7.  
# Sample paired data for fish prices in 1970 and 1980 (replace with your real data)

fish\_prices <- data.frame(

Species = c("Fish1", "Fish2", "Fish3", "Fish4", "Fish5"),

Price\_1970 = c(10, 12, 11, 13, 9),

Price\_1980 = c(15, 18, 16, 19, 14)

)

# 1. Investigate evidence of overfishing:

# (Hypothesis: Price increased from 1970 to 1980 due to reduced supply)

# Summary stats for prices in 1970 and 1980

summary(fish\_prices$Price\_1970)

summary(fish\_prices$Price\_1980)

# Visualize price change

library(ggplot2)

fish\_prices\_long <- reshape2::melt(fish\_prices, id.vars = "Species", variable.name = "Year", value.name = "Price")

fish\_prices\_long$Year <- factor(fish\_prices\_long$Year, levels = c("Price\_1970", "Price\_1980"), labels = c("1970", "1980"))

ggplot(fish\_prices\_long, aes(x = Year, y = Price, group = Species, color = Species)) +

geom\_line() +

geom\_point() +

labs(title = "Fish Prices in 1970 vs 1980", y = "Price", x = "Year") +

theme\_minimal()

# 2. Paired t-test to check if prices increased significantly from 1970 to 1980

t\_test\_result <- t.test(fish\_prices$Price\_1980, fish\_prices$Price\_1970, paired = TRUE)

print(t\_test\_result)

# Interpretation:

cat("\nMean price 1970:", mean(fish\_prices$Price\_1970), "\n")

cat("Mean price 1980:", mean(fish\_prices$Price\_1980), "\n")

cat("Mean difference (1980 - 1970):", mean(fish\_prices$Price\_1980 - fish\_prices$Price\_1970), "\n")

cat("95% Confidence Interval for difference:", t\_test\_result$conf.int, "\n")

cat("p-value:", t\_test\_result$p.value, "\n")  
  
  
8.  
part1  
# Sample patient satisfaction data (replace with your real data)

patient\_satisfaction <- data.frame(

Satisfaction\_Score = c(80, 85, 90, 75, 88, 92, 79, 84, 87, 91)

)

# Summary statistics

summary(patient\_satisfaction$Satisfaction\_Score)

sd(patient\_satisfaction$Satisfaction\_Score)  
  
  
part2  
# Simulated Fitness data (replace with actual data if available)

set.seed(123)

fitness\_data <- data.frame(

Age = sample(20:60, 50, replace = TRUE),

RunTime = rnorm(50, mean = 12, sd = 2), # minutes for run

RestingPulse = rnorm(50, mean = 70, sd = 10),

MaxPulse = rnorm(50, mean = 180, sd = 15),

Weight = rnorm(50, mean = 70, sd = 15)

)

# a. Scatterplot matrix

pairs(fitness\_data, main = "Scatterplot Matrix of Fitness Data")

# b & c. Correlation matrix among continuous variables

cor\_matrix <- cor(fitness\_data)

print(round(cor\_matrix, 3))

# Find strongest positive correlation (excluding 1s on diagonal)

cor\_matrix[lower.tri(cor\_matrix)] <- NA # Ignore duplicate pairs and diagonal

max\_pos <- max(cor\_matrix, na.rm = TRUE)

pos\_pair <- which(cor\_matrix == max\_pos, arr.ind = TRUE)

# Find strongest negative correlation

min\_neg <- min(cor\_matrix, na.rm = TRUE)

neg\_pair <- which(cor\_matrix == min\_neg, arr.ind = TRUE)

# Variable names

var\_names <- colnames(fitness\_data)

cat("Strongest positive correlation is between", var\_names[pos\_pair[1]], "and", var\_names[pos\_pair[2]], "with value", round(max\_pos, 3), "\n")

cat("Strongest negative correlation is between", var\_names[neg\_pair[1]], "and", var\_names[neg\_pair[2]], "with value", round(min\_neg, 3), "\n")

# d. Interpretation of negative correlation

cat("\nInterpretation: A negative correlation indicates that as one variable increases, the other tends to decrease.\n")  
  
