**Exercise Number – 1**

**K-Means Clustering**

**Q1. K-mean Clustering using Mall\_Customers dataset.**

**Aim:** To use the Mall\_Customers dataset with the K-Means Clustering Algorithm and visualise the clusters

**Code:**

library(arules)

library(dplyr)

#k means clustering

df2 = read.csv("D:\\Sem6\\DVP\\ELA\\Assessment4\\Mall\_Customers.csv")

df2 = df2[4:5]

df2

library(cluster)

set.seed(5000)

wcss = vector()

wcss

for(i in 1:50)

wcss[i] = sum(kmeans(df2, i)$withinss)

plot(1:50, wcss, type = 'b') #5 cluster reqd

kmeans = kmeans(x = df2, centers = 5)

y\_kmeans = kmeans$cluster

z=clusplot(df2, y\_kmeans,main = "20BDS0162")

z

**Output:**

Chart

Description automatically generated with medium confidence

Diagram, schematic

Description automatically generated

**Result:**

K-Means Clustering for the dataset Mall Customers was completed successfully. Based on the Annual Income column, I had first put the cluster centroid at 5000. (4th). I used the algorithm to produce five clusters. Following that, all the clusters are shown, and it is also indicated which data points belong to particular clusters and how far away they are from each other.

**Exercise Number - 2**

**Market Basket Analysis**

**Q1. Market-Basket Data analysis Visualization**

**Aim:** To perform Market-Basket-Data-Analysis Visualization using Apriori Algorithm and visualize the results

**Code:**

install.packages("dplyr")

install.packages("arules")

library(arules)

library(dplyr)

df1 = read.csv("D:\\Sem6\\DVP\\ELA\\Assessment4\\Market\_Basket\_Optimisation.csv", header = FALSE)

summary(df1)

dim(df1)

str(df1)

#sparse matrix

df1 = read.transactions(file = "D:\\Sem6\\DVP\\ELA\\Assessment4\\Market\_Basket\_Optimisation.csv",

sep = ",",

rm.duplicates = T)

summary(df1)

itemFrequencyPlot(x = df1, topN = 10,main = "20BDS0162",col = "red", border = "blue")

#apriori algo

rules = apriori(data = df1,

parameter = list(support = 0.004,

confidence = 0.2))

#visualizing

inspect(sort(rules, by = 'lift')[1:10])

**Output:**

Chart, bar chart

Description automatically generated

**Result:**

Calculated the necessary support and confidence that properly suit the provided dataset in order to successfully use the Apriori algorithm and visualize the results. According to the findings, one can predict what a person would purchase in total while visiting the market based on the things they initially choose.

**Exercise Number – 3**

**Text Analytics**

**Q3. Text Analytics on Shakespeare’s plays.**

**Aim:** To perform text analytics for the given shakespeare.rda file using R

**Code:**

#TEXT ALANYTICS ON SHAKESPEARE

#1. load shakespeare.rda into r environment

load("D:\\Sem6\\DVP\\ELA\\Assessment4\\shakespeare.rda")

glimpse(shakespeare)

#2. Pipe the shakespeare data frame to the next line

# Use count to find out how many titles/types there are

shakespeare %>%

count(title, type)

#3. Load tidytext/ tidyverse

library(tidytext)

library(dplyr)

library(tidyverse)

#4. create an object tidy\_shakespeare

# Group by the titles of the plays

# Define a new column line number

# Transform the non-tidy text data to tidy text data

tidy\_shakespeare <- shakespeare %>%

group\_by(title) %>%

mutate(linenumber = row\_number()) %>%

unnest\_tokens(word, text) %>%

ungroup()

#5. Pipe the tidy Shakespeare data frame to the next line

# Use count to find out how many times each word is used

tidy\_shakespeare %>%

count(word, sort = TRUE)

#6. Sentiment analysis of tidy\_shakespeare assign to object shakespeare\_sentiment

# Implement sentiment analysis with the "bing" lexicon

shakespeare\_sentiment <- tidy\_shakespeare %>%

inner\_join(get\_sentiments("bing"),by="word")

#7. shakespeare\_sentiment

# Find how many positive/negative words each play has

shakespeare\_sentiment %>%

count(title, sentiment)

#8. Tragedy or comedy from tidy\_shakespeare assign to sentiment\_counts

# Implement sentiment analysis using the "bing" lexicon

# Count the number of words by title, type, and sentiment

sentiment\_counts <- tidy\_shakespeare %>%

inner\_join(get\_sentiments("bing")) %>%

count(word, title, sentiment)

#9. from sentiment\_counts

# Group by the titles of the plays

# Find the total number of words in each play

# Calculate the number of words divided by the total

# Filter the results for only negative sentiment then arrange percentages in ASC order

sentiment\_counts %>%

group\_by(title) %>%

mutate(total = sum(n),

percent = n / total) %>%

filter(sentiment == "negative") %>%

arrange(percent)

