stderr_phi)
16 stddev_theta <- ma_param/(1.96*stderr_theta)
17 find_conf <- function(param, stderr){
18   low <- param - (stderr*1.96)
19   high <- param + (stderr*1.96)
20   x <- c(low, high)
21   return (x)
22 }
23 confs_phi <- find_conf(ar_param,stderr_phi)
24 confs_theta <- find_conf(ma_param,stderr_theta)
25 cat("AICC:",aicc,"\n")
26 cat("AR Parameter:",ar_param,"\n")
27 cat("MA Parameter",ma_param,"\n")
28 cat("Standard deviations for phi and theta:",stddev_
     phi,stddev_theta,"\n")
29 print("95% Confidence intervals:")
30 cat("for phi:",confs_phi,"\n")
31 cat("for theta:",confs_theta,"\n")
```

R code Exa 5.4.1 Forecasts on overshorts data

```r
1 # Page No.  147
2 # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3 # Answer may vary due to randomization
4 library(tseries)
```

```
5  library(forecast)
6  oshorts<- read.csv("OSHORTS.TSM",header = FALSE)
7  colnames(oshorts)[1]<- "overshorts"
8  Xts <- ts(oshorts$overshorts)
9  best_model <- auto.arima(Xts,max.order = 1, stepwise
       = FALSE, approximation = FALSE)
10 best_model$coef
11 ma_model <- arima(Xts, order = c(0, 0, 1))
12 predictions <- predict(ma_model,7)
13 mean_Xts <- mean(Xts)
14 predicted_values <- as.numeric(predictions$pred)
15 mse <- sqrt(mean((Xts - mean_Xts)^2 ))
16 cat("Predicted Values:\n")
17 print(predicted_values)
18 cat("Mean Squared Error (MSE):\n")
19 print(mse)
```

R code Exa 5.5.1 FPE based selection of an AR model for Lake data

```
1  # Page No. 150
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(tseries)
4  library(itsmr)
5  huron<- read.csv("LAKE.TSM", header = FALSE)
6  colnames(huron)[1] <- 'water'
7  Y_t <- ts(huron$water)
8  X_t <- Y_t - mean(Y_t)
9  ar_orders <- 1:10
10 fpe_values <- numeric(length(ar_orders))
11 sigma_squared_values <- numeric(length(ar_orders))
12 for (p in ar_orders) {
13   ar_model <- arma(X_t, p=p, q=0)
14   n <- length(X_t)
15   sigma_squared <- ar_model$sigma2
```

```
16    fpe_values[p] <- (n + p) / (n - p) * sigma_squared
17    sigma_squared_values[p] <- sigma_squared
18 }
19 for (p in ar_orders) {
20    cat("Order", p, "- FPE:", fpe_values[p], "Sigma^2:
         ", sigma_squared_values[p], "\n")
21 }
```

R code Exa 5.5.2 AICC based model selection

```
1 # Page No. 153
2 # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3 library(forecast)
4 library(tseries)
5 huron<- read.csv("LAKE.TSM", header=FALSE)
6 colnames(huron)[1] <- 'water'
7 Y_t <- ts(huron$water)
8 X_t <- Y_t - mean(Y_t)
9 p <- 1 ; q <- 1
10 best_model2 <- arma(X_t, p=p, q=q)
11 cat("Best ARIMA model based on AICC:\n")
12 print(best_model2$aicc)
13 p <- 2; q <- 0
14 best_model1 <- arma(X_t, p=p, q=q)
15 cat("Best ARIMA model based on AICC:\n")
16 print(best_model1$aicc)
```

Chapter 6

Nonstationary and Seasonal time series models

R code Exa 6.1.1 ARIMA 1 1 0 Process

```
1  # Page No. 159
2  # Answer may vary due to randomization
3  library(forecast)
4  library(ggplot2)
5  phi <- 0.8
6  sigma2 <- 1
7  n <- 200
8  set.seed(123)
9  Xt <- arima.sim(model = list(order = c(1,1,0), ar =
      phi), n = n, sd = sqrt(sigma2))
10 # Figure 6-1
11 autoplot(Xt) +
12   ggtitle("ARIMA(1,1,0)") +
13   geom_point()+
14   xlab("Time") +
15   ylab("Xt") +
16   theme_minimal()
17 # Figure 6-2
18 acf_plot <- ggAcf(Xt) +
```

