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

August 16, 2025

[1]Funded by a grant from the National Mission on Education through ICT - http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and R codes written in it can be downloaded from the "Textbook Companion Project" section at the website - https://r.fossee.in.

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

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

R code Exa 1.1.3 Accidental deaths

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

R code Exa 1.1.4 Signal Detection Problem

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

R code Exa 1.1.5 Population of the USA

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

```
library(ggplot2)
uspop= read.csv("USPOP.TSM")
names(uspop)[names(uspop) == "X3929214"] <- "population"
start_year=1790
num_repeated=20
interval=10
ggplot(uspop, aes(x=seq_len(num_repeated) * interval + start_year, y = population)) +
  geom_point() +
  geom_line() +
  labs(title = "Population",
       x = "Years",
       y = "US population") +
  theme_minimal()
```

R code Exa 1.1.6 Strikes in USA

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

R code Exa 1.3.3 Random walk

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

R code Exa 1.3.4 Regression on population data

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

```
  labs(title = "US Population",
       x = "Years",
       y = "Population") +
  theme_minimal()
```

R code Exa 1.3.5 Level of Lake Huron

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

R code Exa 1.3.6 Harmonic regression on accidental deaths

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

R code Exa 1.4.6 Random noise

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

```
  geom_point()+
  geom_line() +
  labs(x = "Index", y = "Noise", title = "Simulated
     N(0,1) Noise")+
  theme_minimal()
acf_result <- acf(noise, plot = FALSE)
n <- length(noise)
bounds <- 1.96 / sqrt(n)
acf_df <- data.frame(Lag = acf_result$lag, ACF = acf
   _result$acf)
ggplot(acf_df, aes(x = Lag, y = ACF)) +
  geom_hline(yintercept = c(-bounds, bounds)) +
  geom_hline(yintercept = 0) +
  geom_segment(aes(xend = Lag, yend = 0)) +
  labs(x = "Lag", y = "ACF", title = "Sample
     Autocorrelation Function (ACF)") +
  ylim(-1, 1)+
  theme_minimal()
```

R code Exa 1.5.1 Moving average of strikes

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

```
ggplot()+
  geom_line(data=strike, aes(x = seq(start_year,end_year),y=Moving_Avg))+
  geom_point(data=strike, aes(x = seq(start_year,end_year),y=strike$Strikes))+
  labs(x = "Year", y = "Strikes", title = "Strikes Data with Moving Average")+
  theme_minimal()
# Figure 1-19
ggplot(data=strike, aes(x = seq(start_year,end_year),y=residuals))+
  geom_line()+
  geom_point()+
  labs(x = "Year", y = "Strikes", title = "Strikes Data residuals")+
  theme_minimal()
```

R code Exa 1.5.2 Smooth exponential and low pass filter

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

R code Exa 1.5.3 Differenced series

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

R code Exa 1.5.4 Deseasonalization and seasonal component

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

```r
colnames(deaths)[1] <- "deaths"
decomposition <- decompose(ts(deaths$deaths, frequency = period))
seasonal_component <- decomposition$seasonal
deseasonalized_data <- deaths$deaths - seasonal_component
deseasonalized_df <- data.frame(years = deaths$years, deseasonalized_deaths = deseasonalized_data)
seasonal_component_df <- data.frame(years = deaths$years, seasonal_component = seasonal_component)
# Figure 1-24
ggplot(deseasonalized_df, aes(x = years, y = deseasonalized_deaths)) +
  geom_line(color = "blue") +
  geom_point()+
  labs(x = "Years", y = "Deseasonalized Deaths", title = "Deseasonalized Deaths") +
  theme_minimal()
# Figure 1-25
ggplot(seasonal_component_df, aes(x = years, y = seasonal_component)) +
  geom_line(color = "red") +
  geom_point()+
  labs(x = "Years", y = "Seasonal Component", title = "Seasonal Component") +
  theme_minimal()
```

R code Exa 1.5.5 Estimation of seasonal component

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

```
colnames(deaths)[1] <- "deaths"
deaths$months=seq(as.Date("1973-01-01"), as.Date("1978-12-01"),by='month')
diff1 <- diff(deaths$deaths, lag = 12)
deaths <- slice(deaths,-(1:12))
deaths$diff1 <- diff1
# Figure 1-26
ggplot(deaths, aes(x = months, y = diff1)) +
  geom_point()+
  geom_line() +
  labs(title = "First-Order Differences of deaths Data",
       x = "months", y = "First-Order Differences")+
  theme_minimal()
# Figure 1-27
diff2 <- diff(deaths$diff1)
deaths <- slice(deaths,-1)
deaths$diff2 <- diff2
ggplot(deaths, aes(x = months, y = diff2)) +
  geom_point()+
  geom_line() +
  labs(title = "Second-Order Differences of deaths Data",
       x = "months", y = "Second-Order Differences") +
  theme_minimal()
```

