**Basics of R**

> # R as a calculator

> 3+5

[1] 8

> 10-4

[1] 6

> 7\*3

[1] 21

> 20/4

[1] 5

> 10%%3

[1] 1

> # Variables and assignment

> x <- 5

> y <- 10

> print(x+y)

[1] 15

> z <- x+y

> print(z)

[1] 15

> ch <- "Hello! R"

> print(ch)

[1] "Hello! R"

> log <- TRUE

> print(log)

[1] TRUE

> logicals <- c(TRUE, FALSE, TRUE)

> logicals

[1] TRUE FALSE TRUE

> words <- c("apple", "banana", "cherry")

> words

[1] "apple" "banana" "cherry"

> words[1]

[1] "apple"

> numbers <- c(1, 2, 3, 4, 5)

> numbers

[1] 1 2 3 4 5

> sum(numbers)

[1] 15

> mean(numbers)

[1] 3

> length(numbers)

[1] 5

> matrix\_data <- matrix(1:9, nrow = 3, ncol = 3)

> matrix\_data

[,1] [,2] [,3]

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

> matrix\_data[1, 2]

[1] 4

> df <- data.frame(Name = c("John", "Jane", "Joe"),

+ Age = c(28, 23, 34),

+ Salary = c(6000, 5000, 7000))

> df[]

Name Age Salary

1 John 28 6000

2 Jane 23 5000

3 Joe 34 7000

> df[1, ]

Name Age Salary

1 John 28 6000

> mf <- function(x, y) {

+ result <- x+y

+ return(result)

+ }

> mf(2, 5)

[1] 7

1. **Demonstrate data cleaning – missing values**

install.packages("tidyverse")

library(tidyverse)

x <- sample(1:21, 20, replace = TRUE)

y <- sample(1:11, 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] 3 16 78 104 130 163 170 172 197 200 207 210 227 231 300 302 307 330 341

> sum(is.na(mtcars))

[1] 19

> na.exclude(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb

Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4

Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1

Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1

Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2

Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4

Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3

Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4

Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 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

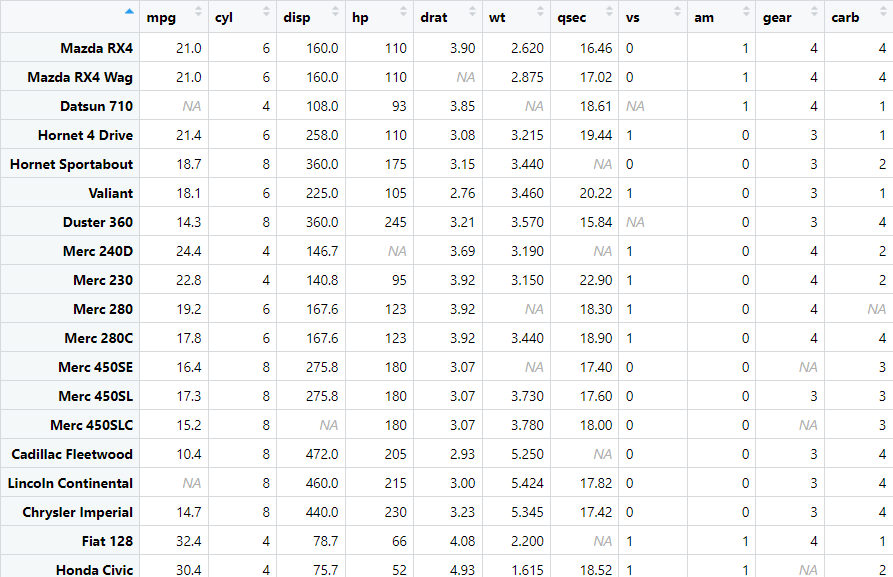
Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4

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

> View(mtcars)



> View(newCars)



1. **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(arr2)

sdarr <- sd(arr2)

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

> print(arr2)

[1] 18.62 11.92 17.40 30.19 39.94 19.84 10.29 5.82 1.35 45.02 19.63 31.82

1. **Implement attribute subset selection for data reduction**

install.packages("leaps")

install.packages("dplyr")

library(leaps)

library(dplyr)

View(Titanic)

sum(is.na(Titanic))

Titanic = Titanic %>%

na.omit()

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)

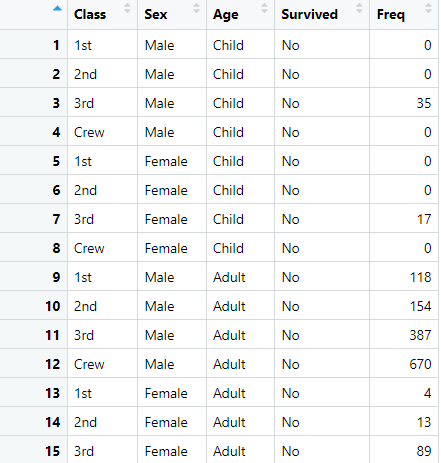
coef(fwd, 3)

coef(bwd, 3)

coef(full, 3)

**Output:**

> View(Titanic)



> sum(is.na(Titanic))

[1] 0

> dim(Titanic)

[1] 4 2 2 2

> 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 ) " " " " " " "\*" "\*" " "

3 ( 1 ) " " " " " " "\*" "\*" "\*"

4 ( 1 ) " " " " "\*" "\*" "\*" "\*"

5 ( 1 ) " " "\*" "\*" "\*" "\*" "\*"