```
19    ggtitle("Sample ACF") +
20    theme_minimal()
21 print(acf_plot)
22 # Figure 6-3
23 pacf(Xt, main ="Sample PACF")
24 # Figure 6-4
25 Yt <- diff(Xt)
26 plot(Yt)
```

R code Exa 6.2.1 Burg model on Australian wine data

```
 1 # Page no. 168
 2 # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
 3 # Answer may vary due to specific software features
 4 library(forecast)
 5 library(tseries)
 6 library(itsmr)
 7 wine_data <- read.csv("WINE.TSM", header = FALSE)
 8 colnames(wine_data)[1] <- 'Sales'
 9 winedata <- ts(wine_data$Sales)
10 M <- c("season",12, "trend",1)
11 newwine <- Resid(winedata,M)
12 plot(newwine, type='l')
13 M <- c("log","diff",12)
14 newwine <- Resid(winedata,M)
15 plot(newwine, type='l')
16 acf(newwine)
17 pacf(newwine)
18 Wts <- newwine-mean(newwine)
19 burg_model <- burg(Wts, p=12)
20 print(burg_model)
21 arma_model <- autofit(Wts, p=0:15, q=0)
22 print(arma_model)
```

48

R code Exa 6.2.2 Autofit for minimum AICC model

```
1  # Page No. 169
2  library(tseries)
3  library(itsmr)
4  huron<- read.csv("LAKE.TSM", header=FALSE)
5  colnames(huron)[1] <- 'water'
6  Y_t <- ts(huron$water)
7  X_t <- Y_t - mean(Y_t)
8  model <- autofit(X_t,p=0:2,q=0:2)
9  cat("Phi:\n", model$phi)
10 cat("Theta:\n", model$theta)
11 cat("Variance:\n", model$sigma2)
12 cat("AICC:\n", model$aicc)
```

R code Exa 6.3.1 Test statistic on simulated data

```
1  # Page no. 171
2  # Answer may vary due to randomization
3  library(forecast)
4  library(tseries)
5  phi <- 0.8
6  sigma2 <- 1
7  n <- 200
8  set.seed(123)
9  X0 <- 0
10 Xt <- arima.sim(model = list(order = c(1,1,0), ar =
      phi), n = n, sd = sigma2)
11 Xt <- c(X0, Xt)
12 dXt <- diff(Xt)
13 Xt_lag1 <- lag(Xt, 1)
14 dXt_lag1 <- lag(dXt, 1)
```

```
15  dXt_lag2 <- lag(dXt, 2)
16  valid_indices <- 4:200
17  reg_data <- data.frame(
18    dXt = dXt[valid_indices - 1],
19    Xt_lag1 = Xt[valid_indices - 1],
20    dXt_lag1 = dXt[valid_indices - 2],
21    dXt_lag2 = dXt[valid_indices - 3]
22  )
23  reg_model <- lm(dXt ~ Xt_lag1 + dXt_lag1 + dXt_lag2,
        data = reg_data)
24  coeff_Xt_lag1 <- summary(reg_model)$coefficients["Xt
      _lag1", "Estimate"]
25  se_Xt_lag1 <- summary(reg_model)$coefficients["Xt_
      lag1", "Std. Error"]
26  test_statistic <- coeff_Xt_lag1 / se_Xt_lag1
27  cat("Test statistic for unit root:", test_statistic,
        "\n")
```

R code Exa 6.3.2 Model parameters for overshorts data

```
1  # Page No. 173
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(tseries)
4  library(forecast)
5  oshorts= read.csv("OSHORTS.TSM", header = FALSE)
6  colnames(oshorts)[1] <- 'overshorts'
7  Xts <- ts(oshorts$overshorts)
8  Y_t <- Xts + 4.035
9  best_model <- auto.arima(Y_t, stepwise = FALSE,
      approximation = FALSE)
10  print(best_model$coef)
11  print((-2)*logLik(best_model))
```

R code Exa 6.4.1 ARIMA 1 1 0 model on Dow jones utilities index

```
1  # Page No. 176
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  library(tseries)
5  dow<- read.csv("DOWJ.TSM", header = FALSE)
6  colnames(dow)[1]<- "jones"
7  dowjones <- ts(dow$jones)
8  dowjones_diff <- diff(dowjones, lag = 1)
9  M = c("diff", 1)
10 dowj <- Resid(dowjones,M)
11 dowj <- dowj - mean(dowj)
12 p <- 1; q <- 0;
13 bmodel <- burg(dowj, p)
14 cat("Mean squared error",bmodel$sigma2)
15 print(bmodel)
```

R code Exa 6.5.2 ACF of seasonal MA model