#10 Most common positive and negative words and assign to word\_could

# Implement sentiment analysis using the "bing" lexicon

# Count by word and sentiment

word\_count <- tidy\_shakespeare %>%

inner\_join(get\_sentiments("bing"), by = "word") %>%

count(word, sentiment, sort = TRUE)

word\_count1 <- select(word\_count,1,3)

word\_count1

colnames(word\_count1) <- c("word","freq")

#install.packages("wordcloud2")

library(wordcloud2)

wordcloud2(data=word\_count1, size=1.4, color='random-dark')

#11. extract the top 10 words from word\_counts and assign to top\_words

# Group by sentiment

# Take the top 10 for each sentiment and ungroup it

# Make word a factor in order of n

top\_words <- word\_count %>%

group\_by(sentiment) %>%

top\_n(10, n) %>%

ungroup() %>%

mutate(word = factor(word, levels = rev(unique(word))))

#12 Use aes() to put words on the x-axis and n on the y-axis

# Make a bar chart with geom\_col()

# facet\_wrap for sentiments and apply scales as free

#Move x to y and y to x

library(ggplot2)

ggplot(top\_words, aes(x = n, y = word, fill = sentiment)) +

geom\_col() +

facet\_wrap(vars(sentiment))

install.packages("wordcloud")

library(wordcloud)

top\_words

set.seed(100)

top\_words <- word\_count %>%

group\_by(sentiment) %>%

top\_n(50, n) %>%

ungroup() %>%

mutate(word = factor(word, levels = rev(unique(word))))

wordcloud(words = top\_words$word, freq = top\_words$n, random.order = TRUE,

colors = brewer.pal(8,"Dark2"))

wordcloud(words = top\_words$word, freq = top\_words$n, random.order = FALSE,

colors = brewer.pal(12,"Paired"))

**Output:**



A picture containing diagram

Description automatically generatedChart, funnel chart

Description automatically generated

**Result:**

We successfully performed text analytics on Shakespeare's plays using R. Sentiment analysis is performed on the tidy data using the "bing" lexicon. The code also extracts the most common positive and negative words and creates a bar chart and a word cloud for the top words. The final result shows the percentage of negative words in each play and arranges them in ascending order.

**Exercise Number – 4**

**Time Series Analysis**

**Q4. Time Series Analysis on the AirPassengers dataset**

**Aim:** To perform time series analysis for AirPassengers dataset in R and predict the values for the next 10 years

**Code:**

data("AirPassengers")

AirPassengers

str(AirPassengers)

class(AirPassengers)

sum(is.na(AirPassengers))

start(AirPassengers)

end(AirPassengers)

frequency(AirPassengers)

summary(AirPassengers)

plot.ts(AirPassengers, main = "20BDS0162")

abline(reg =

lm(AirPassengers~time(AirPassengers)), main = "20BDS0162")

cycle(AirPassengers)

#Make the data is Stationary

#1. Mean should be constant to time

#2. Var should be equal

#3. Cor should be equal

par(mfrow=c(3,3))

plot(AirPassengers)

plot(log(AirPassengers)) #var to be equal

plot(diff(log(AirPassengers))) #Check for Mean - Constant

#Trend

plot(aggregate(AirPassengers, FUN=mean))

boxplot(AirPassengers~cycle(AirPassengers))

plot(decompose(AirPassengers))

#Model Identification and Estimation

# AR I MA

# p d q

acf(AirPassengers, main = "Series AirPassenegers 20BDS0162")

acf(log(AirPassengers),main = "Series log(AirPassengers) 20BDS0162")

acf(diff(log(AirPassengers)), main = "Series diff(log(AirPassengers)) 20BDS0162")#q=1

pacf(diff(log(AirPassengers)), main = "Series diff(log(AirPassengers)) 20BDS0162")#p=0

#d = 1

fit <- arima(log(AirPassengers),

order = c(0, 1, 1),

seasonal = list(order = c(0, 1, 1),

period = 12))