6 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

> summary(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

1. **Demonstrate outlier detection**

install.packages("tidyr")

library(tidyr)

install.packages("lubridate")

library(lubridate)

View(day)

sum(is.na(day))

day <- data.frame(

date = as.Date("2024-11-14") + 0:4,

temp = c(20, 22, 21, 19, 24),

hum = c(30, 40, 35, 45, 50),

windspeed = c(5, 10, 7, 8, 12)

)

boxplot(day[c("temp", "hum", "windspeed")])

for(i in c("hum", "windspeed")){

data <- unlist(day[i])

newData <- data[data %in% boxplot.stats(data) $ out]

data[data %in% newData] = NA

day[i] = data

}

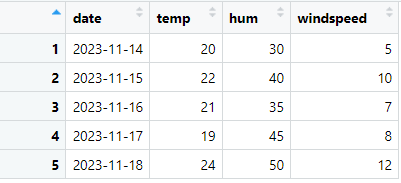
sum(is.na(data))

day = drop\_na(day)

boxplot(day[c("temp", "hum", "windspeed")])

**Output:**

> View(day)



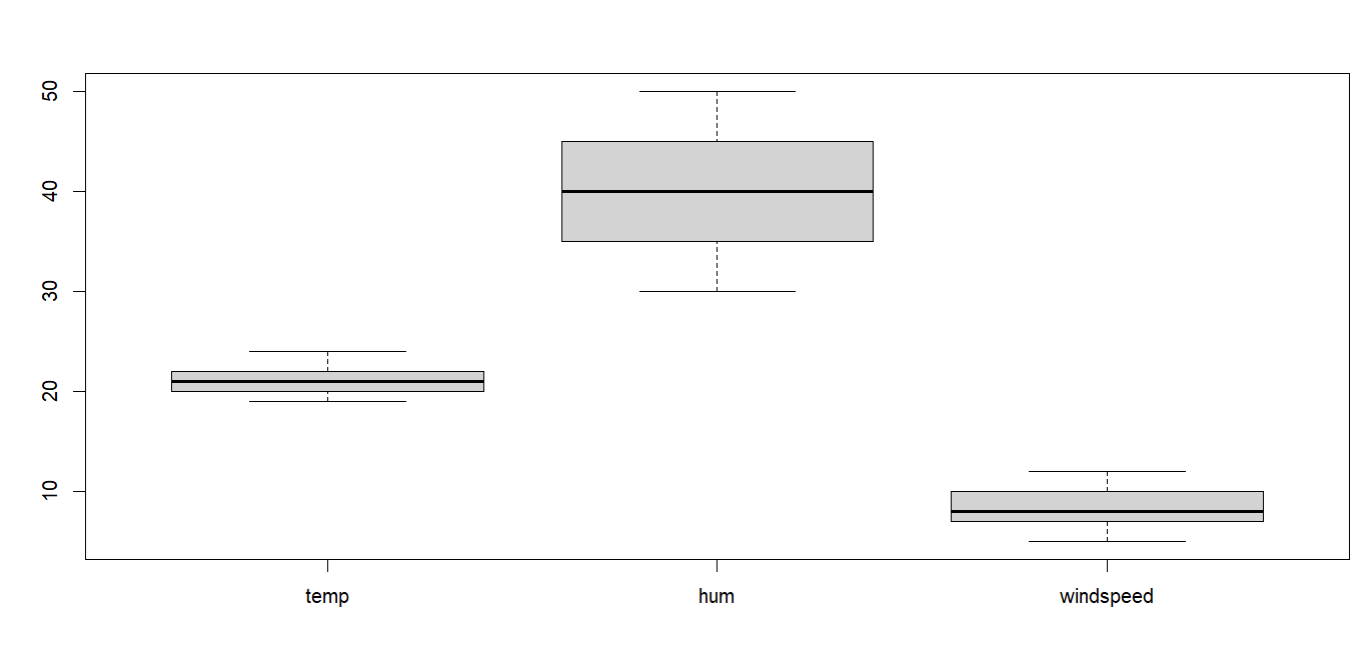
> sum(is.na(day))