```
1  # Page no. 178
2  # Answer may vary due to randomization
3  library(forecast)
4  set.seed(123)
5  n <- 500
6  U_t <- rnorm(n)
7  lag <- 12
8  X_t <- U_t
9  X_t[(lag + 1):n] <- U_t[(lag + 1):n] - 0.4 * U_t[1:(
       n - lag)]
10 acf(X_t, main="ACF")
```

R code Exa 6.5.3 ACF of seasonal AR model

```
1  # Page no. 179
2  # Answer may vary due to randomization
3  library(forecast)
4  set.seed(123)
5  n <- 500
6  U_t <- rnorm(n)
7  X_t <- numeric(n)
8  X_t[1:12] <- U_t[1:12]
9  for (t in (12 + 1):n) {
10    X_t[t] <- U_t[t] + 0.7 * X_t[t - 12]
11 }
12 acf(X_t, main="ACF")
```

R code Exa 6.5.4 ACF of monthly accidental deaths data

```
1  # Page no. 180
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  # Answer may vary due to specific software features.
4  library(forecast)
5  library(astsa)
6  library(itsmr)
7  deaths= read.csv("DEATHS.TSM", header = FALSE)
8  colnames(deaths)[1] <- "deaths"
9  deaths$months=seq(as.Date("1973-01-01"), as.Date("
       1978-12-01"),by='month')
10 diff1 <- diff(deaths$deaths, lag = 12)
11 Yt <- ts(diff(diff1),frequency = 12)
12 # Figure 6-17
```

```
13  acf(Yt, main="ACF")
14  best_model <- auto.arima(Yt, seasonal=TRUE, stepwise
        = FALSE, approximation = FALSE)
15  print(best_model)
16  sarima_model <- Arima(Yt, order = c(0, 1, 1),
        seasonal = c(0, 1, 1))
17  model_params <- sarima_model$coef
18  print(model_params)
```

R code Exa 6.5.5 Forecasting monthly accidental deaths

```
1   # Page no. 180
2   # Answer may vary due to specific software features.
3   library(forecast)
4   library(itsmr)
5   deaths= read.csv("DEATHS.TSM", header = FALSE)
6   dts <- ts(deaths,frequency = 12)
7   dts_diff_12 <- diff(dts, lag = 12)
8   dts_diff_12_1 <- diff(dts_diff_12, lag = 1)
9   dts_mean_corrected <- dts_diff_12_1 - mean(dts_diff_
        12_1)
10  fit <- arma(dts_mean_corrected,p=0,q=13)
11  M <- c("diff",12,"diff",1)
12  forecast_values <- forecast(dts,M,fit,h = 6)
```

R code Exa 6.6.1 GLS based Model parameter estimation

```
1   # Page no. 187
2   # Downloading link: https://storage.googleapis.com/
        springer-extras/zip/2002/978-0-387-21657-7.zip
3   library(itsmr)
4   library(nlme)
5   oshorts= read.csv("OSHORTS.TSM", header = FALSE)
```

53

```
 6  colnames(oshorts)[1] <- "overshorts"
 7  oshorts$time <- seq(1,length(oshorts$overshorts))
 8  ots <- ts(oshorts$overshorts)
 9  ots <- ots-mean(ots)
10  oshorts$overshorts <- oshorts$overshorts-mean(
       oshorts$overshorts)
11  a <- autofit(ots, p=0, q=1)
12  print(a$theta)
13  cat("OLS beta:",mean(oshorts$overshorts))
14  acv <- acf(oshorts$overshorts,type = 'covariance',
       plot=FALSE)
15  cat("Estimator for beta: ",acv$acf[1]/length(ots))
16  model_formula <- overshorts ~ time
17  gls_model <- gls(model_formula, data = oshorts)
18  summary(gls_model)
```

R code Exa 6.6.2 Model parameters estimation for Lake data

```
 1  # Page no. 189
 2  library(forecast)
 3  library(nlme)
 4  hudson<- read.csv("LAKE.TSM", header = FALSE)
 5  colnames(hudson)[1] <- 'level'
 6  hudson$t <- seq(1, length(hudson$level))
 7  ols_model <- lm(hudson$level ~ hudson$t)
 8  ols_residuals <- residuals(ols_model)
 9  beta1_hat <- coef(ols_model)[1]
10  cat("OLS estimate of beta1:", beta1_hat, "\n")
11  ar2_model <- Arima(ols_residuals, order=c(2,0,0))
12  phi1_hat <- coef(ar2_model)["ar1"]
13  phi2_hat <- coef(ar2_model)["ar2"]
14  sigma2_hat <- ar2_model$sigma2
15  cat("phi1:",phi1_hat)
16  cat("phi2:",phi2_hat)
17  cat("std. dev.:",sigma2_hat)
```

