1. Demonstrate data cleaning – missing values

```
library(tidyverse)
x <- sample(1:21, 20, replace = TRUE)
y <- sample(1:10, 20, replace = TRUE)
for(i in 1:20)
{
  a <- x[i]
  b <-  y[i]
  mtcars[a, b] = NA
}
which(is.na(mtcars))
sum(is.na(mtcars))
na.exclude(mtcars)
view(mtcars)
dispna <- apply(mtcars["disp"], 2, mean, na.rm=TRUE)
view(dispna)
newcars <- mtcars %>%
  mutate(disp = ifelse(is.na(disp), dispna, disp), )
view(newcars)
```

Output

```
> which(is.na(mtcars))
 [1]  1  10  33  37  42  48  66  69  73  76  77  85 101 105 112 115 116 136 149 16
2 170 171
[23] 174 175 193 194 196 203 206 213 239 245 261 290 298 305

> sum(is.na(mtcars))
[1] 36
```

```
> na.exclude(mtcars)
mpg cyl  disp  hp drat wt  qsec vs am gear carb
Datsun 710        22.8  4 108.0  93 3.85 2.320 18.61  1  1    4    1
Valiant          18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3  8 360.0 245 3.21 3.570 15.84  0  0    3    4
Fiat 128         32.4  4  78.7  66 4.08 2.200 19.47  1  1    4    1
Dodge Challenger 15.5  8 318.0 150 2.76 3.520 16.87  0   0    3    2
AMC Javelin      15.2  8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28       13.3  8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9        27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2    26.0  4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa     30.4  4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L   15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino     19.7  6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora    15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E       21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

2. Implement data normalization (min-max, z-score)

```
arr <- c(9.5, 6.2, 8.9, 15.2, 20.0, 10.1, 5.4, 3.2, 1.0, 22.5, 10.0, 16.0)

#min-max

minarr  <-  min(arr)

maxarr <- max(arr)

arr2 <- arr

for (i in 1:12){

  arr2[i] = round((arr[i]-minarr)/(maxarr-minarr))

}

print(arr2)


#z-score

meanarr <- mean(arr)

sdarr <- sd(arr)

for (i in 1:12){

  arr2[i] = round((arr[i]-meanarr)/sdarr, 2)

}

print(arr2)
```

Output:

```
> print(arr2)
 [1] 0 0 0 1 1 0 0 0 0 1 0 1
>
> #z-score
> meanarr <- mean(arr)
> sdarr <- sd(arr)
> for (i in 1:12){
+  arr2[i] = round((arr[i]-meanarr)/sdarr, 2)
+ }
> print(arr2)
 [1] -0.18 -0.68 -0.27 0.69  1.42 -0.09 -0.80 -1.13 -1.47  1.79 -0.10  0.81
```

3. Implement attribute subset selection for data reduc tion

```
# Install and load the leaps package (only needs to be installed once)

if (!require(leaps)) install.packages("leaps")

library(leaps)

View(as.data.frame(Titanic))

Titanic <- as.data.frame(Titanic)

sum(is.na(Titanic))

Titanic <- na.omit(Titanic)

dim(Titanic)

fwd <- regsubsets(Freq ~ ., data = Titanic, nvmax = 19, method = "forward")

bwd <- regsubsets(Freq ~ ., data = Titanic, nvmax = 19, method = "backward")

full <- regsubsets(Freq ~ ., data = Titanic, nvmax = 19)

summary(fwd)

summary(bwd)

summary(full)

coef(fwd, 3)

coef(bwd, 3)

coef(full, 3)
```

Output:

```
> summary(fwd)
Subset selection object
Call: regsubsets.formula(Freq ~ ., data = Titanic, nvmax = 19, method = "forward")
6 Variables  (and intercept)
            Forced in Forced out
Class2nd        FALSE      FALSE
Class3rd        FALSE      FALSE
ClassCrew       FALSE      FALSE
SexFemale       FALSE      FALSE
AgeAdult        FALSE      FALSE
SurvivedYes     FALSE      FALSE
```

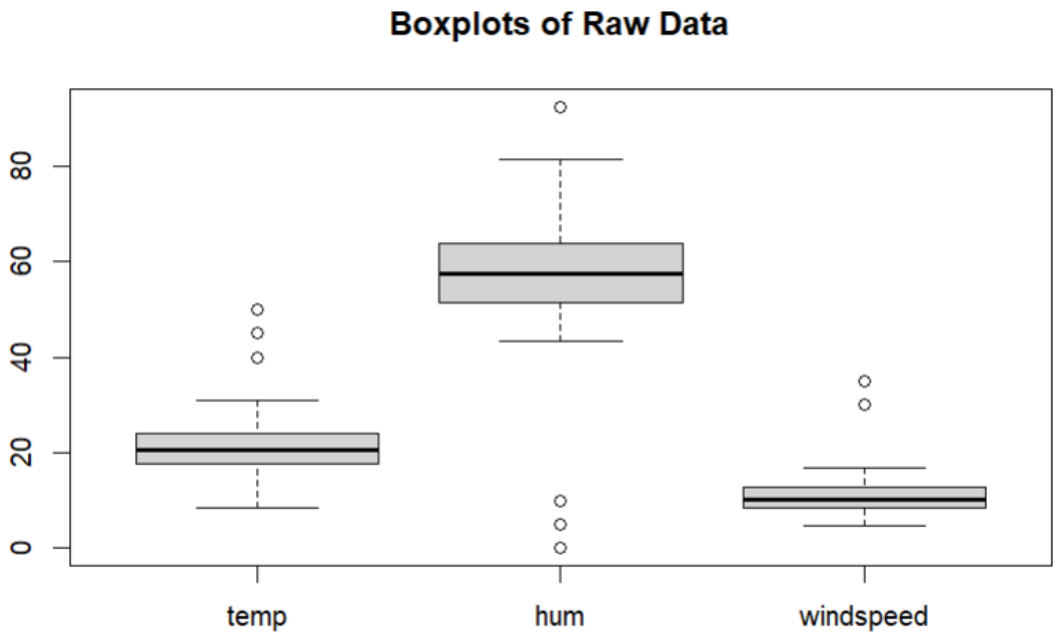
```
1 subsets of each size up to 6
         Selection Algorithm:
forward
       Class2nd Class3rd ClassCrew SexFemale AgeAdult SurvivedYes
1  ( 1 ) " "     " "     " "      " "      "*"     " "
2  ( 1 ) " "     " "     " "      "*"      "*"     " "        >   s
3  ( 1 ) " "     " "     " "      "*"      "*"     "*"        u
4  ( 1 ) " "     " "     "*"      "*"      "*"     "*"        m
5  ( 1 ) " "     "*"     "*"      "*"      "*"     "*"        m
6  ( 1 ) "*"     "*"     "*"      "*"      "*"     "*"        a
                                                              r
y(bwd) Subset
selection object
Call: regsubsets.formula(Freq ~ ., data = Titanic, nvmax = 19, method = "backward")
6 Variables  (and intercept)
                Forced in Forced out
Class2nd        FALSE        FALSE
Class3rd        FALSE      FALSE
ClassCrew       FALSE      FALSE
SexFemale       FALSE      FALSE
AgeAdult        FALSE      FALSE
SurvivedYes     FALSE      FALSE
1 subsets of each size up to 6
Selection Algorithm: backward
       Class2nd Class3rd ClassCrew SexFemale AgeAdult SurvivedYes
1  ( 1 ) " "     " "     " "      " "      "*"     " "
2  ( 1 ) " "     " "     " "      "*"      "*"     " "
3  ( 1 ) " "     " "     " "      "*"      "*"     "*"
4  ( 1 ) " "     " "     "*"      "*"      "*"     "*"
5  ( 1 ) " "     "*"     "*"      "*"      "*"     "*"
6  ( 1 ) "*"     "*"     "*"      "*"      "*"     "*"