[1] 0

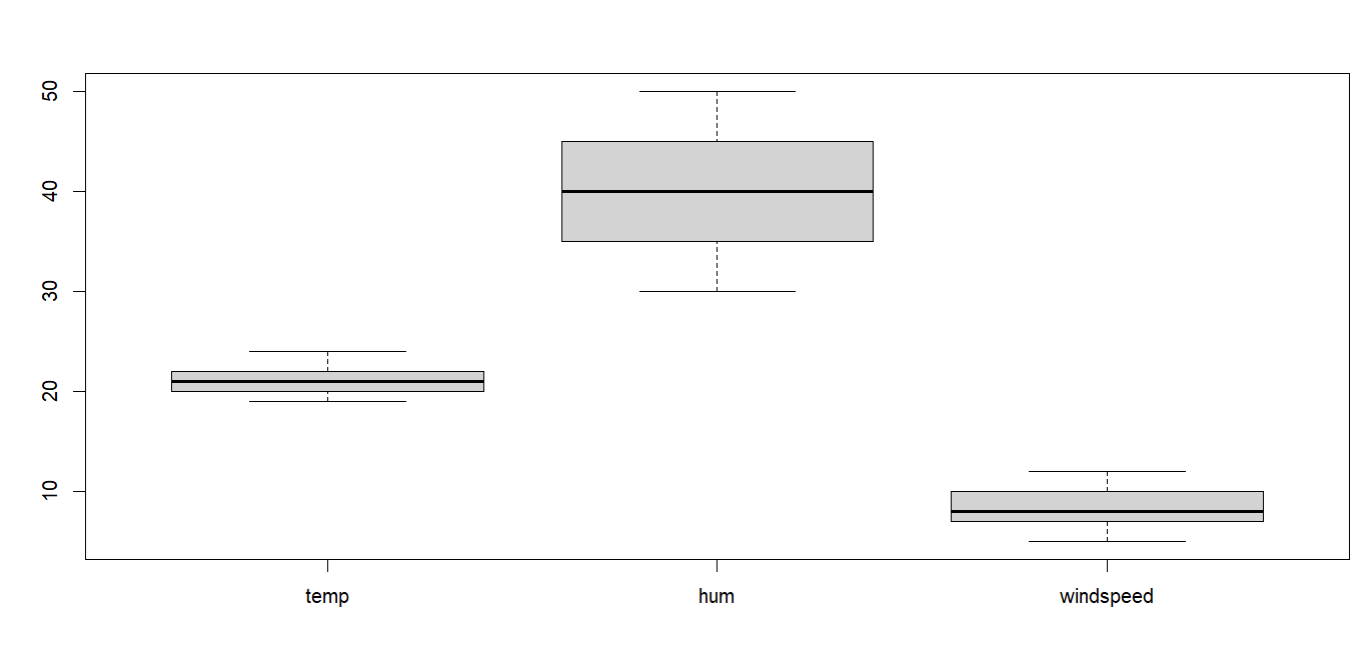
> sum(is.na(data))

[1] 0

> boxplot(day[c("temp", "hum", "windspeed")])



> boxplot(day[c("temp", "hum", "windspeed")])



1. **Perform analytics on any standard data set**

install.packages("titanic")

library(titanic)

titanic <- titanic\_train

# View the first few rows and check data types

head(titanic)

sapply(titanic, class)

# Convert columns to factors

titanic$Sex <- as.factor(titanic$Sex)

titanic$Survived <- as.factor(titanic$Survived)

# Summary of the dataset

summary(titanic)

# Remove rows with missing values

# dropnull\_titanic <- titanic[complete.cases(titanic), ]

dropnull\_titanic = titanic[rowSums(is.na(titanic)) <= 0, ]

# Separate survived and not survived passengers

survivedList <- dropnull\_titanic[dropnull\_titanic$Survived == 1, ]

notSurvivedList <- dropnull\_titanic[dropnull\_titanic$Survived == 0, ]

# Pie chart of survival

mytable <- table(titanic$Survived)

lbls <- paste(c("Not Survived", "Survived"), "\n", mytable, sep = "")

pie(

mytable,

labels = lbls,

main = "Survival Pie Chart"

)

# Histogram of age distribution

hist(

titanic$Age,

xlab = "Age",

ylab = "Frequency"

)

# Bar plot of gender distribution for not survived passengers

barplot(

table(notSurvivedList$Sex),

xlab = "Gender",

ylab = "Frequency"

)

# Density plot for Fare of survived passengers

temp <- density(table(survivedList$Fare))

plot(temp, type = "n", main = "fare charged")

polygon(temp, col = "lightgray", border = "gray")

# Boxplot of Fare

boxplot(

titanic$Fare,

main = "Boxplot of Fare"

)

**Output:**

> head(titanic)

PassengerId Survived Pclass Name Sex Age SibSp

1 1 0 3 Braund, Mr. Owen Harris male 22 1

2 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female 38 1

3 3 1 3 Heikkinen, Miss. Laina female 26 0

4 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35 1

5 5 0 3 Allen, Mr. William Henry male 35 0

6 6 0 3 Moran, Mr. James male NA 0

Parch Ticket Fare Cabin Embarked

1 0 A/5 21171 7.2500 S

2 0 PC 17599 71.2833 C85 C

3 0 STON/O2. 3101282 7.9250 S

4 0 113803 53.1000 C123 S

5 0 373450 8.0500 S

6 0 330877 8.4583 Q

> sapply(titanic, class)

PassengerId Survived Pclass Name Sex Age SibSp Parch

"integer" "integer" "integer" "character" "character" "numeric" "integer" "integer"

Ticket Fare Cabin Embarked

"character" "numeric" "character" "character"

> summary(titanic)