```
18  glsEstimate() <- gls(lm(level~t),data = hudson)
```

R code Exa 6.6.3 Seat belt legislation

```
1  # Page no. 189
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  library(nlme)
5  library(ggplot2)
6  seat<- read.csv("SBL.TSM", header = FALSE)
7  gt <- read.csv("SBLDIN.TSM", header = FALSE)
8  colnames(gt)[1] <- 'Y'
9  colnames(seat)[1] <- "acc"
10 seat$Years <- seq(as.Date("1975-01-01"), as.Date("
      1984-12-01"), by = "month")
11 ggplot(seat, aes(x = Years, y = acc)) +
12   geom_point(shape = 15, size = 1) +
13   geom_line() +
14   labs(title = "Road injuries (Jan 1975 - Dec 1984)"
        ,
15       x = "Months",
16       y = "Injuries") +
17   theme_minimal()
18 # Prediction may differ due to specific software
      methods
19 Yt <- ts(seat$acc)
20 Xt <-  Yt-diff(Yt,lag = 12)
21 data <- data.frame(X = Xt,Y = gt)
22 gls_model <- gls(X~Y, data = data)
23 fitted_values <- fitted(gls_model)
24 seat <-seat[-c(1:12), ]
25 seat$fit <- fitted_values
26 plot(seat$Years,seat$acc, main = "Original Data and
      Fitted GLS Line",
```

```
27        xlab = "Time", ylab = "Value", type = "o-")
28  lines(seat$Years, fitted_values, col = "red", lwd =
       2)
```

Chapter 7

Time Series Models for Financial Data

R code Exa 7.2.1 ARCH 1 Series

```r
1  # Page no. 199
2  # Answer may vary due to randomization
3  alpha0 <- 1
4  alpha1 <- 0.5
5  n <- 1000
6  set.seed(123)
7  epsilon <- rnorm(n)
8  sigma2 <- numeric(n)
9  y <- numeric(n)
10 for (t in 2:n) {
11   sigma2[t] <- alpha0 + alpha1 * y[t-1]^2
12   y[t] <- sqrt(sigma2[t]) * epsilon[t]
13 }
14 plot(y, type = "l", main = "Simulated ARCH(1)
      Process", xlab = "Time", ylab = "Value")
15 acf(y)
```

R code Exa 7.2.2 Fitting GARCH models to stock data

```
1  # Page No. 201
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  library(tseries)
5  library(rugarch)
6  E1032<- read.csv("E1032.TSM")
7  char_array <- E1032[39:193,]
8  matches <- gregexpr("-?[0-9.]+(?:\\s*[Ee
       ][+-]?[0-9]+)?", char_array)
9  stock <- ts(as.numeric(unlist(regmatches(char_array,
       matches))))
10 garch_spec <- ugarchspec(mean.model = list(armaOrder
       = c(0,0)),
11                          variance.model = list(model
                                 = "sGARCH", garchOrder
                                 = c(1,1)))
12 garch_fit <- ugarchfit(data = stock, spec = garch_
       spec)
13 sigma <- sigma(garch_fit)
14 par(mfrow=c(2,1))
15 plot(stock,type = 'l', col = 'blue',ylab = '
       percentage returns')
16 plot(sigma, type = 'l', col = 'red', ylab = '
       Volatility')
```

R code Exa 7.2.3 Fitting ARMA Models Driven by GARCH Noise

```
1  # Page No. 203
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  # Answer may vary due to software specifications
```

```
5  library(forecast)
6  library(tseries)
7  library(rugarch)
8  sunspot<- read.csv("SUNSPOTS.TSM")
9  colnames(sunspot)[1]<- "spots"
10 sunspots<- ts(sunspot$spots)
11 sunspots_mean_corrected <- sunspots - mean(sunspots,
       na.rm = TRUE)
12 fit_arima <- Arima(sunspots_mean_corrected, order =
     c(4,0,3))
13 print(fit_arima)
14 residuals_arima <- fit_arima$residuals
15 p <- 1
16 q <- 1
17 spec <- ugarchspec(variance.model = list(model = "
     sGARCH", garchOrder = c(p, q)),
18                     mean.model = list(armaOrder = c
                         (4, 3), include.mean = TRUE),
19                     distribution.model = "norm")
20 fit_garch <- ugarchfit(spec = spec, data = residuals
     _arima)
21 print(fit_garch)
22 n <- as.numeric(length(sunspots_mean_corrected))
23 aicc <- (((-2)*(fit_garch@fit$LLH))*(n/(n-p)))+ (((p
     +q+2)*(2*n))/(n-p-q-2))
24 print(paste("AICC value for the GARCH model:", aicc)
     )
25 print("Parameters of the GARCH(1,1) model:")
26 print(coef(fit_garch))
```

R code Exa 7.5.1 Brownian motion