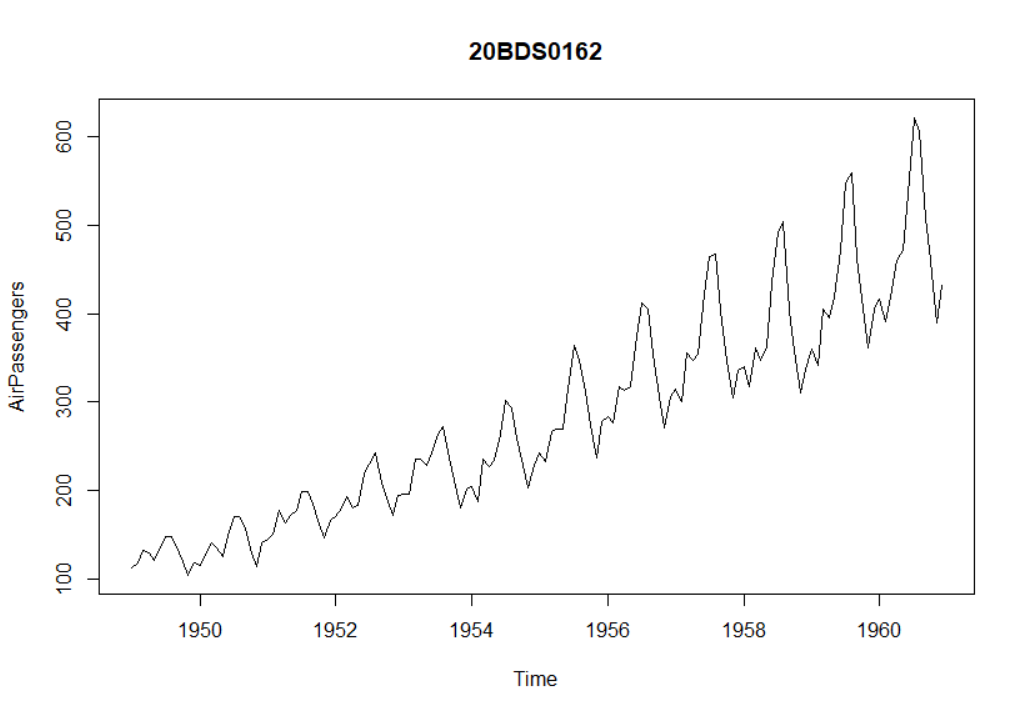
fit

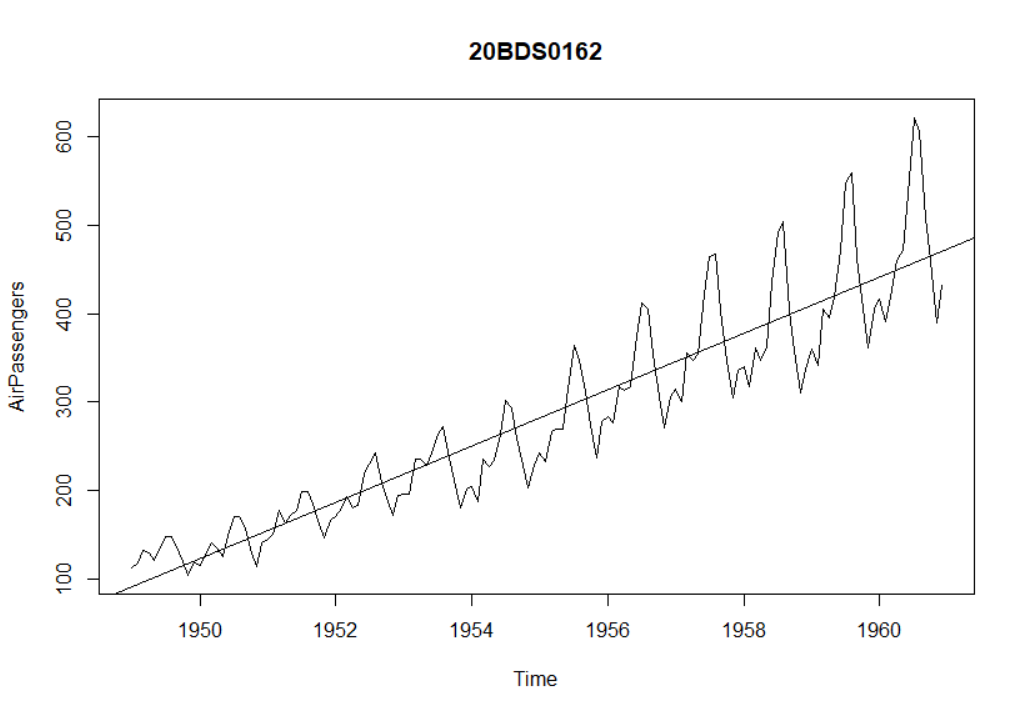
pred<- predict(fit, n.ahead = 10\*12)