> summary(full)
Subset selection object
Call: regsubsets.formula(Freq ~ ., data = Titanic, nvmax = 19)
6 Variables  (and intercept)
         Forced in Forced out
Class2nd        FALSE      FALSE
Class3rd        FALSE      FALSE
ClassCrew       FALSE      FALSE
SexFemale       FALSE      FALSE
```

```
AgeAdult      FALSE      FALSE
SurvivedYes    FALSE      FALSE 1
subsets of each size up to 6
Selection Algorithm: exhaustive
         Class2nd Class3rd ClassCrew SexFemale AgeAdult SurvivedYes
1  ( 1 ) " "      " "      " "       " "       "*"      " "
2  ( 1 ) " "      " "      " "       "*"       "*"      " "
3  ( 1 ) " "      " "      " "       "*"       "*"      "*"
4  ( 1 ) " "      " "      "*"       "*"       "*"      "*"
5  ( 1 ) " "      "*"      "*"       "*"       "*"      "*"
6  ( 1 ) "*"      "*"      "*"       "*"       "*"      "*"
>
> coef(fwd, 3)
(Intercept)   SexFemale    AgeAdult SurvivedYes
    70.5625    -78.8125    123.9375    -48.6875
> coef(bwd, 3)
(Intercept)   SexFemale    AgeAdult SurvivedYes
    70.5625    -78.8125    123.9375    -48.6875
> coef(full, 3)
(Intercept)   SexFemale    AgeAdult SurvivedYes
    70.5625    -78.8125    123.9375    -48.6875

>
```

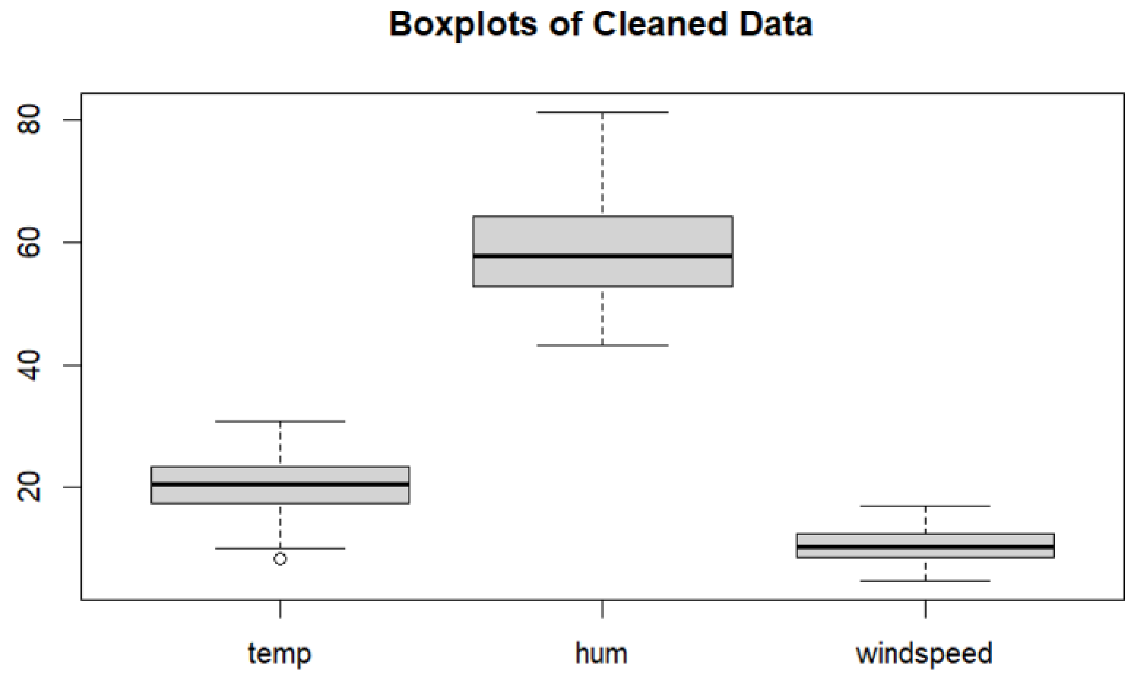
4. Demonstrate outlier detection

```
install.packages("tidyr")  # For drop_na function
library(tidyr)
set.seed(123)
day <- data.frame(
  temp = rnorm(100, mean = 20, sd = 5),       # Random temperatures
  hum = rnorm(100, mean = 60, sd = 10),       # Random humidity percentages
  windspeed = rnorm(100, mean = 10, sd = 3)    # Random wind speeds
)
# Add some outliers
day$temp[c(10, 20, 30)] <- c(40, 45, 50)      # Adding extreme temperatures
day$hum[c(15, 25, 35)] <- c(10, 5, 0)         # Adding extreme humidity values
day$windspeed[c(40, 50, 60)] <- c(25, 30, 35) # Adding extreme wind speeds