```
1 # Page no. 213
2 # Answer may vary due to randomization
3 T <- 10; n <- 1000; dt <- T / n
```

```
4  time_points <- seq(0, T, by = dt)
5  set.seed(123)
6  increments <- rnorm(n, mean = 0, sd = sqrt(dt))
7  B_t <- c(0, cumsum(increments))
8  plot(time_points, B_t, type = "l",
9       main = "Standard Brownian Motion B(t)",
10      xlab = "Time", ylab = "B(t)",
11      col = "blue", lwd = 2)
```

R code Exa 7.5.2 Poisson process

```
1  # Page no. 214
2  lambda <- 5
3  T <- 10
4  set.seed(123)
5  jump_times <- cumsum(rexp(100, rate = lambda))
6  jump_times <- jump_times[jump_times <= T]
7  N_t <- seq_along(jump_times)
8  jump_times <- c(0, jump_times)
9  N_t <- c(0, N_t)
10 plot(jump_times, N_t, type = "s",
11      main = "Poisson Process N(t)",
12      xlab = "Time", ylab = "N(t)",
13      col = "blue", lwd = 2)
```

R code Exa 7.5.3 Compound Poisson Process

```
1  # Page no. 214
2  lambda <- 5; T <- 10; mu <- 0; sigma <- 1
3  set.seed(123)
4  jump_times <- cumsum(rexp(100, rate = lambda))
5  jump_times <- jump_times[jump_times <= T]
```

```r
6  jump_sizes <- rnorm(length(jump_times), mean = mu,
      sd = sigma)
7  X_t <- cumsum(jump_sizes)
8  jump_times <- c(0, jump_times)
9  X_t <- c(0, X_t)
10 plot(jump_times, X_t, type = "s",
        main = "Compound Poisson Process X(t)",
        xlab = "Time", ylab = "X(t)",
        col = "blue", lwd = 2)
```

Chapter 8

Multivariate Time Series

R code Exa 8.1.1 Dow Jones and All Ordinaries Indices

```
1  # Page No. 229
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(forecast)
4  library(tseries)
5  dow<- read.csv("DJAO2.TSM", header = FALSE)
6  pc <- read.csv("DJAOPC2.TSM", header = FALSE)
7  colnames(pc)[1]<- "stocks"
8  char_array <- dow[,1]
9  matches <- gregexpr("\\b\\d{3,}\\b", char_array)
10 stock <- as.numeric(unlist(regmatches(char_array,
       matches)))
11 dowjones <- ts(stock[c(TRUE, FALSE)])
12 Aus <- ts(stock[c(FALSE, TRUE)])
13 index <- seq_along(dowjones)
14 plot(index, dowjones, type = 'l', col = 'blue', lwd
       = 2, ylim = range(c(dowjones,1000)),
15      xlab = 'Index', ylab = 'Values', main = 'Dow
           jones and Australian ordinary')
16 lines(index, Aus, col = 'red', lwd = 2)
17
```

```
18 pcs <- separate(pc, col = 1, into = c("dow", "aus"),
       sep = "\\s+")
19 dowjones1 <- ts(as.numeric(pcs$dow))
20 Aus1 <- ts(as.numeric(pcs$aus))
21 acf(dowjones1, main = "Series 1")
22 acf(Aus1, main = "Series 2")
23 ccf1 <- ccf(dowjones1, Aus1,plot = FALSE)
24 positive_lag1 <- ccf1$lag >= 0
25 plot(ccf1$lag[positive_lag1], ccf1$acf[positive_lag1
      ], type = "h",
26        main = "Series 1 * Series 2",
27        xlab = "Lag", ylab = "CCF")
28 abline(h = 0)
29 ccf2 <- ccf(Aus1,dowjones1,plot = FALSE)
30 positive_lag2 <- ccf2$lag >= 0
31 plot(ccf2$lag[positive_lag2], ccf2$acf[positive_lag2
      ], type = "h",
32        main = "Series 2 * Series 1",
33        xlab = "Lag", ylab = "CCF")
34 abline(h = 0)
35 plot(lag(dowjones1, -1), Aus1, main="Scatterplot",
36        xlab="Lagged TS1", ylab="TS2", pch=19)
```

R code Exa 8.1.2 Sales with a leading indicator