R code Exa 1.6.1 ACF on signal data

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

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

Chapter 2

Stationary Processes

R code Exa 2.4.3 MA1 Process

```
# 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

```r
#..
# Page No. 54
# Downloading link: https://storage.googleapis.com/springer-extras/zip/2002/978-0-387-21657-7.zip
hudson= read.csv("LAKE.TSM")
names(hudson)[names(hudson) == "X10.38"] <- "level"
start_year=1876
end_year=1972
hudson$years <- seq(start_year,end_year)
fit<-lm(level~years,data = hudson)
residuals <- resid(fit)
residuals_df <- data.frame(years = hudson$years, residuals = residuals)
n <- nrow(residuals_df)
phi <- 0.791
model_acf <- function(i) {
  phi^i
}
confidence_bounds <- function(i) {
  1.96 * (n^(-0.5)) * sqrt(((1 - (phi^(2*i))) * (1 + (phi^2))) / (1 - (phi^2)))
}
acf_values <- acf(residuals_df$residuals, plot = FALSE)$acf
upper_conf_bounds <- sapply(1:40, function(i) {
  confidence_bounds(i) + (phi^i)
})
lower_conf_bounds <- sapply(1:40, function(i) {
  (phi^i) - confidence_bounds(i)
})
plot(0:40, acf_values[1:41], type = "h", ylim = c(-1, 1),
     xlab = "Lag", ylab = "ACF", main = "Sample Autocorrelation Function of Residuals (AR(1))")
lines(1:40, upper_conf_bounds, col = "red", lty = 2)
lines(1:40, lower_conf_bounds, col = "red", lty = 2)
```

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

R code Exa 2.5.5 Durbin Levinson and innovations algorithm

```r
# Page no. 64
compute_autocovariance <- function(phi) {
  gamma_0 <- 1 + phi^2
  gamma_1 <- -phi
  return(list(gamma_0 = gamma_0, gamma_1 = gamma_1))
}
innovation_algorithm <- function(gamma) {
  theta_11 <- -gamma$gamma_1 / gamma$gamma_0
  return(list(theta_11 = theta_11))
}
durbin_levinson_algorithm <- function(gamma) {
  phi_11 <- gamma$gamma_1 / gamma$gamma_0
  sigma_1_squared <- gamma$gamma_0 * (1 - phi_11^2)
  return(list(phi_11 = phi_11, sigma_1_squared = sigma_1_squared))
}
phi <- 0.9
gamma <- compute_autocovariance(phi)
theta <- innovation_algorithm(gamma)
phi_result <- durbin_levinson_algorithm(gamma)
cat(paste0("theta_11 = ", theta$theta_11, "\n"))
cat(paste0("phi_11 = ", phi_result$phi_11, "\n"))
```

Chapter 3

ARMA Models

R code Exa 3.1.1 ARMA 1 1

```
# Page no.76
ar_params <- c(0.5)
ma_params <- c(0.4)
is_invertible <- function(ma_params) {
  roots <- polyroot(c(1, ma_params))
  all(abs(roots) > 1)
}

invertibility_status <- is_invertible(ma_params)
invertibility_status
```

R code Exa 3.1.2 AR2 Process

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

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

R code Exa 3.1.3 ARMA 2 1

```
# Page no. 77
ar_params <- c(-0.75, 0.5625)
ma_params <- c(1.25)
is_invertible <- function(ma_params) {
  roots <- polyroot(c(1, ma_params))
  all(abs(roots) > 1)
}
invertibility_status <- is_invertible(ma_params)
invertibility_status
```

R code Exa 3.2.4 General AR2 process

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

```r
xi1 <- 10/9
xi2 <- 2
phi1 <- 1/xi1 + 1/xi2
phi2 <- -(1/xi1) * (1/xi2)
ar_process <- arima.sim(model = list(ar = c(phi1, phi2)), n = n)
acf(ar_process, main = "Sample ACF of AR(2) Process")
# Figure 3-3
xi1 <- -10/9
xi2 <- 2
phi1 <- 1/xi1 + 1/xi2
phi2 <- -(1/xi1) * (1/xi2)
ar_process <- arima.sim(model = list(ar = c(phi1, phi2)), n = n)
acf(ar_process, main = "Sample ACF of AR(2) Process")