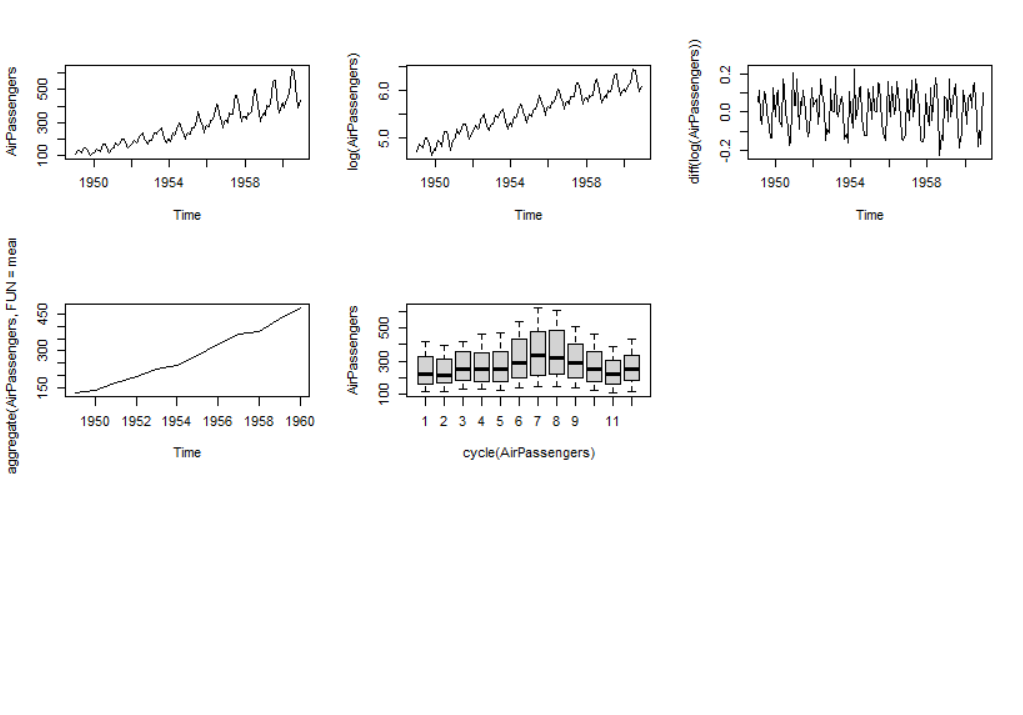
pred1<-round(2.718^pred$pred,0)

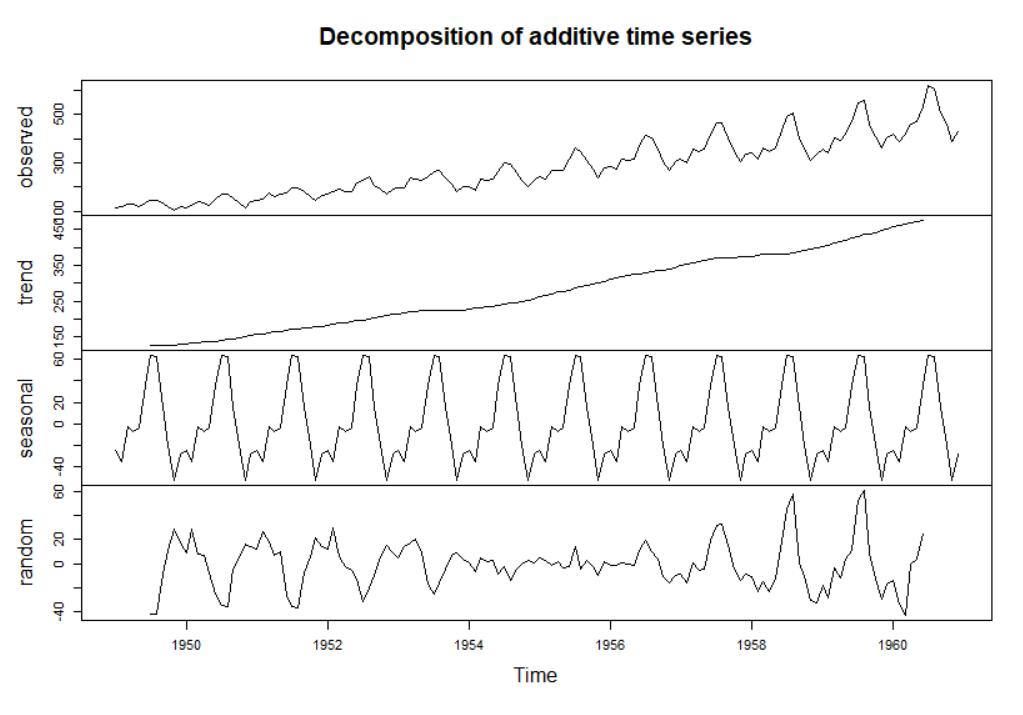
ts.plot(AirPassengers, pred1, log="y", lty=c(1,3), main = "Predicted Values 20BDS0162")