```
1 # Page No. 230
2 # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3 library(forecast)
4 library(tseries)
5 sales<- read.delim("SALES.TSM", header = FALSE)
6 leads<- read.delim("LEAD.TSM", header = FALSE)
7 colnames(sales)[1]<- "sale"
8 colnames(leads)[1]<- "lead"
9 ls2 <-  cbind(sales, leads)
```

```
10  lst <- ts(ls2)
11  lst <- diff(lst)
12  par(mfrow = c(2, 2))
13  acf(lst[, 2], main = "Series 1")
14  acf(lst[, 1], main = "Series 2")
15  ccf1 <- ccf(lst[, 1], lst[, 2],plot = FALSE)
16  positive_lag1 <- ccf1$lag >= 0
17  plot(ccf1$lag[positive_lag1], ccf1$acf[positive_lag1
       ], type = "h",
18      main = "Series 2 * Series 1",
19      xlab = "Lag", ylab = "CCF")
20  abline(h = 0)
21  ccf2 <- ccf(lst[,2],lst[,1],plot = FALSE)
22  positive_lag2 <- ccf2$lag >= 0
23  plot(ccf2$lag[positive_lag2], ccf2$acf[positive_lag2
       ], type = "h",
24      main = "Series 1 * Series 2",
25      xlab = "Lag", ylab = "CCF")
26  abline(h = 0)
```

R code Exa 8.3.1 Sample correlations

```
1  # Page No. 239
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(forecast)
4  library(tseries)
5  E731 <- read.delim("E731A.TSM", header=FALSE)
6  Ets <- ts(E731)
7  par(mfrow = c(2, 2))
8  acf(Ets[, 2], main = "Series 1")
9  acf(Ets[, 1], main = "Series 2")
10  ccf1 <- ccf(Ets[, 1], Ets[, 2],plot = FALSE)
11  positive_lag1 <- ccf1$lag >= 0
12  plot(ccf1$lag[positive_lag1], ccf1$acf[positive_lag1
```

```
          ], type = "h",
13          main = "Series 1 * Series 2",
14          xlab = "Lag", ylab = "CCF")
15  abline(h = 0)
16  ccf2 <- ccf(Ets[,2],Ets[,1],plot = FALSE)
17  positive_lag2 <- ccf2$lag >= 0
18  plot(ccf2$lag[positive_lag2], ccf2$acf[positive_lag2
          ], type = "h",
19          main = "Series 2 * Series 1",
20          xlab = "Lag", ylab = "CCF")
21  abline(h = 0)
```

R code Exa 8.6.1 Multivariate models fitted on stock data

```
1  # Page No. 249
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  # Answer may vary to unspecified function in problem
4  library(tidyr)
5  library(vars)
6  pc <- read.csv("DJAOPC2.TSM", header = FALSE)
7  pcs <- separate(pc, col = 1, into = c("dow", "aus"),
       sep = "\\s+")
8  pcs$dow <- as.numeric(pcs$dow)
9  pcs$aus <- as.numeric(pcs$aus)
10  pcs_ts <- ts(pcs)
11  var_model <- VAR(pcs_ts,p=1,type = "none")
12  summary(var_model)
```

R code Exa 8.6.2 Multivariate models fitted on sales data

```
1  # Page No. 249
```

```
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(vars)
4  library(tidyr)
5  library(stringr)
6  library(dplyr)
7  ls <- read.csv("LS2.TSM",header = FALSE)
8  colnames(ls)[1]<- "ll"
9  ls$ll <- trimws(ls$ll,which = "left")
10 lts <- separate(ls, col = ll, into = c("ld", "sales"
       ), sep = "\\s+")
11 lts$ld <- as.numeric(lts$ld)
12 lts$sales <- as.numeric(lts$sales)
13 lts <- ts(lts)
14 ltds <- diff(lts, lag = 1)
15 lag<-VARselect(lts,lag.max=10)
16 optimal <- lag$selection
17 estim <- VAR(ltds,p=5,type = "none")
18 summary(estim)
19 estim$varresult
```

R code Exa 8.6.3 VAR 1 model on stock data