# Figure 3-4
xi1 <- complex(real = 2/3, imaginary = 2*sqrt(3)/3)
xi2 <- complex(real = 2/3, imaginary = -2*sqrt(3)/3)
phi1 <- Re(1/xi1 + 1/xi2)
phi2 <- Re(-(1/xi1) * (1/xi2))
ar_process <- arima.sim(model = list(ar = c(phi1, phi2)), n = n)
acf(ar_process, main = "Sample ACF of AR(2) Process")
```

R code Exa 3.2.8 Overshorts series

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

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

R code Exa 3.2.9 The sunspot numbers

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

R code Exa 3.3.4 Numerical prediction of ARMA 2 3

```r
# Page no. 90
# Answer may vary due to randomization in simulation
library(forecast)
ar_params <- c(1,-0.24)
ma_params <- c(0.4, 0.2, 0.1)
set.seed(46)
n <- 10
arma_process <- arima.sim(model = list(ar = ar_params, ma = ma_params), n = n)
print(arma_process)
acf_values <- acf(arma_process, type="covariance", plot=FALSE)$acf
gamma_0 <- acf_values[1]
gamma_1 <- acf_values[2]
gamma_2 <- acf_values[3]
cat("gamma_0 =", gamma_0, "\n")
cat("gamma_1 =", gamma_1, "\n")
cat("gamma_2 =", gamma_2, "\n")
innovations_algorithm <- function(arma_process, n_steps) {
  n <- length(arma_process)
  predictions <- numeric(n_steps)
  e <- numeric(n + n_steps)
  phi <- numeric(n + n_steps)
  theta <- numeric(n + n_steps)
  for (i in 1:n_steps) {
    predictions[i] <- sum(ar_params * arma_process[(n-i+1):(n-i+2)])
    + sum(ma_params * e[(n-i+1):(n-i+3)])
    e[n+i] <- arma_process[i] - predictions[i]
  }
  return(predictions)
```

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

R code Exa 3.3.5 h step prediction of ARMA

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

Chapter 4

Spectral Analysis

R code Exa 4.1.2 Linear combination of sinusoids

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

```r
F_lambda <- function(lambda, omega, sigma2) {
  sapply(lambda, function(l) {
    sum(sigma2 * (0.5 * (l >= -omega & l < omega) + 1.0 * (l >= omega)))
  })
}
lambda <- seq(-pi, pi, length.out = 1000)
F_values <- F_lambda(lambda, omega, sigma2)
df_F <- data.frame(Lambda = lambda, F_Lambda = F_values)
ggplot(df_F, aes(x = Lambda, y = F_Lambda)) +
  geom_step() +
  ggtitle("Spectral Distribution Function F(  )") +
  xlab("  ") +
  ylab("F(  )") +
  theme_minimal()
```

R code Exa 4.1.4 Spectral density of AR 1

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

```
ggplot(df_spectral, aes(x = Lambda, y = SpectralDensity)) +
  geom_line() +
  ggtitle("Spectral Density") +
  xlab("  ") +
  ylab("Spectral Density") +
  theme_minimal()
# Figure 4-4
phi <- -0.7
sigma2 <- 1
density <- function(lambda, phi, sigma2) {
  1 / (2 * pi) * sigma2 / (1 + phi^2 - 2 * phi * cos(lambda))
}
lambda <- seq(0, pi, length.out = 1000)
values <- density(lambda, phi, sigma2)
df_spectral <- data.frame(Lambda = lambda, SpectralDensity = values)
ggplot(df_spectral, aes(x = Lambda, y = SpectralDensity)) +
  geom_line() +
  ggtitle("Spectral Density") +
  xlab("  ") +
  ylab("Spectral Density") +
  theme_minimal()
# Figure 4-5
phi <- 0.7
ar_process <- arima.sim(model = list(ar = c(phi)), n = n)
acf(ar_process, main = "ACF of AR(1) Process")
# Figure 4-6
phi <- -0.7
ar_process <- arima.sim(model = list(ar = c(phi)), n = n)
acf(ar_process, main = "ACF of AR(1) Process")
```

R code Exa 4.1.5 Spectral density of MA 1

```
# 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)) +
  geom_line() +
  ggtitle("Spectral Density of MA(1) Process") +
  xlab(expression(lambda)) +
  ylab(expression(f(lambda))) +
  theme_minimal()
```

R code Exa 4.2.2 Sunspot numbers spectral density

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