# Add some missing values
day$temp[c(5, 15)] <- NA
day$hum[c(10, 20)] <- NA
day$windspeed[c(30, 40)] <- NA

# View the first few rows of the dataset
print(head(day))
print(sum(is.na(day)))
boxplot(day[, c("temp", "hum", "windspeed")], main = "Boxplots of Raw Data")

# Handle outliers in specific columns
for(i in c("hum", "windspeed")) {
  data <- unlist(day[i])
  outliers <- boxplot.stats(data)$out
  data[data %in% outliers] <- NA
  day[i] <- data
}

# Check for missing values after handling outliers
print(sum(is.na(day)))

# Drop rows with missing values
day_clean <- drop_na(day)
boxplot(day_clean[, c("temp", "hum", "windspeed")], main = "Boxplots of Cleaned
Data")
```

Output:



Boxplots of Raw Data
0
20
40
60
80
temp
hum
windspeed



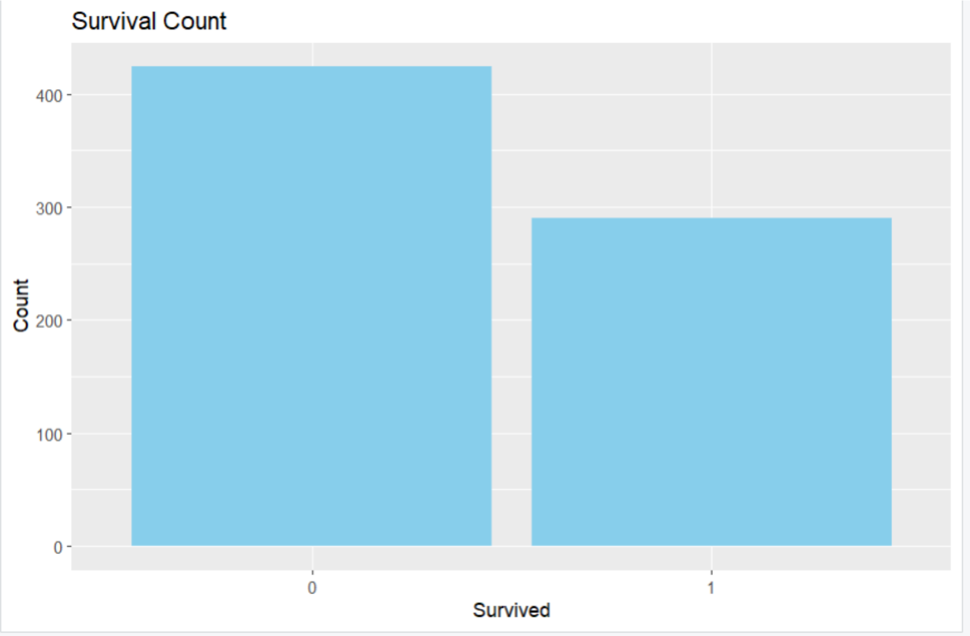
Boxplots of Cleaned Data
20
40
60
80
temp
hum
windspeed

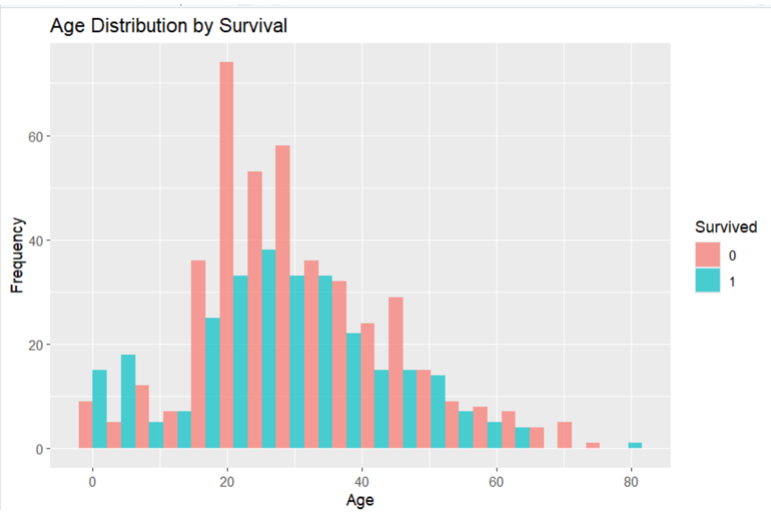
5. Perform analytics on any standard data set

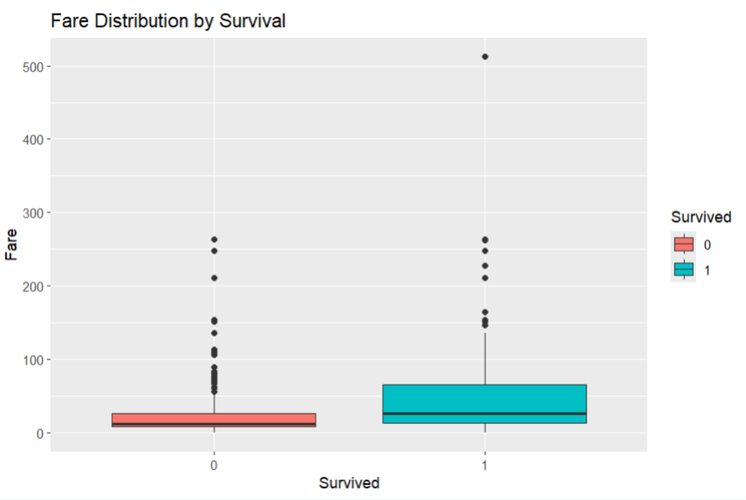
```
# Install and load necessary libraries
if(!require(titanic)) install.packages("titanic")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(ggcorrplot)) install.packages("ggcorrplot")
library(titanic)
library(tidyverse)
library(ggcorrplot)
# Load Titanic dataset from the titanic package
data("titanic_train")
Titanic_df <- titanic_train
# Convert relevant columns to factors
Titanic_df$Survived <- as.factor(Titanic_df$Survived)