PassengerId Survived Pclass Name Sex Age

Min. : 1.0 0:549 Min. :1.000 Length:891 female:314 Min. : 0.42

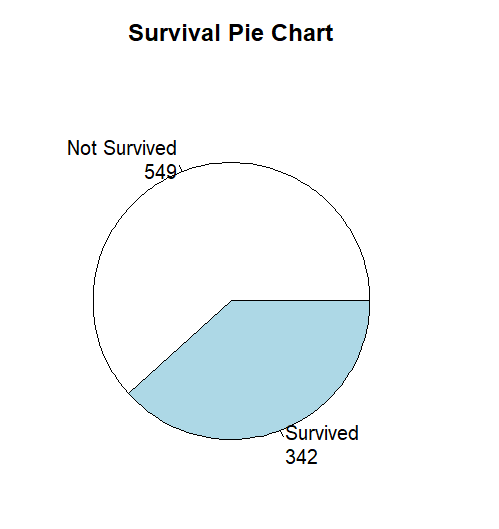
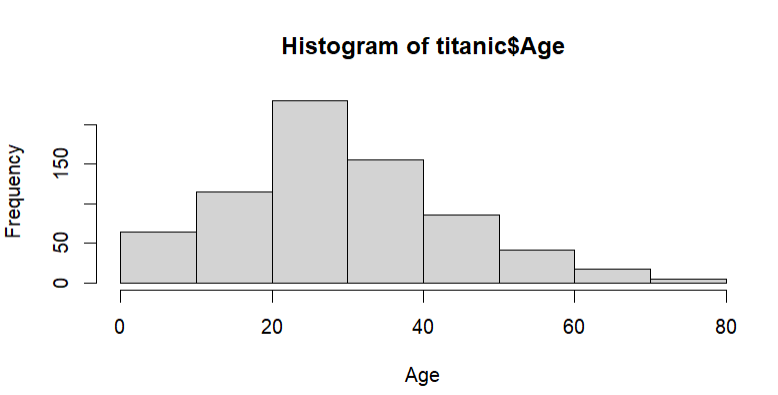
1st Qu.:223.5 1:342 1st Qu.:2.000 Class :character male :577 1st Qu.:20.12

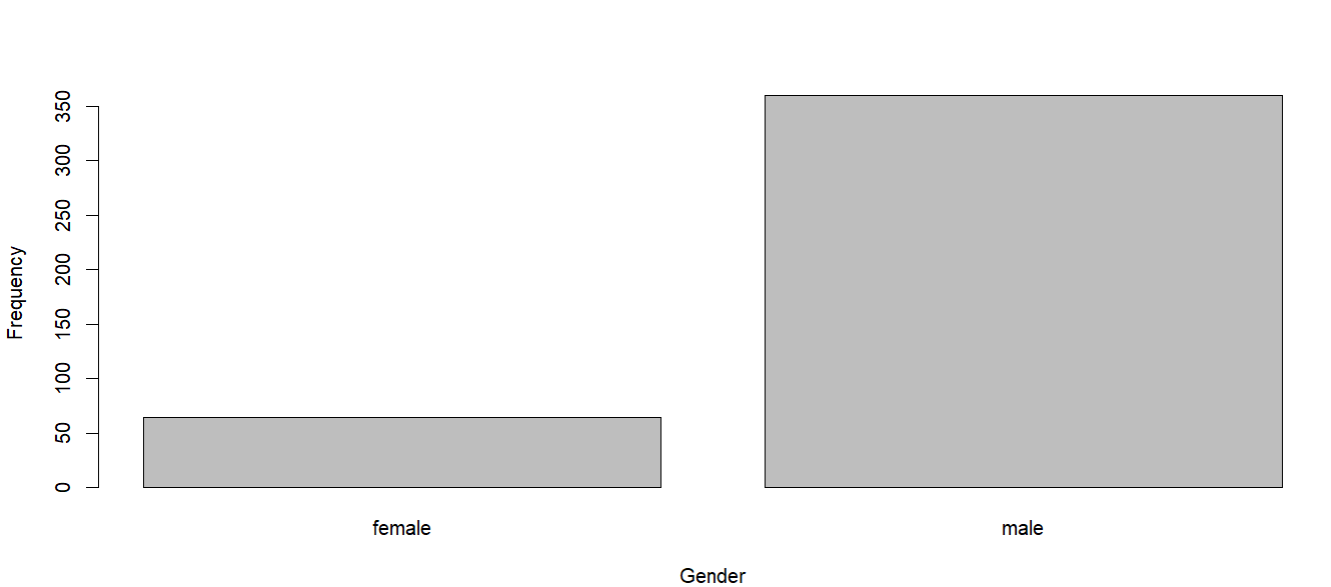
Median :446.0 Median :3.000 Mode :character Median :28.00