```
  scale_x_continuous(limits = c(0, pi)) +
  labs(
    x = expression(lambda),
    y = expression(hat(f)(lambda)),
    title = "Spectral Density Estimate"
  ) +
  theme_minimal()
# Figure 4-11
weights <- c(1/15, 2/15, 3/15, 3/15, 3/15, 2/15, 1/15)
smoothed_spec <- stats::filter(spec, filter=weights, sides=2)
df <- data.frame(freq = freq, smoothed_spec = smoothed_spec)
ggplot(df, aes(x = freq, y = smoothed_spec)) +
  geom_line() +
  scale_x_continuous(limits = c(0, pi)) +
  labs(
    x = expression(lambda),
    y = expression(hat(f)(lambda)),
    title = "Spectral Density Estimate"
  ) +
  theme_minimal()
```

R code Exa 4.4.1 Spectral density of AR 2

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

```r
}
q <- 10
lambda <- seq(0, pi, length.out = 1000)
D_10 <- sapply(lambda, D_q, q = q)
df <- data.frame(lambda = lambda, D_10 = D_10)
ggplot(df, aes(x = lambda, y = D_10)) +
  geom_line() +
  labs(
    x = expression(lambda),
    y = expression(D[10](lambda)),
    title = "Transfer Function D[10](lambda) for Simple Moving-Average Filter"
  ) +
  theme_minimal()
# Figure 4-13
ideal_low_pass <- function(lambda, wc) {
  ifelse(abs(lambda) <= wc, 1, 0)
}
wc <- pi / 4
q_values <- c(2, 10)
ideal_values <- ideal_low_pass(lambda, wc)
D_2_values <- sapply(lambda, D_q, q = 2)
D_10_values <- sapply(lambda, D_q, q = 10)
df <- data.frame(
  lambda = rep(lambda, 3),
  value = c(ideal_values, D_2_values, D_10_values),
  type = factor(rep(c("Ideal", "q = 2", "q = 10"), each = length(lambda)))
)
ggplot(df, aes(x = lambda, y = value, color = type)) +
  geom_line() +
  labs(
    x = expression(lambda),
    y = "Transfer Function",
    title = "Transfer Functions: Ideal Low-Pass Filter and Truncated Fourier Approximations"
  ) +
```

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

Chapter 5

Modeling and Forecasting with ARMA Processes

R code Exa 5.1.1 The Dow Jones Utilities Index

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

```r
print(sample_autocovariance)
print(ar_coefficient)
```

R code Exa 5.1.2 MA 1 model forecasting

```r
# Page No. 128
library(forecast)
library(tseries)
oshorts<- read.csv("OSHORTS.TSM", header = FALSE)
colnames(oshorts)[1]<- "overshorts"
ots <- ts(oshorts$overshorts)
rho_1 <- acf(ots, plot=FALSE)$acf[2]
gamma <- acf(ots, plot = FALSE, type = 'covariance')$acf[1]

if (abs(rho_1) > 0.5) {
  theta_hat <- rho_1/abs(rho_1)
} else {
  theta_hat <- (rho_1) * sqrt(4 * rho_1^2 - 4 * rho_1) / (2 * abs(rho_1))
}
sigma2_hat <- gamma / (1 + theta_hat^2)
cat("Estimated theta_hat:", theta_hat, "\n")
cat("Estimated sigma2_hat:", sigma2_hat, "\n")
```

R code Exa 5.1.3 Dow jones utilities index using burg model

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

```r
colnames(dow)[1] <- "jones"
time_series <- ts(dow$jones)
Y_t <- diff(time_series, lag=1)
ar_order <- 1
burg_model <- burg(Y_t, ar_order)
ar_param <- burg_model$phi
stderror <- (burg_model$se.phi)
aicc <- burg_model$aicc
cat("AR(1) model parameter:", ar_param, "\n")
cat("AICC:", aicc, "\n")
find_conf <- function(param, stderr){
  low <- param - (stderr*1.96)
  high <- param + (stderr*1.96)
  x <- c(low, high)
  return (x)
}
confs <- find_conf(ar_param,stderror)
cat("95% Confidence Bounds: ",confs)
```

R code Exa 5.1.4 Modeling on Lake data

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

```
pacf(X_t, main = "PACF")
ar_order <-2

# Burg model
burg_model <- burg(X_t, ar_order)
arb_param <- burg_model$phi
stderr <- (burg_model$se.phi)
aicc <- burg_model$aicc
conf_lower <- arb_param - (stderr*1.96)
conf_upper <- arb_param + (stderr*1.96)
print(" For burg model: ")
cat("AR(1) model parameter:", arb_param, "\n")
cat("AICC:", aicc, "\n")
cat("95% Confidence Bounds: (", conf_lower, ", ",
    conf_upper, ")\n")