Titanic_df$Pclass <- as.factor(Titanic_df$Pclass)
Titanic_df$Sex <- as.factor(Titanic_df$Sex)
# Remove rows with missing 'Age' values for simplicity
Titanic_df <- Titanic_df %>% drop_na(Age)
# View the first few rows of the dataset
head(Titanic_df)
# Check the structure of the dataset
str(Titanic_df)
# Check for missing values
colSums(is.na(Titanic_df))
# Summary statistics of numerical columns
summary(Titanic_df)
# Summary statistics by survival status
Titanic_df %>% group_by(Survived) %>% summarize(across(where(is.numeric), mean,
```

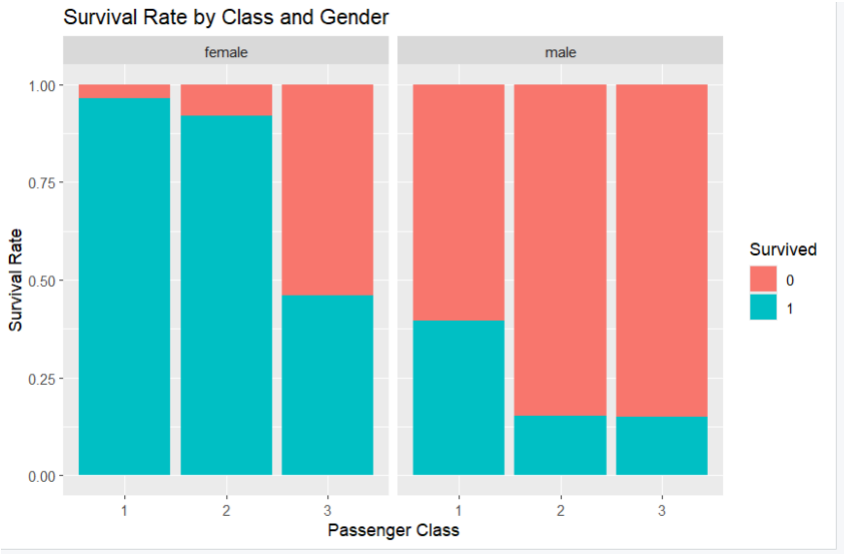
```
na.rm = TRUE))
# Visualization
# 1. Bar plot of survival counts
ggplot(Titanic_df, aes(x = Survived)) +
  geom_bar(fill = "skyblue") +
  labs(title = "Survival Count", x = "Survived", y = "Count")
# 2. Histogram of Age by Survival
ggplot(Titanic_df, aes(x = Age, fill = Survived)) +
  geom_histogram(position = "dodge", bins = 20, alpha = 0.7) +
  labs(title = "Age Distribution by Survival", x = "Age", y = "Frequency")
# 3. Boxplot of Fare by Survival
ggplot(Titanic_df, aes(x = Survived, y = Fare, fill = Survived)) +
  geom_boxplot() +
  labs(title = "Fare Distribution by Survival", x = "Survived", y = "Fare")
# 4. Bar plot of survival rate by class and gender
ggplot(Titanic_df, aes(x = Pclass, fill = Survived)) +
  geom_bar(position = "fill") +
  facet_wrap(~ Sex) +
  labs(title = "Survival Rate by Class and Gender", x = "Passenger Class", y = "Survival
Rate")