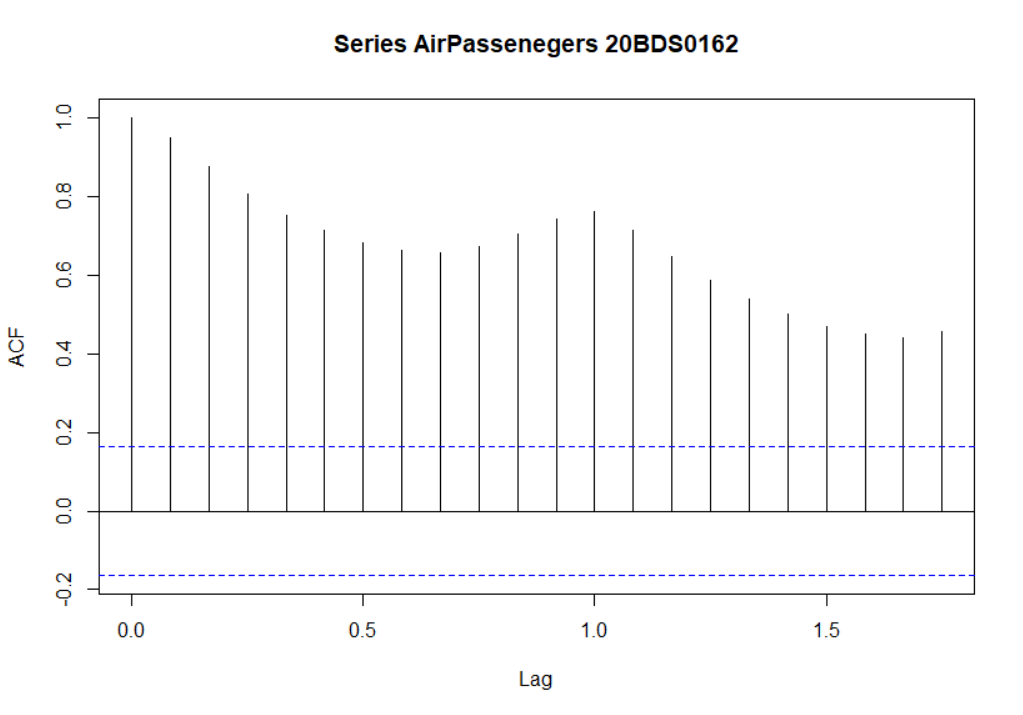
**Output:**

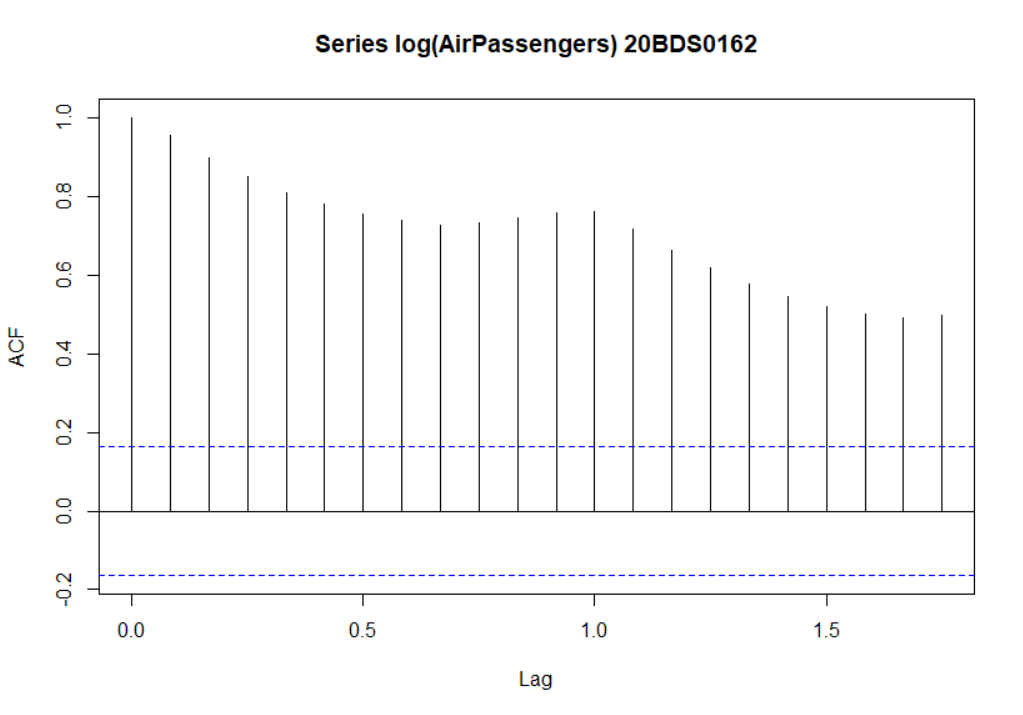


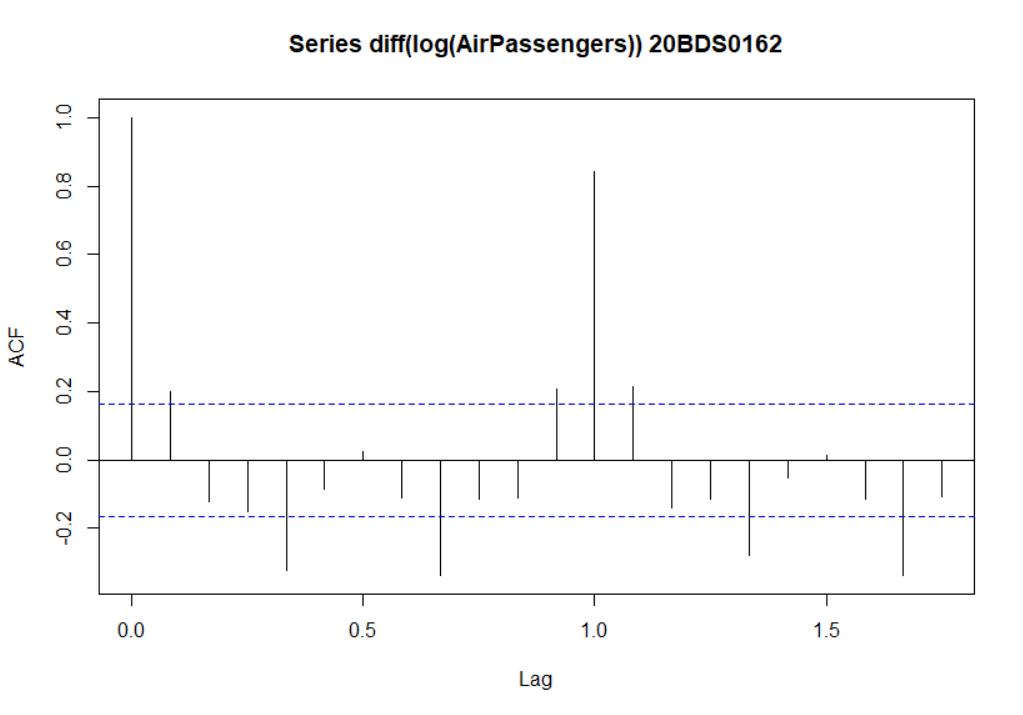


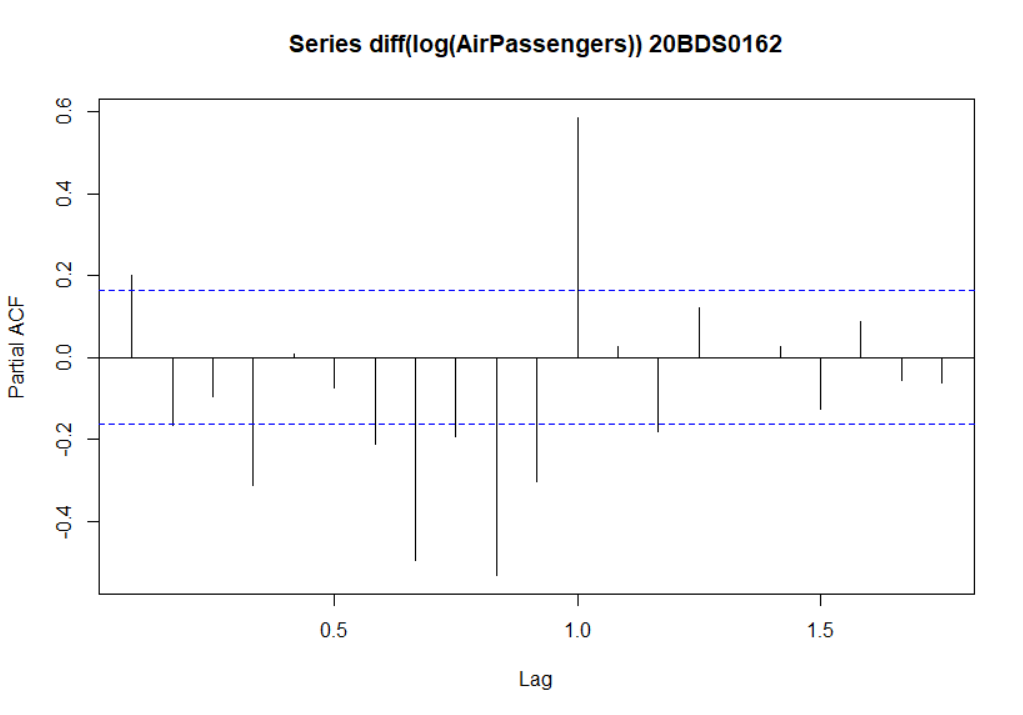


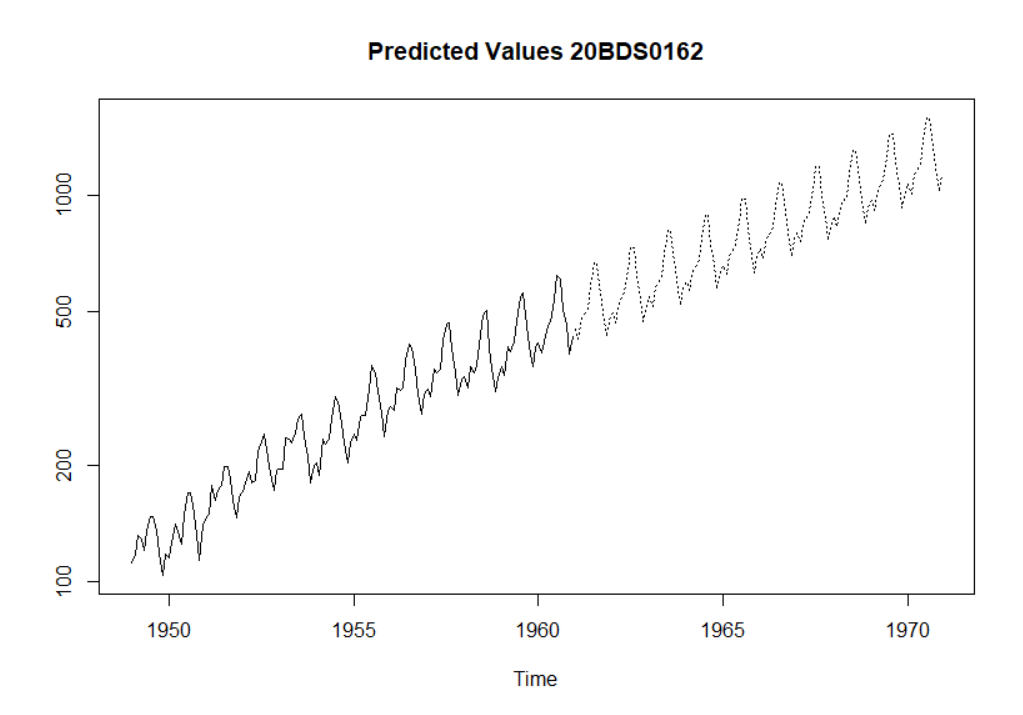












**Result:**

We successfully performed time series analysis for the AirPassengers dataset in R. We used the ARIMA model for the prediction and fitted the model to the dataset. The fitted model is used to make predictions for the next 10 years and finally plotted the predicted values graph.

**Exercise Number – 5**

**Dashboard in R**

**Q5. Create a Simple dashboard using Shiny.**

**Aim:** To create a simple dashboard in R using the Shiny library on the gapminder dataset

**Code:**

library(shiny)

library(shinydashboard)