# Yule walker model
yw_model <- yw(X_t, ar_order)
ary_param <- yw_model$phi
stderr <- (yw_model$se.phi)
aicc <- yw_model$aicc
conf_lower <- ary_param - (stderr*1.96)
conf_upper <- ary_param + (stderr*1.96)
print(" For yule walker model: ")
cat("AR(1) model parameter:", ary_param, "\n")
cat("AICC:", aicc, "\n")
cat("95% Confidence Bounds: (", conf_lower, ", ",
    conf_upper, ")\n")
```

R code Exa 5.1.5 Estimations on Dow jones utilities index

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

```
library(itsmr)
dow<- read.csv("DOWJ.TSM", header = FALSE)
colnames(dow)[1] <- "jones"
time_series <- ts(dow$jones)
Y_t <- diff(time_series, lag=1)
ma_order <- 2
inno_model <- ia(Y_t, ma_order, m = 17)
ma_param <- inno_model$theta
stderr <- (inno_model$se.theta)
aicc <- inno_model$aicc
stddev_1 <- ma_param[1]/(1.96*stderr[1])
stddev_2 <- ma_param[2]/(1.96*stderr[2])
wnvar <- inno_model$sigma2
cat("MA(2) model parameter:", ma_param, "\n")
cat("AICC:", aicc, "\n")
print("Standard deviations for first two MA
    parameters:")
print(stddev_1);print(stddev_2)
cat("White noise variance: ", wnvar)
```

R code Exa 5.1.6 Estimations on Lake data

```
# Page No. 137
library(itsmr)
library(tseries)
huron<- read.csv("LAKE.TSM", header = FALSE)
colnames(huron)[1] <- 'water'
Y_t <- ts(huron$water)
X_t <- Y_t - mean(Y_t)
arma_model <- arma(X_t, p=1, q=1)
ma_param <- arma_model$theta
ar_param <- arma_model$phi
stderr_phi <- arma_model$se.phi
stderr_theta <- arma_model$se.theta
aicc <- arma_model$aicc
```

```r
stddev_phi <- ar_param/(1.96*stderr_phi)
stddev_theta <- ma_param/(1.96*stderr_theta)
cat("Estimated AR coefficient: ", ar_param, "\n")
cat("Estimated MA coefficient: ", ma_param, "\n")
cat("AICC: ", aicc, "\n")
cat(" Standard deviations: ", stddev_phi, " ", stddev_theta)
find_conf <- function(param, stderr){
  low <- param - (stderr*1.96)
  high <- param + (stderr*1.96)
  x <- c(low, high)
  return (x)
}
conf_phi <- find_conf(ar_param, stderr_phi)
cat("95% Confidence Bounds for phi: ", conf_phi)
conf_theta <- find_conf(ma_param, stderr_theta)
cat("95% Confidence Bounds for theta: ", conf_theta)
```

R code Exa 5.1.7 Lake data analysis using Hannan algorithm

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

```
aicc <- h_model$aicc
stderr_phi <- h_model$se.phi
stderr_theta <- h_model$se.theta
stddev_phi <- ar_param/(1.96*stderr_phi)
stddev_theta <- ma_param/(1.96*stderr_theta)
cat("Estimated AR coefficient: ", ar_param, "\n")
cat("Estimated MA coefficient: ", ma_param, "\n")
cat("AICC: ", aicc, "\n")
cat(" Standard deviations, phi and theta respectively: ", stddev_phi, stddev_theta)
find_conf <- function(param, stderr){
  low <- param - (stderr*1.96)
  high <- param + (stderr*1.96)
  x <- c(low, high)
  return (x)
}
confs_phi <- find_conf(ar_param,stderr_phi)
cat("95% Confidence Bounds for phi: ",confs_phi)
confs_theta <- find_conf(ma_param,stderr_theta)
cat("95% Confidence Bounds for theta: ",confs_theta)
```

R code Exa 5.2.4 Burg and yule walker model comparison

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