# 5. Correlation analysis for numeric columns (Age, Fare, SibSp, Parch)
numeric_data <- Titanic_df %>% select(Age, Fare, SibSp, Parch)
cor_matrix <- cor(numeric_data, use = "complete.obs")

# Plot correlation heatmap
ggcorrplot(cor_matrix, lab = TRUE)
```

OUTPUT

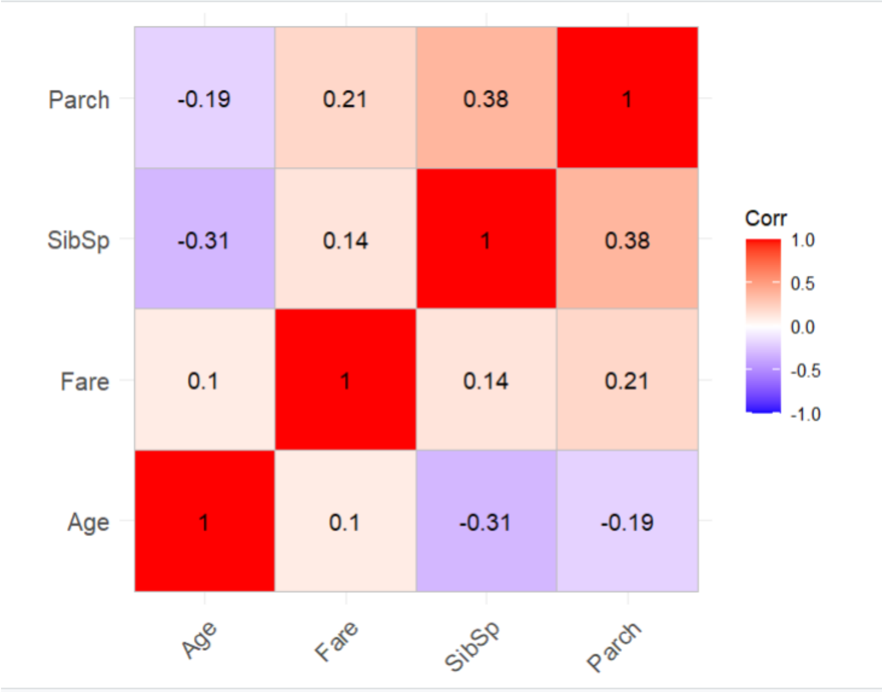

Survival Count
Count
400
300
200
100
0
0
1
Survived


Age Distribution by Survival
Frequency
60
40
20
0
0
20
40
60
80
Age
Survived
0
1


Fare Distribution by Survival
Fare
500
400
300
200
100
0
0
1
Survived
Survived
0
1



Survival Rate by Class and Gender
female
male
Survival Rate
1.00
0.75
0.50
0.25
0.00
1
2
3
1
2
3
Passenger Class
Survived
0
1



Parch
SibSp
Fare
Age
-0.19
0.21
0.38
1
-0.31
0.14
1
0.38
0.1
1
0.14
0.21
1
0.1
-0.31
-0.19
Age
Fare
SibSp
Parch
Corr
1.0
0.5
0.0
-0.5
-1.0

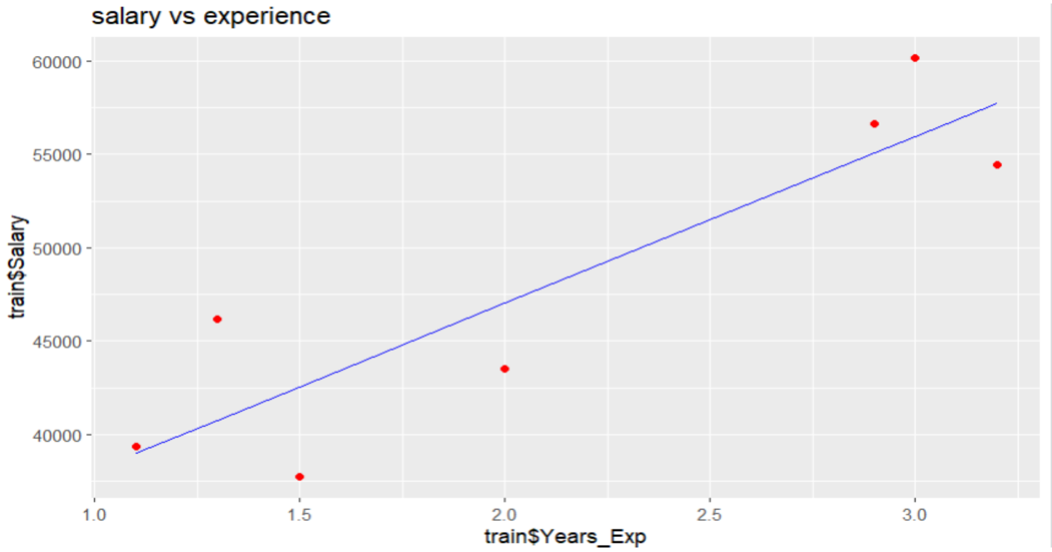
6. Implement linear regression

```
library(caTools)
library(ggplot2)

# Create the data frame
data <- data.frame(
  Years_Exp = c(1.1, 1.3, 1.5, 2.0, 2.2, 2.9, 3.0, 3.2, 3.2, 3.7),
  Salary = c(39343.00, 46205.00, 37731.00, 43525.00,
             39891.00, 56642.00, 60150.00, 54445.00, 64445.00, 57189.00)
)

# Split the data into training and testing sets
set.seed(123)  # Set seed for reproducibility
split = sample.split(data$Salary, SplitRatio = 0.7)
train = subset(data, split == TRUE)
test = subset(data, split == FALSE)

# Fit the linear model
lm.r = lm(formula = Salary ~ Years_Exp, data = train)

# Print the coefficients
print(coef(lm.r))

# Create the ggplot
ggplot() +
  geom_point(aes(x = train$Years_Exp, y = train$Salary), col = 'red') +
  geom_line(aes(x = train$Years_Exp, y = predict(lm.r, newdata = train)), col = "blue")
+
```

```
ggtitle("Salary vs Experience") +
xlab("Years of Experience") +
ylab("Salary")
```

Output:

```
(Intercept) Years_Exp
  29172.310    8922.322
```



7. Implement logistic regression

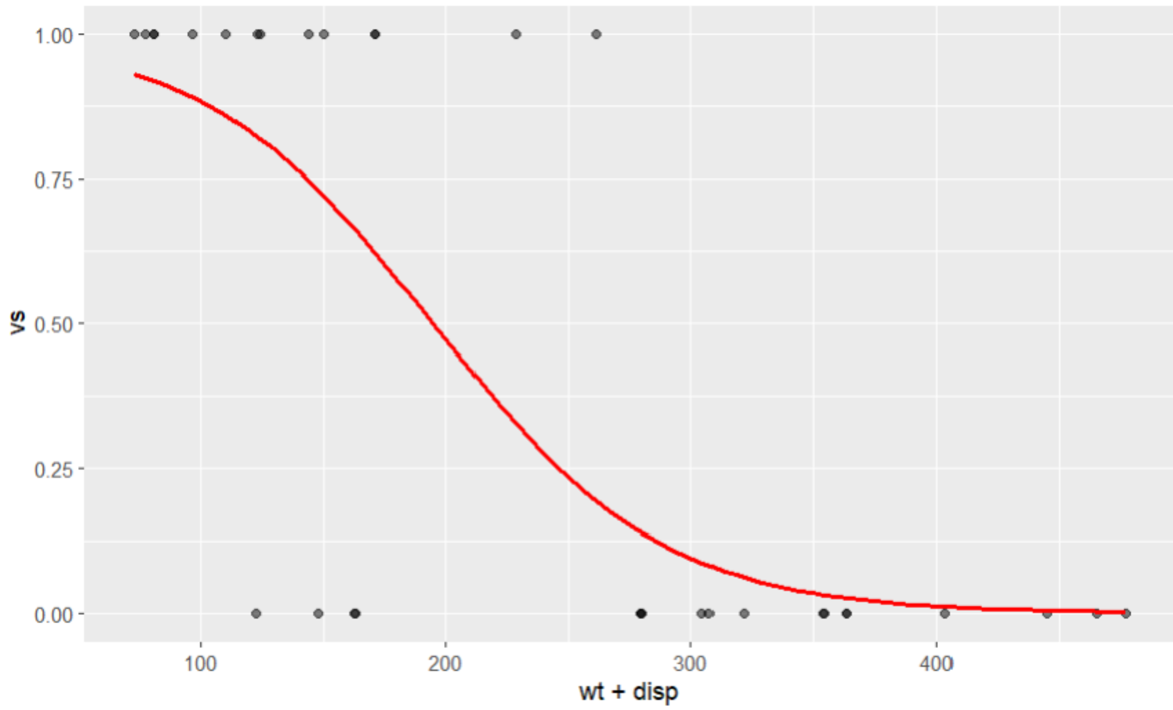
```
library(tidyverse)
library(ROCR)
library(caTools)
library(ggplot2)