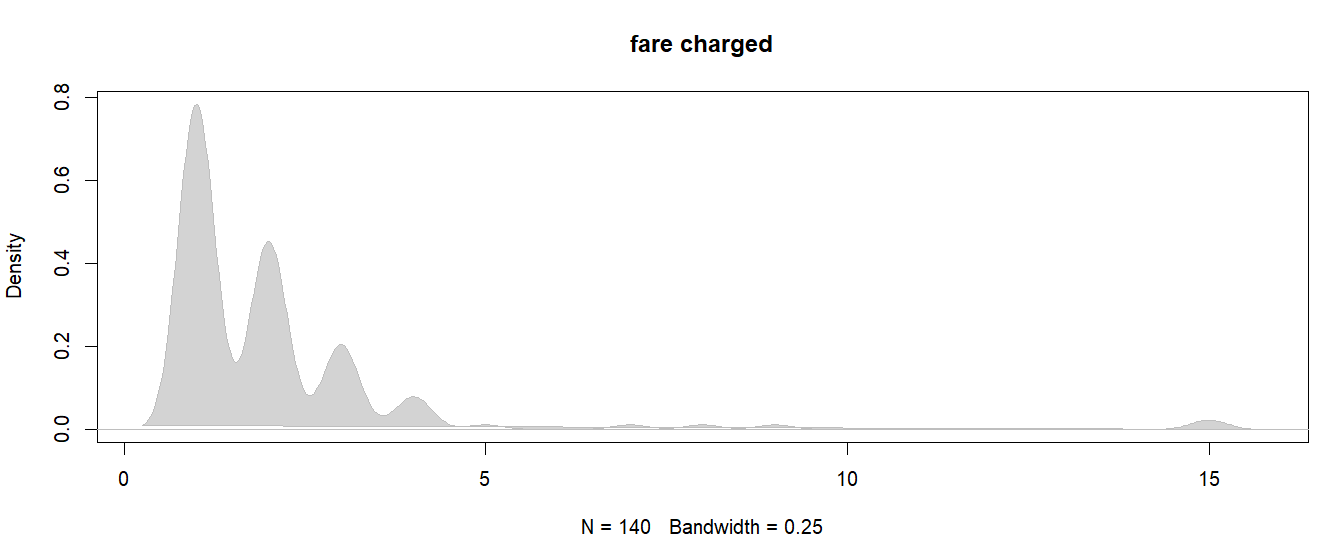
Mean :446.0 Mean :2.309 Mean :29.70

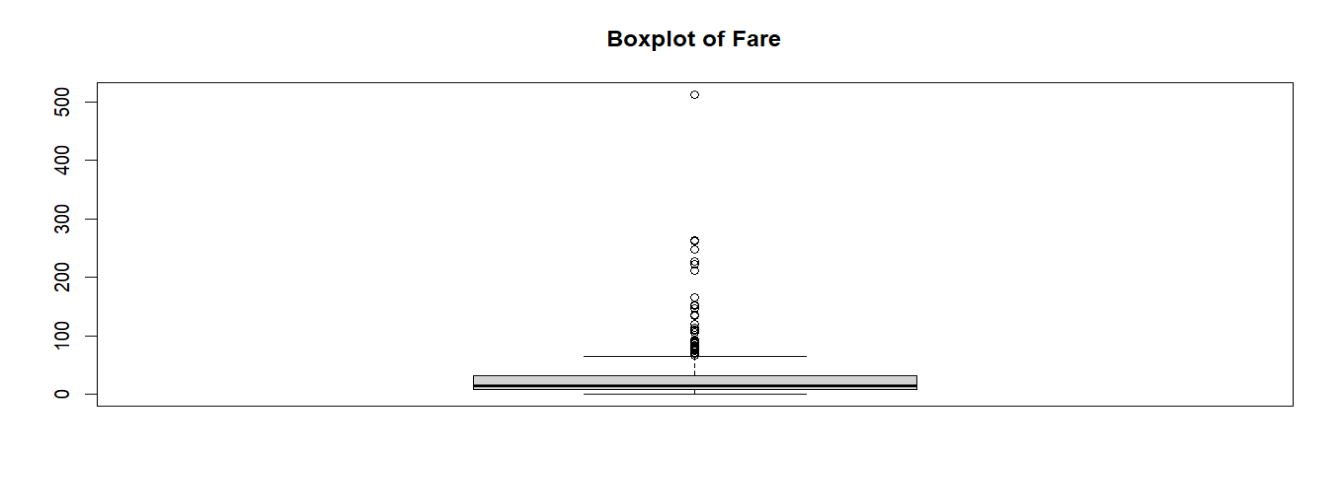
3rd Qu.:668.5 3rd Qu.:3.000 3rd Qu.:38.00

Max. :891.0 Max. :3.000 Max. :80.00







1. **Implement linear regression**

install.packages("caTools")

library(caTools)

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 <- sample.split(data $ Salary, SplitRatio = 0.7)

train = subset(data, split == TRUE)

test = subset(data, split == FALSE)

lm.r = lm(formula = Salary ~ Years\_Exp, data = train)

coef(lm.r)

install.packages("ggplot2")

library(ggplot2)

ggplot() + geom\_point(aes(x=train$Years\_Exp, y=train$Salary), col="red") +

geom\_line(aes(x=train$Years\_Exp, y=predict(lm.r, data=train)), col="blue") +

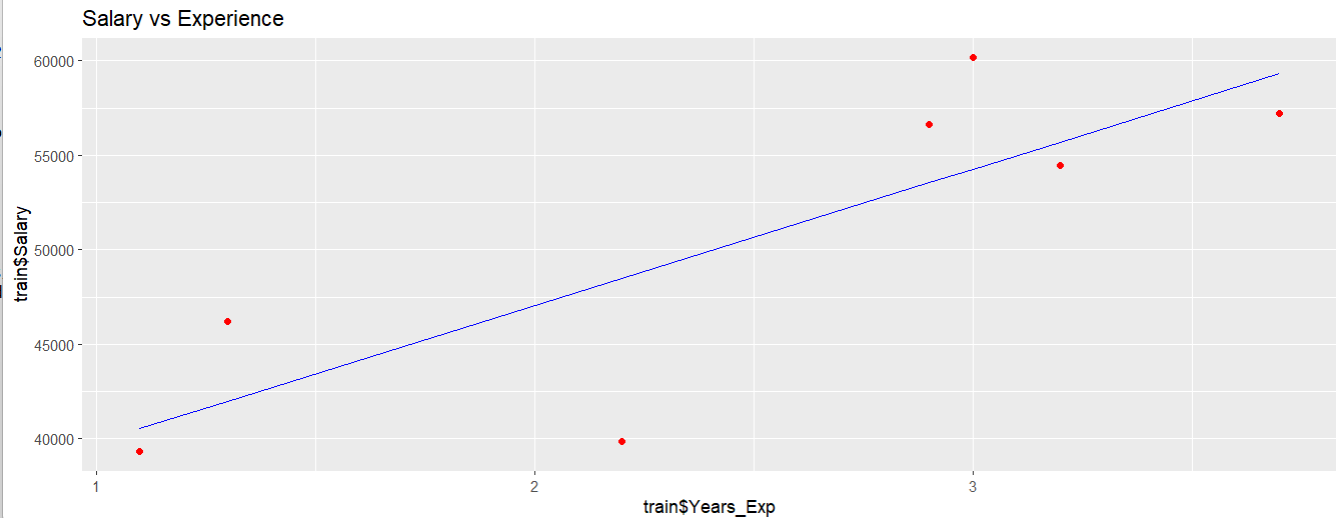
ggtitle("Salary vs Experience")