```
bmodel <- burg(dow_mean_diff, p)
model <- autofit(dow_mean_diff, p=0:5, q=0:5)
aicc <- model$aicc
aicc_yw <- ywmodel$aicc
aicc_b <- bmodel$aicc
LL_yw <- aicc_yw - (2*(p+q+1)*n/(n-p-q-2))
LL_b <- aicc_b - (2*(p+q+1)*n/(n-p-q-2))
LL <- aicc - (2*(p+q+1)*n/(n-p-q-2))
b_param <- bmodel$phi
stderr <- model$se.phi
ar_param <- model$phi
find_conf <- function(param, stderr){
  low <- param - (stderr*1.96)
  high <- param + (stderr*1.96)
  x <- c(low, high)
  return (x)
}
confs <- find_conf(ar_param,stderr)

cat("Minimum AICC:",aicc,"\n")
cat("Standard error:",stderr,"\n")
cat("95% Confidence Bounds: ",confs)
cat("Log likelihood for autofit:",LL,"\n")
cat("Parameters in burg model:",b_param,"\n")
cat("Log likelihood for yule walker:",LL_yw,"\n")
cat("Log likelihood for burg:",LL_b,"\n")
```

R code Exa 5.2.5 Autofit on Lake data

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

```
colnames(hudson)[1]<- "level"
Y_t <- ts(hudson$level)
X_t <- Y_t - mean(Y_t)
arma_model <- autofit(X_t, p=0:5, q=0:5)
aicc <- arma_model$aicc
ar_param <- arma_model$phi
ma_param <- arma_model$theta
stderr_phi <- arma_model$se.phi
stderr_theta <- arma_model$se.theta
stddev_phi <- ar_param/(1.96*stderr_phi)
stddev_theta <- ma_param/(1.96*stderr_theta)
find_conf <- function(param, stderr){
  low <- param - (stderr*1.96)
  high <- param + (stderr*1.96)
  x <- c(low, high)
  return (x)
}
confs_phi <- find_conf(ar_param,stderr_phi)
confs_theta <- find_conf(ma_param,stderr_theta)
cat("AICC:",aicc,"\n")
cat("AR Parameter:",ar_param,"\n")
cat("MA Parameter",ma_param,"\n")
cat("Standard deviations for phi and theta:",stddev_phi,stddev_theta,"\n")
print("95% Confidence intervals:")
cat("for phi:",confs_phi,"\n")
cat("for theta:",confs_theta,"\n")
```

R code Exa 5.4.1 Forecasts on overshorts data

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

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

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

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

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

R code Exa 5.5.2 AICC based model selection

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

Chapter 6

Nonstationary and Seasonal time series models

R code Exa 6.1.1 ARIMA 1 1 0 Process

```
# Page No. 159
# Answer may vary due to randomization
library(forecast)
library(ggplot2)
phi <- 0.8
sigma2 <- 1
n <- 200
set.seed(123)
Xt <- arima.sim(model = list(order = c(1,1,0), ar =
   phi), n = n, sd = sqrt(sigma2))
# Figure 6-1
autoplot(Xt) +
  ggtitle("ARIMA(1,1,0)") +
  geom_point()+
  xlab("Time") +
  ylab("Xt") +
  theme_minimal()
# Figure 6-2
acf_plot <- ggAcf(Xt) +
```

```
  ggtitle("Sample ACF") +
  theme_minimal()
print(acf_plot)
# Figure 6-3
pacf(Xt, main ="Sample PACF")
# Figure 6-4
Yt <- diff(Xt)
plot(Yt)
```

R code Exa 6.2.1 Burg model on Australian wine data

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

R code Exa 6.2.2 Autofit for minimum AICC model

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

R code Exa 6.3.1 Test statistic on simulated data

```
# Page no. 171
# Answer may vary due to randomization
library(forecast)
library(tseries)
phi <- 0.8
sigma2 <- 1
n <- 200
set.seed(123)
X0 <- 0
Xt <- arima.sim(model = list(order = c(1,1,0), ar =
   phi), n = n, sd = sigma2)
Xt <- c(X0, Xt)
dXt <- diff(Xt)
Xt_lag1 <- lag(Xt, 1)
dXt_lag1 <- lag(dXt, 1)
```

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

R code Exa 6.3.2 Model parameters for overshorts data

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

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

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

R code Exa 6.5.2 ACF of seasonal MA model

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

R code Exa 6.5.3 ACF of seasonal AR model

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

R code Exa 6.5.4 ACF of monthly accidental deaths data

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

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

R code Exa 6.5.5 Forecasting monthly accidental deaths

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

R code Exa 6.6.1 GLS based Model parameter estimation

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

```
colnames(oshorts)[1] <- "overshorts"
oshorts$time <- seq(1,length(oshorts$overshorts))
ots <- ts(oshorts$overshorts)
ots <- ots-mean(ots)
oshorts$overshorts <- oshorts$overshorts-mean(oshorts$overshorts)
a <- autofit(ots, p=0, q=1)
print(a$theta)
cat("OLS beta:",mean(oshorts$overshorts))
acv <- acf(oshorts$overshorts,type = 'covariance',plot=FALSE)
cat("Estimator for beta: ",acv$acf[1]/length(ots))
model_formula <- overshorts ~ time
gls_model <- gls(model_formula, data = oshorts)
summary(gls_model)
```

R code Exa 6.6.2 Model parameters estimation for Lake data

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

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

R code Exa 6.6.3 Seat belt legislation

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