# View the mtcars dataset
view(mtcars)

# Split the data into training and testing sets
split <- sample.split(mtcars$vs, SplitRatio = 0.8)  # Ensure using 'vs' for splitting
train <- subset(mtcars, split == TRUE)  # Use TRUE without quotes
test <- subset(mtcars, split == FALSE)  # Use FALSE without quotes

# Build the logistic regression model
logistic_model <- glm(vs ~ wt + disp, data = train, family = binomial)
summary(logistic_model)

# Make predictions
predict_reg <- predict(logistic_model, test, type = "response")

# Convert probabilities to binary outcomes
predict_reg <- ifelse(predict_reg > 0.5, 1, 0)

# Create a confusion matrix
confusion_matrix <- table(test$vs, predict_reg)
print(confusion_matrix)

# Calculate and print classification error and accuracy
missing_classerr <- mean(predict_reg != test$vs)
print(paste("Classification Error = ", missing_classerr))
print(paste("Accuracy = ", (1 - missing_classerr)))

# Plot logistic regression curve
ggplot(train, aes(x = wt + disp, y = vs)) +
  geom_point(alpha = .5) +
  stat_smooth(method = "glm", se = FALSE, method.args = list(family = binomial), col =
"red") +
  labs(title = "Logistic Regression Curve", x = "Weight + Displacement", y = "VS")
```

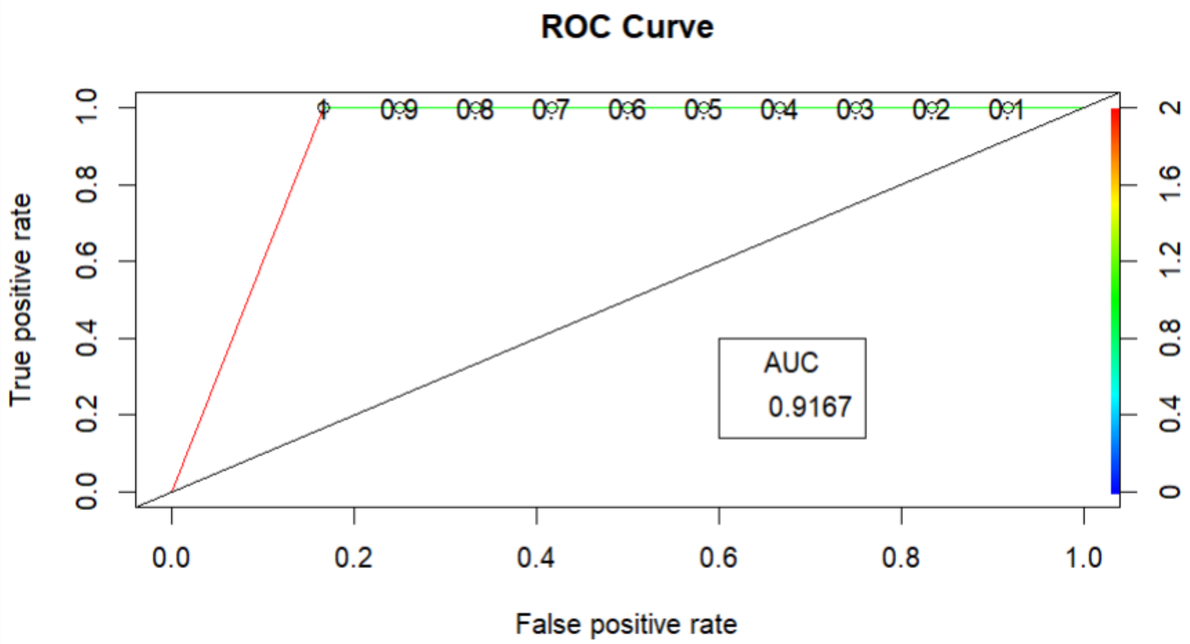
```
# ROC Curve
ROCPred = prediction(predict_reg, test$vs)
ROCPer = performance(ROCPred, measure = "tpr", x.measure = "fpr")
auc <- performance(ROCPred, measure = "auc")
auc_value <- auc@y.values[[1]]
auc_value <- round(auc_value, 4)

# Plot ROC Curve
plot(ROCPer, colorize = TRUE, print.cutoffs.at = seq(0.1, by = 0.1), main = "ROC Curve")
abline(a = 0, b = 1)
legend(0.6, 0.4, paste("AUC =", auc_value), title = "AUC", cex = 1)
```

Output

```
Call:
glm(formula = vs ~ wt + disp, family = binomial, data = train)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.79114    2.96489   0.941    0.347
wt           0.85989    1.55388   0.553    0.580
disp        -0.02718    0.01456  -1.866    0.062 .
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 31.841 on 22 degrees of freedom
Residual deviance: 17.188  on 20  degrees of freedom
```