**Output:**

> coef(lm.r)

(Intercept) Years\_Exp

32626.810 7211.341



1. **Implement logistic regression**

# Load necessary libraries

library(tidyverse)

library(ROCR)

library(caTools)

# View and split the data

data(mtcars) # Ensure 'mtcars' dataset is loaded

mtcars$vs <- as.factor(mtcars$vs) # Convert vs to factor for classification

split <- sample.split(mtcars$vs, SplitRatio = 0.8) # Split dataset based on vs

train <- subset(mtcars, split == TRUE)

test <- subset(mtcars, split == FALSE)

# Logistic regression model

logistic\_model <- glm(vs ~ wt + disp, data = train, family = binomial)

summary(logistic\_model)

# Predict probabilities on the test dataset

predict\_reg <- predict(logistic\_model, test, type = "response")

# Convert probabilities to binary predictions using threshold 0.5

predict\_class <- ifelse(predict\_reg > 0.5, 1, 0)

# Confusion matrix

conf\_mat <- table(Actual = test$vs, Predicted = predict\_class)

print(conf\_mat)

# Calculate accuracy

missing\_classerr <- mean(predict\_class != as.numeric(as.character(test$vs)))

accuracy <- 1 - missing\_classerr

print(paste("Accuracy =", round(accuracy, 4)))

# Plot logistic regression curve

ggplot(mtcars, aes(x = wt + disp, y = as.numeric(as.character(vs)))) +

geom\_point(alpha = 0.5) +

stat\_smooth(method = "glm", method.args = list(family = binomial), se = FALSE, color = "red") +

labs(title = "Logistic Regression Curve", x = "wt + disp", y = "vs") +

theme\_minimal()

# Generate ROC curve

ROCPred <- prediction(predict(logistic\_model, test, type = "response"), test$vs)

ROCPer <- performance(ROCPred, measure = "tpr", x.measure = "fpr")

# Plot ROC curve

plot(ROCPer, colorize = TRUE, print.cutoffs.at = seq(0.1, by = 0.1), main = "ROC Curve")

abline(a = 0, b = 1, col = "gray", lty = 2)

# Calculate and display AUC

auc <- performance(ROCPred, measure = "auc")@y.values[[1]]

auc <- round(auc, 4)

legend("bottomright", legend = paste("AUC =", auc), title = "AUC", cex = 1)

**Output:**

> summary(logistic\_model)

Call:

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

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 3.41788 3.03781 1.125 0.2605

wt 0.60992 1.62181 0.376 0.7069

disp -0.02709 0.01497 -1.809 0.0704 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 34.296 on 24 degrees of freedom

Residual deviance: 17.577 on 22 degrees of freedom

AIC: 23.577

Number of Fisher Scoring iterations: 6

> print(conf\_mat)