```
      xlab = "Time", ylab = "Value", type = "o-")
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
# Page no. 199
# Answer may vary due to randomization
alpha0 <- 1
alpha1 <- 0.5
n <- 1000
set.seed(123)
epsilon <- rnorm(n)
sigma2 <- numeric(n)
y <- numeric(n)
for (t in 2:n) {
  sigma2[t] <- alpha0 + alpha1 * y[t-1]^2
  y[t] <- sqrt(sigma2[t]) * epsilon[t]
}
plot(y, type = "l", main = "Simulated ARCH(1)
   Process", xlab = "Time", ylab = "Value")
acf(y)
```

R code Exa 7.2.2 Fitting GARCH models to stock data

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

R code Exa 7.2.3 Fitting ARMA Models Driven by GARCH Noise

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

```
library(forecast)
library(tseries)
library(rugarch)
sunspot<- read.csv("SUNSPOTS.TSM")
colnames(sunspot)[1]<- "spots"
sunspots<- ts(sunspot$spots)
sunspots_mean_corrected <- sunspots - mean(sunspots, na.rm = TRUE)
fit_arima <- Arima(sunspots_mean_corrected, order = c(4,0,3))
print(fit_arima)
residuals_arima <- fit_arima$residuals
p <- 1
q <- 1
spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(p, q)),
                   mean.model = list(armaOrder = c(4, 3), include.mean = TRUE),
                   distribution.model = "norm")
fit_garch <- ugarchfit(spec = spec, data = residuals_arima)
print(fit_garch)
n <- as.numeric(length(sunspots_mean_corrected))
aicc <- (((-2)*(fit_garch@fit$LLH))*(n/(n-p)))+ (((p+q+2)*(2*n))/(n-p-q-2))
print(paste("AICC value for the GARCH model:", aicc))
print("Parameters of the GARCH(1,1) model:")
print(coef(fit_garch))
```

R code Exa 7.5.1 Brownian motion

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

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

R code Exa 7.5.2 Poisson process

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

R code Exa 7.5.3 Compound Poisson Process

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

```
jump_sizes <- rnorm(length(jump_times), mean = mu,
   sd = sigma)
X_t <- cumsum(jump_sizes)
jump_times <- c(0, jump_times)
X_t <- c(0, X_t)
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

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

```

```
pcs <- separate(pc, col = 1, into = c("dow", "aus"),
     sep = "\\s+")
dowjones1 <- ts(as.numeric(pcs$dow))
Aus1 <- ts(as.numeric(pcs$aus))
acf(dowjones1, main = "Series 1")
acf(Aus1, main = "Series 2")
ccf1 <- ccf(dowjones1, Aus1,plot = FALSE)
positive_lag1 <- ccf1$lag >= 0
plot(ccf1$lag[positive_lag1], ccf1$acf[positive_lag1
   ], type = "h",
     main = "Series 1 * Series 2",
     xlab = "Lag", ylab = "CCF")
abline(h = 0)
ccf2 <- ccf(Aus1,dowjones1,plot = FALSE)
positive_lag2 <- ccf2$lag >= 0
plot(ccf2$lag[positive_lag2], ccf2$acf[positive_lag2
   ], type = "h",
     main = "Series 2 * Series 1",
     xlab = "Lag", ylab = "CCF")
abline(h = 0)
plot(lag(dowjones1, -1), Aus1, main="Scatterplot",
     xlab="Lagged TS1", ylab="TS2", pch=19)
```

R code Exa 8.1.2 Sales with a leading indicator

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

```
lst <- ts(ls2)
lst <- diff(lst)
par(mfrow = c(2, 2))
acf(lst[, 2], main = "Series 1")
acf(lst[, 1], main = "Series 2")
ccf1 <- ccf(lst[, 1], lst[, 2],plot = FALSE)
positive_lag1 <- ccf1$lag >= 0
plot(ccf1$lag[positive_lag1], ccf1$acf[positive_lag1
    ], type = "h",
     main = "Series 2 * Series 1",
     xlab = "Lag", ylab = "CCF")
abline(h = 0)
ccf2 <- ccf(lst[,2],lst[,1],plot = FALSE)
positive_lag2 <- ccf2$lag >= 0
plot(ccf2$lag[positive_lag2], ccf2$acf[positive_lag2
    ], type = "h",
     main = "Series 1 * Series 2",
     xlab = "Lag", ylab = "CCF")
abline(h = 0)
```

R code Exa 8.3.1 Sample correlations

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