```
AIC: 23.188

Number of Fisher Scoring iterations: 6

>
> predict_reg <- predict(logistic_model, test, type = "response")
> predict_reg
          Datsun 710   Hornet Sportabout            Merc 230         Merc 450SLC
         0.864210634         0.017371341         0.841966715         0.189302645
Lincoln Continental      Toyota Corolla    Pontiac Firebird       Porsche 914-2
         0.006385438         0.919574847         0.008373046         0.796023875
       Maserati Bora
         0.089476536
>
> predict_reg <- ifelse(predict_reg >0.5, 1, 0)
>
> table(test$vs, predict_reg)
   predict_reg
    0 1
  0 5 1
  1 0 3
>
> missing_classerr <- mean(predict_reg != test$vs)
> missing_classerr
[1] 0.1111111
> print(paste("accuracy = ", (1 - missing_classerr)))
[1] "accuracy =  0.888888888888889"
>
> library(ggplot2)
>
> #plot logistic regression curve
> ggplot(mtcars, aes(x=wt + disp, y=vs)) +
+  geom_point(alpha=.5) +
+  stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial),
+              col="red")
`geom_smooth()` using formula = 'y ~ x'
> auc
[1] 0.9166667
```


1.00
0.75
0.50
0.25
0.00
vs
100
200
300
400
wt + disp

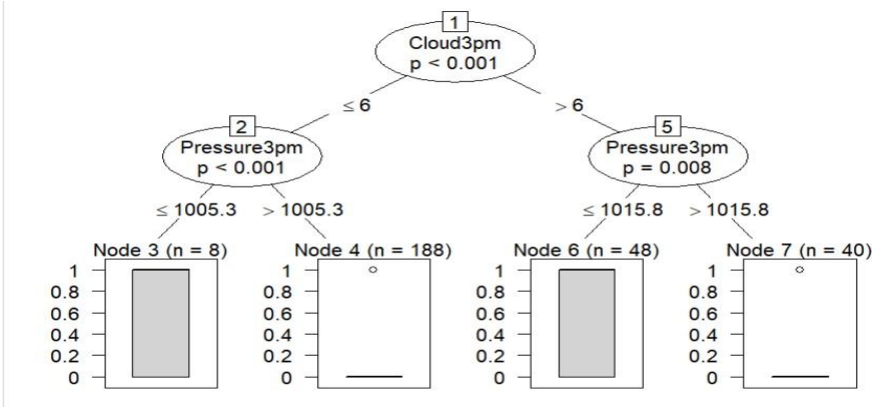

ROC Curve
True positive rate
False positive rate
AUC
0.9167
0.0
0.2
0.4
0.6
0.8
1.0
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2
1.6
1.2
0.8
0.4
0

8. Construct decision tree for weather data set

```
sample = sample(c(TRUE, FALSE), nrow(weatherdata), replace = TRUE, prob = c
(0.8, 0.2))

train <- weatherdata[sample, ]
test <- weatherdata[!sample, ]

library(partykit)
model <- ctree(RainTomorrow ~ ., train)
plot(model)

predict_model <- predict(model, test)
predict_model

mat <- table(test$RainTomorrow, predict_model)
mat

accuracy <- sum(diag(mat)) / sum(mat)
accuracy
```

Output:

```
   predict_model
    0.0478723404255319 0.175 0.625
  0                 52    10     5
  1                  7     4     4
>
> accuracy <- sum(diag(mat)) / sum(mat)
> accuracy
[1] 0.6829268
```

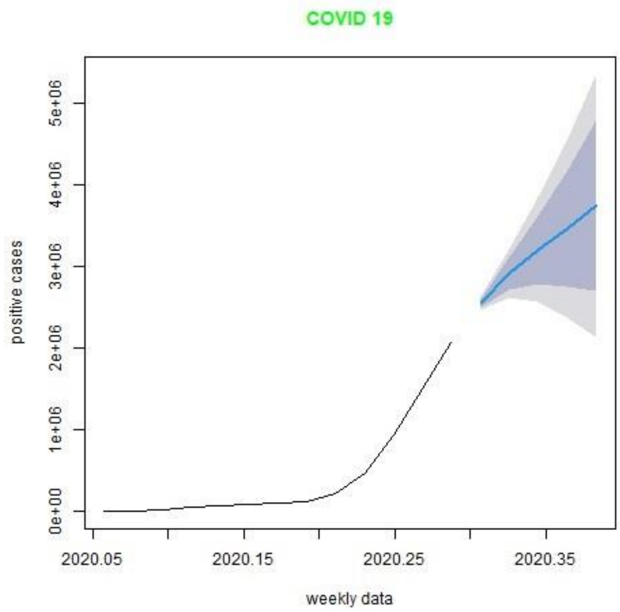


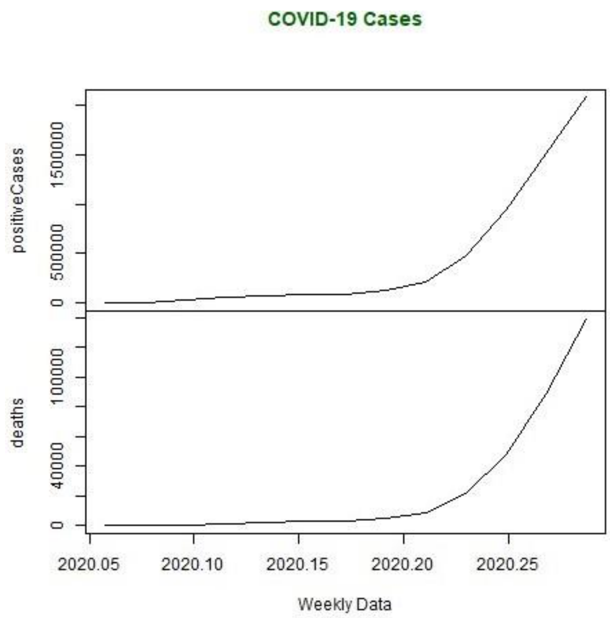
9. Analyse time-series data

