**Q1) Correlation and Scatter Plot**

r

Copy code

# Load the built-in dataset

data(mtcars)

# View the dataset

head(mtcars)

# Compute correlation between mpg (Miles per Gallon) and hp (Horsepower)

correlation <- cor(mtcars$mpg, mtcars$hp)

# Print the correlation

print(paste("correlation between mpg and hp is:", round(correlation,2)))

# Create scatter plot with regression line

library(ggplot2)

ggplot(data = mtcars, aes(x = hp, y = mpg)) +

geom\_point(shape = 4, color = "blue", size = 3) +

geom\_smooth(method = "lm", se = FALSE) +

labs(

title = "MPG vs HorsePower",

x = "HorsePower (HP)",

y = "Miles per Gallon (mpg)"

) +

theme\_minimal()

**Q2) Pearson Correlation & Normality (Q-Q Plot)**

r

Copy code

data()

mtcars

library(ggplot2)

correlation <- cor(mtcars$mpg, mtcars$hp)

print(paste("correlation between mpg and hp is:", round(correlation,2)))

ggplot(data = mtcars, aes(sample = mpg)) +

stat\_qq() +

stat\_qq\_line() +

labs(title = "Q-Q plot for MPG",

x = "Theoretical Quantiles",

y = "MPG (sample)") +

theme\_minimal()

**Q3) Summary Stats: Price Quotes (Mary & Barry)**

r

Copy code

library(ggplot2)

data = read.csv("C:/Users/hp/Desktop/Stats Lab Dataset/pricequotes.csv")

print(summary(data))

n\_barry <- length(data$Barry\_Price)

n\_mary <- length(data$Mary\_Price)

sd\_barry <- sd(data$Barry\_Price)

sd\_mary <- sd(data$Mary\_Price)

se\_barry <- sd\_barry/sqrt(n\_barry)

se\_mary <- sd\_mary/sqrt(n\_mary)

cat("Mary: SD=",round(sd\_mary,2)," | SE = ",round(se\_mary,2))

cat("Barry: SD=",round(sd\_barry,2)," | SE = ",round(se\_barry,2))

ggplot(data,aes(x="Barry",y=Barry\_Price))+

geom\_boxplot(fill="skyblue")+

geom\_boxplot(aes(x="Mary",y=Mary\_Price),fill="lightgreen")+

labs(title="BoxPlot of Price QUotes",x="Person",y="Price")

**Q4) Treatment Facility - Pre/Post Analysis**

r

Copy code

library(dplyr)

library(ggplot2)

df<- read.csv("treatmentfacility.csv")

df$Reengineer <- factor(df$Reengineer,levels=c("Prior","Post"))

summary\_stats = df %>%

group\_by(Reengineer) %>%

summarize(

n=n(),

mean\_turnover = mean(Employee\_Turnover),

sd\_turnover = sd(Employee\_Turnover),

mean\_TRFF = mean(TRFF),

sd\_TRFF = sd(TRFF),

mean\_CI = mean(CI),

sd\_CI = sd(CI)

)

print(summary\_stats)

ggplot(df,aes(x=Reengineer,y=Employee\_Turnover,fill=Reengineer))+

geom\_boxplot()+

labs(title = "Employee Turnover Before and Afterr Rengineering")

ggplot(df,aes(x=Reengineer,y=TRFF,fill=Reengineer))+

geom\_boxplot()+

labs(title="TRFF Before and After Engineering")

**Q5) Baggage Complaints Comparison**

r

Copy code

library(readr)

library(ggplot2)

df <- read.csv("baggagecomplaints.csv")

df <- df %>%

mutate(Rate = 100 \* Baggage/Enplaned )

print(summary(df[c("Baggage","Rate")]))

summary\_airline = df %>%

group\_by(Airline) %>%

summarize(

total\_months = n(),

total\_passangers = sum(Enplaned),

mean\_complaints = mean(Baggage),

median\_complaints = median(Baggage),

sd\_complaints = sd(Baggage),

mean\_rate = mean(Rate),

median\_rate = median(Rate),

sd\_rate = sd(Rate),

min\_rate = min(Rate),

max\_rate = max(Rate)