```
    ], type = "h",
      main = "Series 1 * Series 2",
      xlab = "Lag", ylab = "CCF")
abline(h = 0)
ccf2 <- ccf(Ets[,2],Ets[,1],plot = FALSE)
positive_lag2 <- ccf2$lag >= 0
plot(ccf2$lag[positive_lag2], ccf2$acf[positive_lag2
    ], type = "h",
      main = "Series 2 * Series 1",
      xlab = "Lag", ylab = "CCF")
abline(h = 0)
```

R code Exa 8.6.1 Multivariate models fitted on stock data

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

R code Exa 8.6.2 Multivariate models fitted on sales data

```
# Page No. 249
```

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

R code Exa 8.6.3 VAR 1 model on stock data

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

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

Chapter 9

State Space Models

R code Exa 9.2.1 Random walk plus noise model

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

R code Exa 9.5.2 International airline passengers

```r
# Page No. 278
```

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

R code Exa 9.8.3 Polio in the USA

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

```
      y = "Polio cases") +
  theme_minimal()
polio$Month <- 1:length(polio$pol)
polio <- polio %>%
  mutate(
    t = Month,
    u1 = 1,
    u2 = t / 1000,
    u3 = cos(2 * pi * t / 12),
    u4 = sin(2 * pi * t / 12),
    u5 = cos(2 * pi * t / 6),
    u6 = sin(2 * pi * t / 6)
  )
model <- lm(pol ~ u1 + u2 + u3 + u4 + u5 + u6, data
   = polio)
polio$Trend <- fitted(model)
ggplot(polio, aes(x = Month)) +
  geom_point(aes(y = pol, color = "Actual Cases")) +
  geom_line(aes(y = Trend, color = "Trend Estimate")
    ) +
  labs(
    title = "Trend Estimate for Monthly U.S. Polio
       Cases",
    x = "Month",
    y = "Number of Cases",
    color = "Legend"
  ) +
  scale_color_manual(values = c("Actual Cases" = "
    blue", "Trend Estimate" = "red")) +
  theme_minimal()
```

R code Exa 9.8.7 Goals Scored by England Against Scotland

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

```
    springer-extras/zip/2002/978-0-387-21657-7.zip
# Answer varies due to inadequate data
library(ggplot2)
library(tidyr)
library(itsmr)
goals <- read.table("GOALS.TSM", header = FALSE)
colnames(goals)[1] <- "goal"
colnames(goals)[2] <-"Year"
# Figure 9-8
ggplot(goals, aes(x = Year, y = goal)) +
  geom_point() +
  geom_line(col='blue') +
  labs(title = "Goals by England",
       x = "Years",
       y = "Goals") +
  theme_minimal()
# Figure 9-9
ggplot(na.omit(goals), aes(x = factor(goal))) +
  geom_bar() +
  xlab("Goals") +
  ylab("Count") +
  ggtitle("Histogram of Goals") +
  theme_minimal()

data <- na.omit(goals)
delta_hat <- 0.844
alpha_0 <- 0.154
lambda_0 <- delta_hat / (1 - delta_hat)
n <- nrow(data)
alpha <- numeric(n);lambda <- numeric(n);pred <-
    numeric(n)
alpha[1] <- alpha_0
lambda[1] <- lambda_0
for (t in 2:n) {
  alpha[t] <- alpha[t-1] + delta_hat * (data$goal[t
     -1] - alpha[t-1])
  lambda[t] <- lambda[t-1] + delta_hat * (1 - lambda
     [t-1])
```

```r
  pred[t] <- alpha[t] / (1 + lambda[t])
}

ggplot(data.frame(Time = data$Year, pred = pred), aes(x = Time, y = pred)) +
  geom_line(color = "blue") +
  geom_point(data = data, aes(x = Year , y = goal), color = "red") +
  xlab("Year") +
  ylab("Goals") +
  ggtitle("One-Step Predictors for Goals Data") +
  theme_minimal()
```

Chapter 10

Forecasting Techniques

R code Exa 10.1.1 Predicted deaths by ARAR algorithm

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

R code Exa 10.2.1 Holt Winters non seasonal forecast

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

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

R code Exa 10.3.1 Holt Winters seasonal forecast

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

Chapter 11

Further Topics

R code Exa 11.4.1 Annual Minimum Water Levels in the Nile

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