```
positiveCases <- c(580, 7813, 28266, 59287,75700, 87820, 95314, 126214,
218843, 471497, 936851,1508725, 2072113)
deaths <- c(17, 270, 565, 1261, 2126, 2800,
            3285, 4628, 8951, 21283, 47210,
            88480, 138475)
library(lubridate)
# output to be created as png file
png(file="multivariateTimeSeries.png")
# creating multivariate time series object
# from date 22 January, 2020
mts <- ts(cbind(positiveCases, deaths),
          start = decimal_date(ymd("2020-01-22")),
          frequency = 365.25 / 7)
# plotting the graph
plot(mts, xlab ="Weekly Data",
     main ="COVID-19 Cases",
     col.main ="darkgreen")
library(forecast)
library(lubridate)
png(file = "timeseries.png")
mts1 <- ts(positiveCases, decimal_date(ymd("2020-01-22")), frequency =
365.25/7)
fit <- auto.arima(mts1)
fit <- forecast(fit, 5)
```

```
plot(forecast(fit, 5), xlab="weekly data", ylab = "positive cases", main = "COVID 19", col.main = "green")

dev.off()
```

Output:

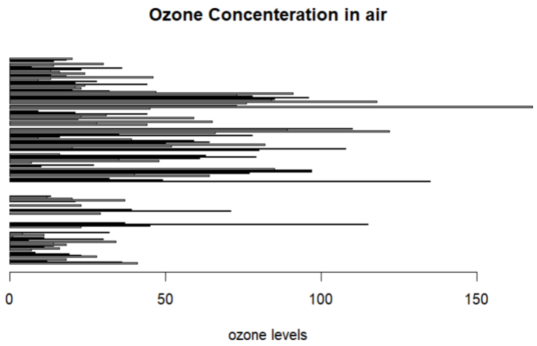




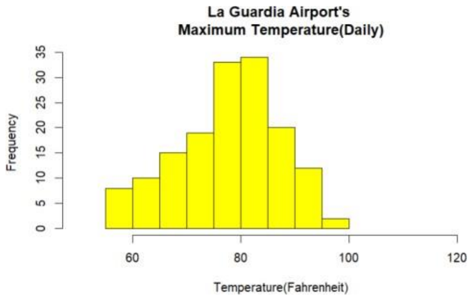
10. Work on any data visualization tool

```
view(airquality)
barplot(airquality$Ozone,
        main = 'Ozone Concenteration in air',
        xlab = 'ozone levels', horiz = TRUE)
hist(airquality$Temp, main ="La Guardia Airport's\
Maximum Temperature(Daily)",
     xlab ="Temperature(Fahrenheit)",
     xlim = c(50, 125), col ="yellow",
     freq = TRUE)
boxplot(airquality[, 0:4],
        main ='Box Plots for Air Quality Parameters')
plot(airquality$Ozone, airquality$Month,
     main ="Scatterplot Example",
     xlab ="Ozone Concentration in parts per billion",
     ylab =" Month of observation ", pch = 19)
data <- matrix(rnorm(50, 0, 5), nrow = 5, ncol = 5)

# Column names
colnames(data) <- paste0("col", 1:5)
rownames(data) <- paste0("row", 1:5)

# Draw a heatmap
heatmap(data)
```

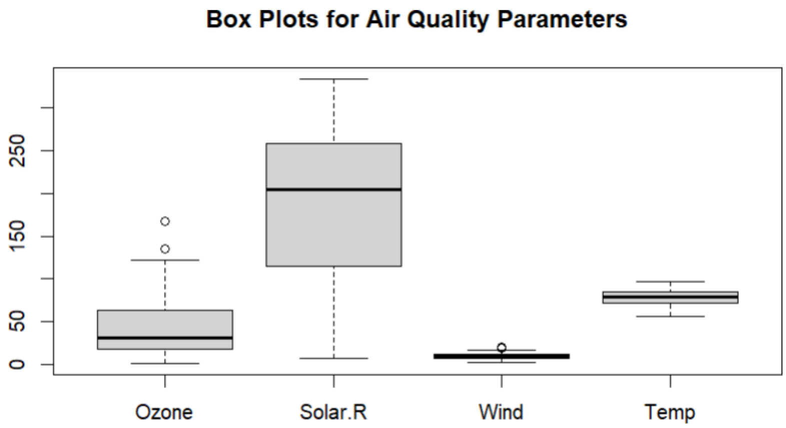
Output



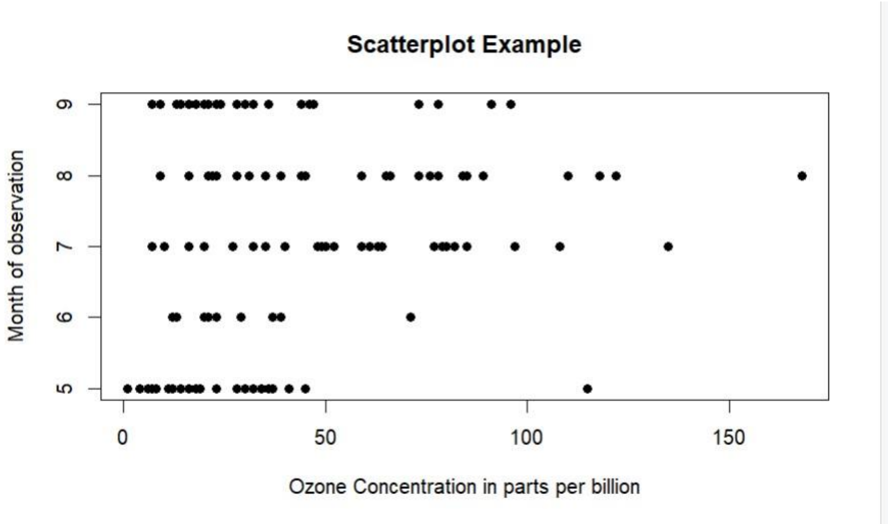
Ozone Concenteration in air
0
50
100
150
ozone levels



La Guardia Airport's
Maximum Temperature(Daily)
Frequency
0
5
10
15
20
25
30
35
60
80
100
120
Temperature(Fahrenheit)



Box Plots for Air Quality Parameters
0
50
150
250
Ozone
Solar.R
Wind
Temp



Scatterplot Example
Month of observation
5
6
7
8
9
0
50
100
150
Ozone Concentration in parts per billion