)

print(summary\_airline,n=Inf,width = Inf)

yearly\_avg <- df%>%

group\_by(Year,Airline) %>%

summarise(

avg\_complaints = mean(Baggage),

.groups="drop" )

ggplot(yearly\_avg, aes(x=Year,y=avg\_complaints,color=Airline))+

geom\_line(linewidth=1.2)+

geom\_point(size=2,color="black")+

theme\_minimal()+

labs(

title="Yearly Avg Baggage Complaints",

x="Year",

y="Avg Complaints"

)

**Q6) Medical Malpractice Descriptive Analysis**

r

Copy code

library(ggplot2)

library(dplyr)

library(readr)

data <- read.csv('medicalmalpractice.csv')

summary(data$Amount)

ggplot(data, aes(x = log10(Amount))) +

geom\_histogram(fill = "lightblue", bins = 20) +

labs(title = "Histogram of Claim Amounts (Log Scale)",

x = "Log Amount",

y = "Frequency")

top3\_specialty <- data %>%

count(Specialty,name="n") %>%

slice\_max(n,n=3) %>%

pull(Specialty)

data %>%

filter(Specialty %in% top3\_specialty) %>%

ggplot(aes(x=Specialty,y=Amount,fill=Specialty))+

geom\_boxplot()+

coord\_flip()+

theme\_minimal()

total = length(data$Specialty)

spec\_percent <- data %>%

group\_by(Specialty) %>%

summarise(

n=n(),

pct = 100\*n/total ) %>%

filter(Specialty %in% c("Anesthesiology","Dermatology","Orthopedic Surgery"))

spec\_percent

cor.test(data$Age, data$Amount)

**Q9) Spearman Rank Correlation in Python**

python

Copy code

import scipy.stats as stats

smip\_scores = [70, 46, 94, 34, 20, 86, 18, 12, 56, 64]

dbms\_scores = [60, 66, 90, 46, 16, 98, 24, 8, 32, 54]

correlation, p\_value = stats.spearmanr(smip\_scores, dbms\_scores)

print(f"Spearman Rank Correlation Coefficient: {correlation:.4f}")

print(f"P-value: {p\_value:.4f}")

**Q10) Simple Linear Regression in Python**

python

Copy code

import pandas as pd

import statsmodels.api as sm

import matplotlib.pyplot as plt

data = {

'TV': [1,2,4,7,9,11,15],

'Sales': [2,4,6,9,12,34,45]

}

df = pd.DataFrame(data)

X = df['TV']

Y = df['Sales']

X = sm.add\_constant(X)

model = sm.OLS(Y,X).fit()

print(model.summary())

plt.scatter(df['TV'],df['Sales'], color='blue', label='Actual Data')

plt.plot(df['TV'],model.predict(X),color='red', label='Fitted Line')

plt.title("Liner Regression: Sales vs TV Advertising Budget")

plt.xlabel("TV Advertising Budget")

plt.ylabel("Sales")

plt.legend()

plt.show()

**Q11) Time Series Forecasting (Australia Drug Sales)**

python

Copy code

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.holtwinters import ExponentialSmoothing

df = pd.read\_csv("AustraliaDrugSales.csv",parse\_dates=['date'])

df.set\_index('date', inplace=True)

df.index.freq = 'MS'

model = ExponentialSmoothing(

df['value'],

trend='add',

seasonal='add',

seasonal\_periods=12

).fit()

forecast = model.forecast(24)

plt.plot(df.index,df['value'],label='Observed',color='blue')

plt.plot(model.fittedvalues.index, model.fittedvalues, label='Fitted', color='red')

plt.plot(forecast.index, forecast,label='Forecast',color='green')

plt.legend()

plt.title("Monthly Drug Sales in Australia")

plt.xlabel("Date")

plt.ylabel("Sales")

plt.show()