library(datasets)

library(ggplot2)

library(dplyr)

library(gapminder)

data = gapminder

View(data)

ui <- dashboardPage(

dashboardHeader(title = "Gapminder Dashboard 20BDS0162",titleWidth = 400),

dashboardSidebar(

selectInput("year", "Choose a year:", choices = unique(data$year), multiple = TRUE, selected = "1952"),

checkboxGroupInput("continent", "Choose a continent:", choices = c(unique(data$continent)),selected = "Asia"),

sliderInput("gdpPercap", "Select GDP Per Cap Range:", min = min(data$gdpPercap), max = max(data$gdpPercap), value = c(241.1658765,113523)),

actionButton("clear\_button", "Clear Selection")

),

dashboardBody(

fluidRow(

box(plotOutput("plot1", height = 250)),

box(plotOutput("plot2", height = 250)),

box(plotOutput("plot3", height = 250)),

box(plotOutput("plot4", height = 250))

)

)

)

server <- function(input, output,session) {

filter\_data <- reactive({

data %>% filter(gdpPercap >= input$gdpPercap[1] & gdpPercap <= input$gdpPercap[2]) %>%

filter(continent %in% input$continent) %>%

filter(year %in% input$year)

})

output$plot1 <- renderPlot({

ggplot(filter\_data(), aes(

x = year,

y = gdpPercap,

color = as.factor(continent)))+

geom\_point() + labs(x = "Year", y = "GDP Per Cap")

})

output$plot2 <- renderPlot({

ggplot(filter\_data(), aes(

x = year, fill = as.factor(year)))+

geom\_bar() + facet\_wrap(~continent) +

scale\_x\_log10()

})

output$plot3 <- renderPlot({

ggplot(filter\_data(), aes(

x = year, y = gdpPercap,

color = as.factor(continent)))+

geom\_boxplot()