```
1  # Page No. 251
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(tidyr)
4  library(itsmr)
5  library(vars)
6  pc <- read.csv("DJAOPC2.TSM", header = FALSE)
7  pcs <- separate(pc, col = 1, into = c("dow", "aus"),
       sep = "\\s+")
8  pcs$dow <- as.numeric(pcs$dow)
9  pcs$aus <- as.numeric(pcs$aus)
10 pcs_ts <- ts(pcs)
```

```
11  var_model <- VAR(pcs_ts,p=1,type = "none")
12  summary(var_model)
13  k <- 9
14  n <- length(pcs_ts)
15  log_likelihood <- LogLik(var_model)
16  aicc <- -2 * log_likelihood + 2 * k + (2 * k * (k +
        1)) / (n - k - 1)
17  arm <- autofit(ts(pcs$aus),p=0:2,q=0)
18  print(arm)
```

Chapter 9

State Space Models

R code Exa 9.2.1 Random walk plus noise model

```
1  # Page no.261
2  # Answer varies due to randomness
3  set.seed(46)
4  n <- 100
5  sigma_v <- 4
6  sigma_w <- 8
7  M <- cumsum(rnorm(n, mean = 0, sd = sqrt(sigma_w)))
8  W <- rnorm(n, mean = 0, sd = sqrt(sigma_v))
9  Y <- M + W
10 plot(1:n, M, type = "l", col = "blue", xlab = "Time"
       , ylab = "Value",
11     main = "Random Walk Plus Noise Model")
12 points(1:n, Y, pch = 15, col = "red")
13 acf(diff(Y), lag.max = 20)
```

R code Exa 9.5.2 International airline passengers

```
1  # Page No. 278
```

```
 2 # Downloading link: https://storage.googleapis.com/
     springer−extras/zip/2002/978−0−387−21657−7.zip
 3 # Adequate data not provided in example
 4 library(ggplot2)
 5 library(MASS)
 6 library(KFAS)
 7 airpass <- read.csv("AIRPASS.TSM", header = FALSE)
 8 colnames(airpass)[1] <- "pass"
 9 ggplot(airpass, aes(x = seq(as.Date("1949−01−01"),
     as.Date("1960−12−01"), by = "month"), y = pass))
     +
10   geom_point() +
11   geom_line() +
12   labs(title = "Air passengers (Jan 1949 − Dec 1960)
     ",
13        x = "Time",
14        y = "Passengers") +
15   theme_minimal()
16 pass <- ts(airpass$pass)
```

R code Exa 9.8.3 Polio in the USA

```
 1 # Page No. 292
 2 # Downloading link: https://storage.googleapis.com/
     springer−extras/zip/2002/978−0−387−21657−7.zip
 3 library(ggplot2)
 4 library(dplyr)
 5 polio <- read.csv("POLIO.TSM", header = FALSE)
 6 colnames(polio)[1] <- "pol"
 7 ggplot(polio, aes(x = seq(as.Date("1970−01−01"), as.
     Date("1983−12−01"), by = "month"), y = pol)) +
 8   geom_point() +
 9   geom_line() +
10   labs(title = "Polio in US (Jan 1970 − Dec 1983)",
11        x = "Time",
```

```
12            y = "Polio cases") +
13    theme_minimal()
14  polio$Month <- 1:length(polio$pol)
15  polio <- polio %>%
16    mutate(
17      t = Month,
18      u1 = 1,
19      u2 = t / 1000,
20      u3 = cos(2 * pi * t / 12),
21      u4 = sin(2 * pi * t / 12),
22      u5 = cos(2 * pi * t / 6),
23      u6 = sin(2 * pi * t / 6)
24    )
25  model <- lm(pol ~ u1 + u2 + u3 + u4 + u5 + u6, data
       = polio)
26  polio$Trend <- fitted(model)
27  ggplot(polio, aes(x = Month)) +
28    geom_point(aes(y = pol, color = "Actual Cases")) +
29    geom_line(aes(y = Trend, color = "Trend Estimate")
         ) +
30    labs(
31      title = "Trend Estimate for Monthly U.S. Polio
           Cases",
32      x = "Month",
33      y = "Number of Cases",
34      color = "Legend"
35    ) +
36    scale_color_manual(values = c("Actual Cases" = "
         blue", "Trend Estimate" = "red")) +
37    theme_minimal()
```

R code Exa 9.8.7 Goals Scored by England Against Scotland