Predicted

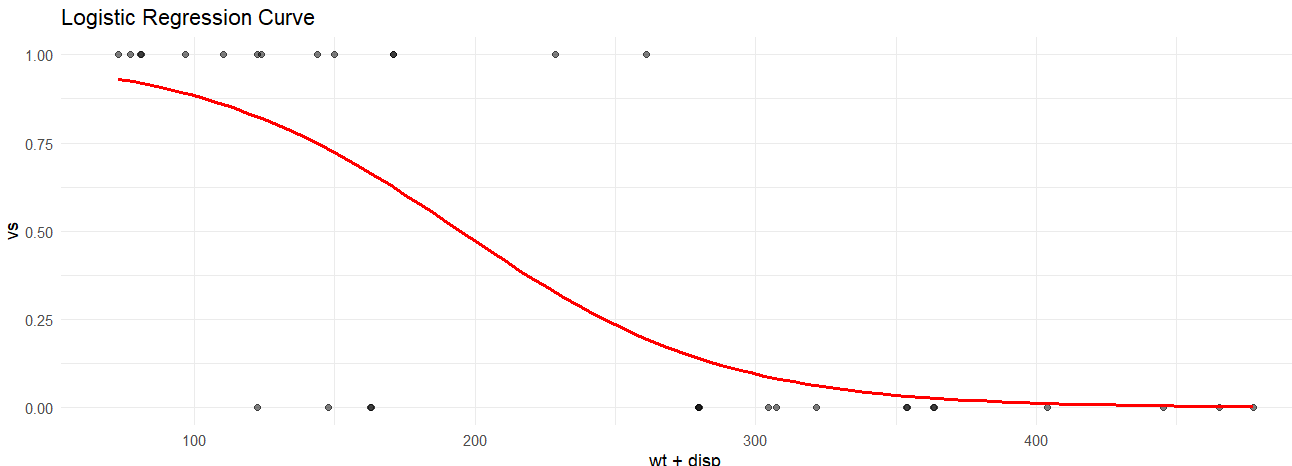
Actual 0 1

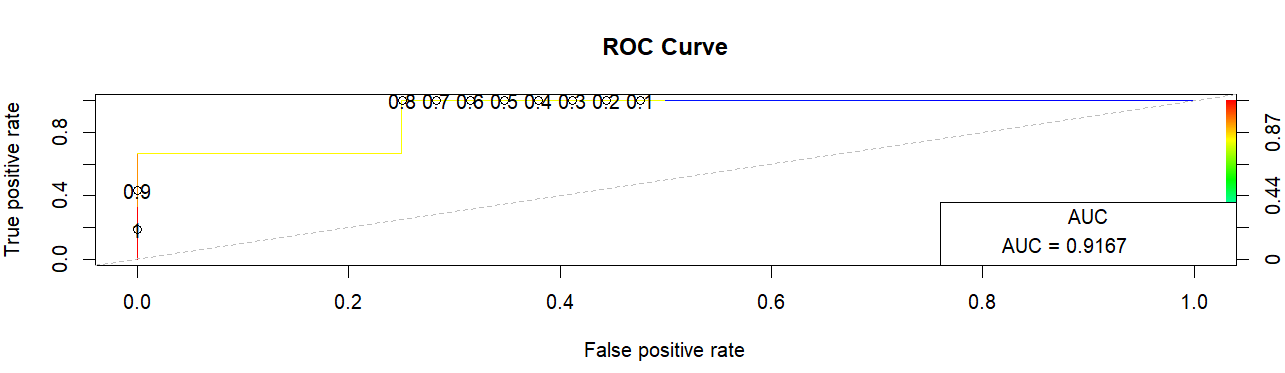
0 3 1

1 0 3

> print(paste("Accuracy =", round(accuracy, 4)))

[1] "Accuracy = 0.8571"





1. **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:**

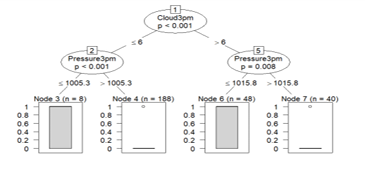
predict\_model 0.0478723404255319 0.175 0.625

0 52 10 5

1 7 4 4

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

> accuracy [1] 0.6829268



1. **Analyze time-series data**

# Loading necessary libraries

library(forecast)

library(lubridate)

# Multivariate Time Series Plot

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)

# Creating multivariate time series object

mts <- ts(cbind(positiveCases, deaths),

start = decimal\_date(ymd("2020-01-22")),

frequency = 365.25 / 7)

# Plotting the multivariate time series

plot(mts, xlab ="Weekly Data",

main ="COVID-19 Cases",

col.main ="darkgreen")

# Univariate Time Series and Forecasting

mts1 <- ts(positiveCases, decimal\_date(ymd("2020-01-22")), frequency = 365.25/7)

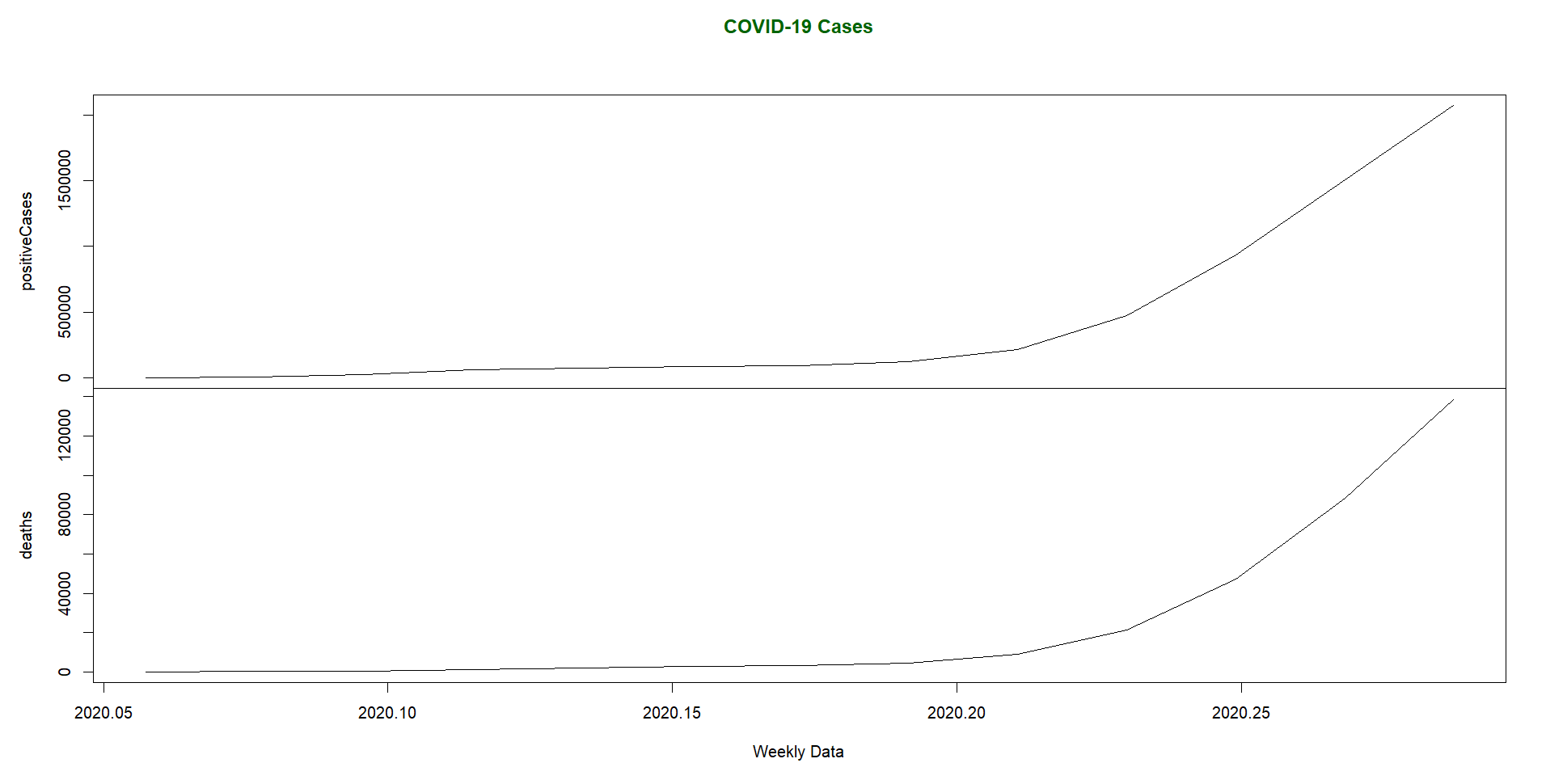
fit <- auto.arima(mts1)