})

output$plot4 <- renderPlot({

ggplot(filter\_data(), aes(

x = gdpPercap))+

geom\_density(fill = "darkred")+scale\_x\_log10()

})

observeEvent(input$clear\_button, {

updateSelectInput(session, "year", selected = "1952")

updateRadioButtons(session, "continent", selected = "Asia")

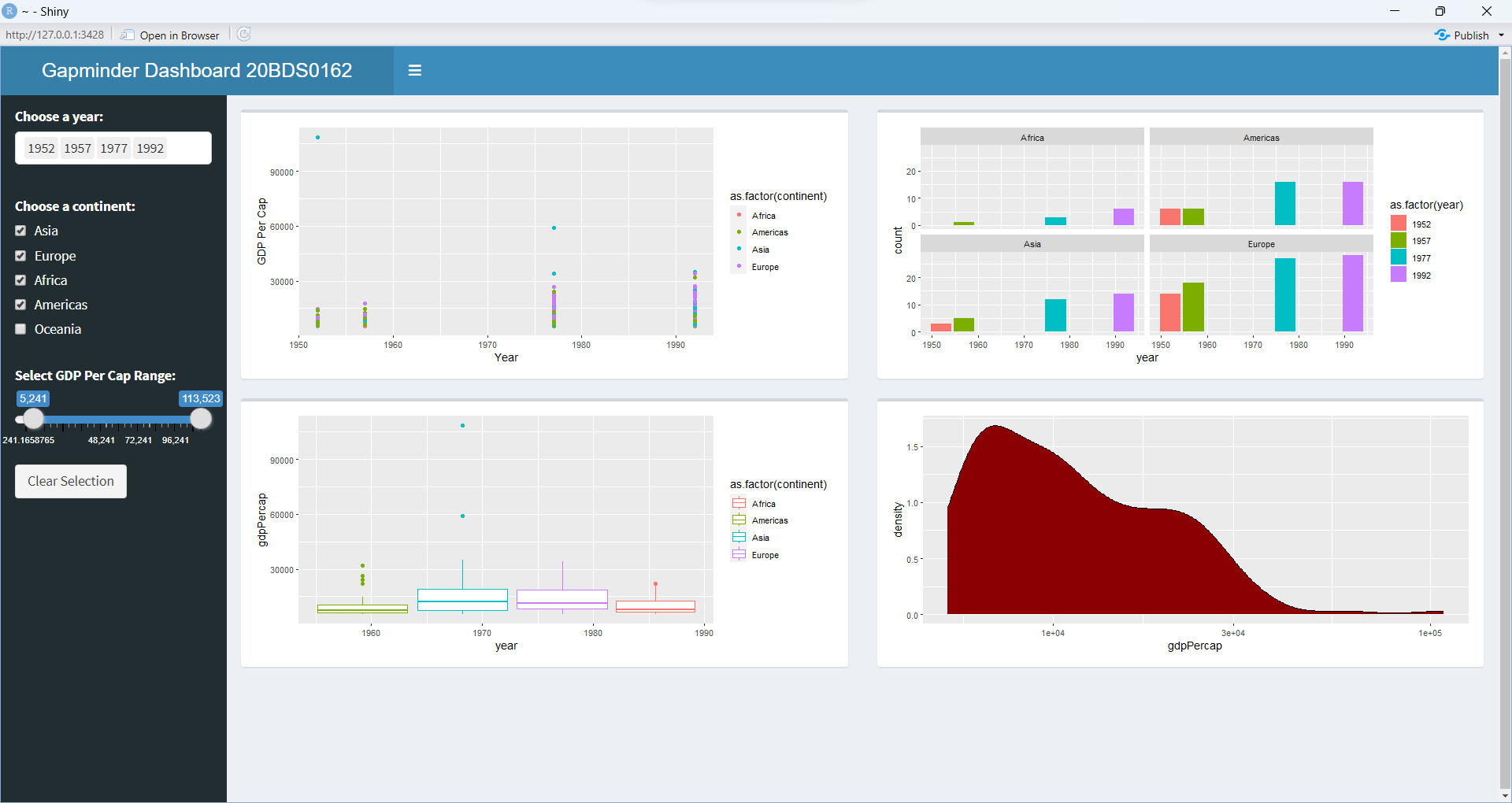
updateSliderInput(session, "gdpPercap", value = c(241.1658765,113523))

})

}

shinyApp(ui, server)

**Output:**

****

**Result:**

Successfully created a straightforward dashboard in R using the Shiny package on the gapminder dataset. The dashboard gives the option to the user to select multiple years and continents and also the range of GDP Per Cap. Also, a clear selection button is there to reset the values to default. The dashboard has totally 4 graphs.