```
1  # Page No. 299
2  # Downloading link: https://storage.googleapis.com/
```

```
         springer−extras/zip/2002/978−0−387−21657−7.zip
 3  # Answer varies due to inadequate data
 4  library(ggplot2)
 5  library(tidyr)
 6  library(itsmr)
 7  goals <- read.table("GOALS.TSM", header = FALSE)
 8  colnames(goals)[1] <- "goal"
 9  colnames(goals)[2] <-"Year"
10  # Figure 9−8
11  ggplot(goals, aes(x = Year, y = goal)) +
12    geom_point() +
13    geom_line(col='blue') +
14    labs(title = "Goals by England",
15         x = "Years",
16         y = "Goals") +
17    theme_minimal()
18  # Figure 9−9
19  ggplot(na.omit(goals), aes(x = factor(goal))) +
20    geom_bar() +
21    xlab("Goals") +
22    ylab("Count") +
23    ggtitle("Histogram of Goals") +
24    theme_minimal()
25
26  data <- na.omit(goals)
27  delta_hat <- 0.844
28  alpha_0 <- 0.154
29  lambda_0 <- delta_hat / (1 - delta_hat)
30  n <- nrow(data)
31  alpha <- numeric(n);lambda <- numeric(n);pred <-
        numeric(n)
32  alpha[1] <- alpha_0
33  lambda[1] <- lambda_0
34  for (t in 2:n) {
35    alpha[t] <- alpha[t-1] + delta_hat * (data$goal[t
        -1] - alpha[t-1])
36    lambda[t] <- lambda[t-1] + delta_hat * (1 - lambda
        [t-1])
```

```r
37    pred[t] <- alpha[t] / (1 + lambda[t])
38 }
39
40 ggplot(data.frame(Time = data$Year, pred = pred),
       aes(x = Time, y = pred)) +
41    geom_line(color = "blue") +
42    geom_point(data = data, aes(x = Year , y = goal),
         color = "red") +
43    xlab("Year") +
44    ylab("Goals") +
45    ggtitle("One-Step Predictors for Goals Data") +
46    theme_minimal()
```

Chapter 10

Forecasting Techniques

R code Exa 10.1.1 Predicted deaths by ARAR algorithm

```
1  # Page No. 312
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(itsmr)
4  library(forecast)
5  deaths <- read.csv("DEATHS.TSM", header = FALSE)
6  colnames(deaths)[1]<- "death"
7  dts <- ts(deaths$death)
8  arar_model <- arar(dts,h=24,opt=2)
```

R code Exa 10.2.1 Holt Winters non seasonal forecast

```
1  # Page No. 316
2  # Answer may vary due to the nature of forecast
       function.
3  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
4  library(forecast)
```

```r
5  deaths <- read.csv("DEATHS.TSM", header = FALSE)
6  colnames(deaths)[1]<- "death"
7  dts <- ts(deaths$death, freq=12, start = 1973)
8  hw_model <- HoltWinters(dts, gamma = FALSE)
9  forecast_values <- forecast::forecast(hw_model, n.
      steps=2)
10 plot(forecast_values, main="Holt-Winters Forecast",
      xlab="Time", ylab="Values")
11 lines(dts, col="blue")
```

R code Exa 10.3.1 Holt Winters seasonal forecast

```r
1  # Page No. 316
2  # Downloading link: https://storage.googleapis.com/
      springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(forecast)
4  deaths <- read.delim("DEATHS.TSM", header = FALSE)
5  colnames(deaths)[1]<- "death"
6  dts <- ts(deaths$death, freq=12, start = 1973)
7  hw_model <- HoltWinters(dts)
8  forecast_values <- forecast::forecast(hw_model, h
      =24)
9  plot(forecast_values, main="Holt-Winters Forecast",
      xlab="Time", ylab="Values")
10 lines(dts, col="blue")
```

Chapter 11

Further Topics

R code Exa 11.4.1 Annual Minimum Water Levels in the Nile

```
1  # Page No. 340
2  # Downloading link: https://storage.googleapis.com/
       springer-extras/zip/2002/978-0-387-21657-7.zip
3  library(ggplot2)
4  nile <- read.csv("NILE.TSM", header = FALSE)
5  colnames(nile)[1]<- "water"
6  plot(nile$water,xlab="time",ylab="water level",main=
       "Nile river",type = 'l')
7  acf(nile$water,main="ACF")
8  best_model <- auto.arima(nile$water, stepwise =
       FALSE, ic="aicc", approximation = FALSE)
9  print(best_model$aicc)
10 best_arfima <-arfima(nile$water,model = best_model)
11 print(best_arfima$aicc)
```