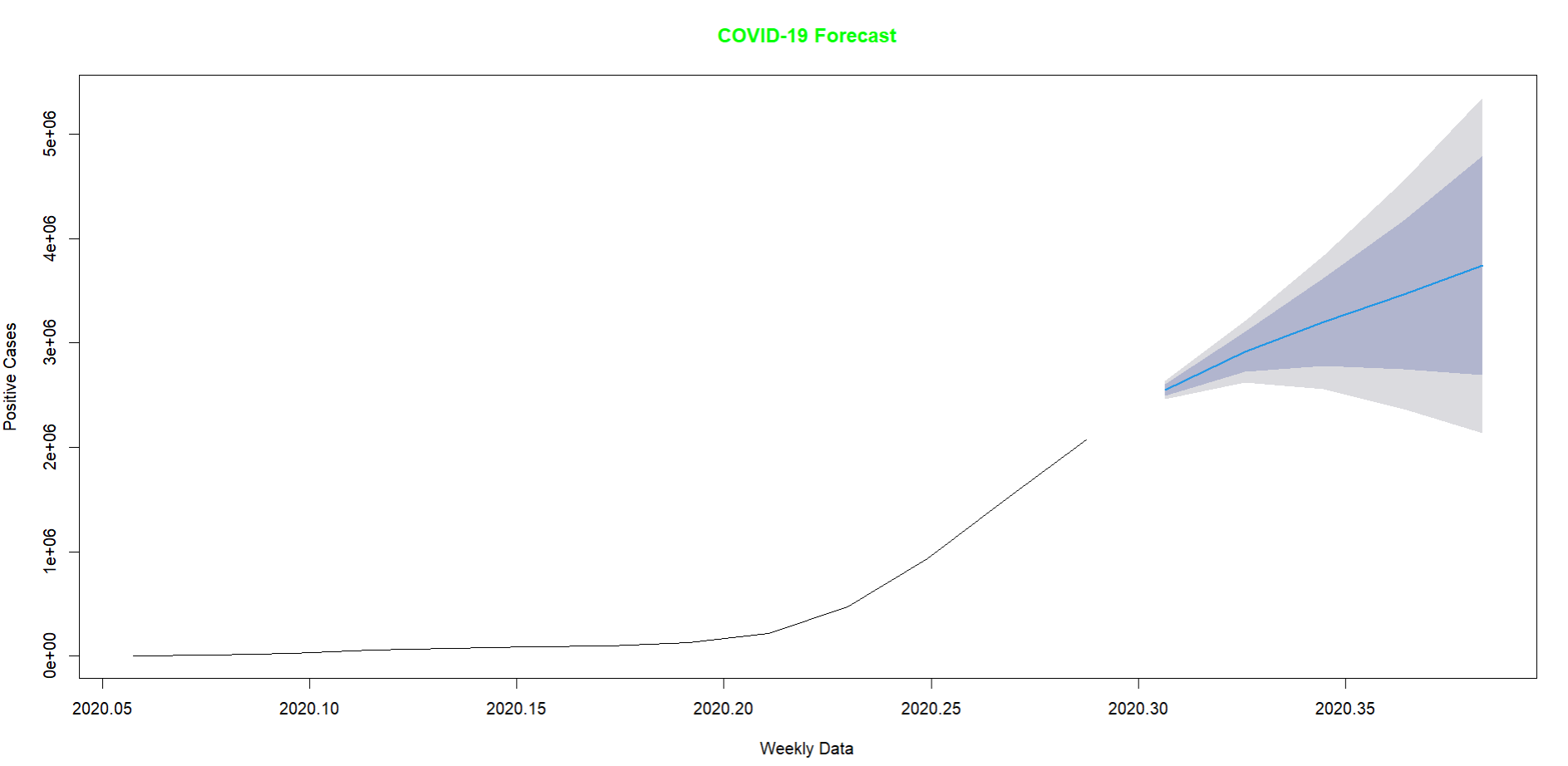
forecast\_fit <- forecast(fit, 5)

# Plotting the forecast

plot(forecast\_fit, xlab="Weekly Data", ylab = "Positive Cases", main = "COVID-19 Forecast", col.main = "green")

**Output:**





1. **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:**

> View(airquality)