  
9  
python  
import scipy.stats as stats

# Scores of students in SMIP and DBMS

smip\_scores = [70, 46, 94, 34, 20, 86, 18, 12, 56, 64, 42]

dbms\_scores = [60, 66, 90, 46, 16, 98, 24, 8, 32, 54, 62]

# Compute Spearman rank correlation

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

print(f"Spearman rank correlation coefficient: {spearman\_corr:.3f}")

print(f"P-value: {p\_value:.3f}")

# Interpretation

if p\_value < 0.05:

print("The correlation is statistically significant.")

else:

print("The correlation is not statistically significant.")

if spearman\_corr > 0:

print("There is a positive association: as SMIP scores increase, DBMS scores tend to increase.")

elif spearman\_corr < 0:

print("There is a negative association: as SMIP scores increase, DBMS scores tend to decrease.")

else:

print("There is no association between SMIP and DBMS scores.")  
  
  
rcode  
# Scores of students in SMIP and DBMS

smip\_scores <- c(70, 46, 94, 34, 20, 86, 18, 12, 56, 64, 42)

dbms\_scores <- c(60, 66, 90, 46, 16, 98, 24, 8, 32, 54, 62)

# Compute Spearman rank correlation

# The cor.test() function provides both the correlation coefficient and the p-value

spearman\_test\_result <- cor.test(smip\_scores, dbms\_scores, method = "spearman")

# Extract the correlation coefficient and p-value

spearman\_corr <- spearman\_test\_result$estimate

p\_value <- spearman\_test\_result$p.value

cat(sprintf("Spearman rank correlation coefficient: %.3f\n", spearman\_corr))

cat(sprintf("P-value: %.3f\n", p\_value))

# Interpretation

if (p\_value < 0.05) {

cat("The correlation is statistically significant.\n")

} else {

cat("The correlation is not statistically significant.\n")

}

if (spearman\_corr > 0) {

cat("There is a positive association: as SMIP scores increase, DBMS scores tend to increase.\n")

} else if (spearman\_corr < 0) {

cat("There is a negative association: as SMIP scores increase, DBMS scores tend to decrease.\n")

} else {

cat("There is no association between SMIP and DBMS scores.\n")

}  
  
  
10.  
python  
import pandas as pd

import statsmodels.api as sm

# Data

sales = [2, 4, 6, 9, 12, 34, 45]

tv\_budget = [1, 2, 4, 7, 9, 11, 15]

# Create DataFrame

data = pd.DataFrame({

'Sales': sales,

'TV': tv\_budget

})

# Define predictor (independent variable) and response (dependent variable)

X = data['TV']

y = data['Sales']

# Add constant term to predictor (for intercept)

X = sm.add\_constant(X)

# Build linear regression model

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

# Display summary

print(model.summary())  
  
  
rcode  
# Load necessary libraries

# For data manipulation (similar to pandas DataFrame)

# For linear regression model (equivalent to statsmodels.api.OLS)

# Data

sales <- c(2, 4, 6, 9, 12, 34, 45)

tv\_budget <- c(1, 2, 4, 7, 9, 11, 15)

# Create a data frame (equivalent to pandas DataFrame)

data <- data.frame(

Sales = sales,

TV = tv\_budget

)

# Build linear regression model

# In R, the 'lm()' function automatically adds an intercept by default.

# The formula 'Sales ~ TV' means 'Sales' is the dependent variable and 'TV' is the independent variable.

model <- lm(Sales ~ TV, data = data)

# Display summary of the model

# The 'summary()' function in R provides detailed output similar to statsmodels.summary()

summary(model)

# You can also access specific components of the model object if needed:

# coefficients <- coef(model)

# r\_squared <- summary(model)$r.squared

# etc.  
  
  
11.  
# Load required libraries

library(ggplot2)

library(forecast)

library(tseries)

library(lubridate)

# Load dataset

data <- read.csv("australian\_drug\_sales.csv")

# Convert date column to Date format

data$Date <- as.Date(paste0(data$Date, "-01")) # assuming 'Date' is in 'YYYY-MM' format

# Create time series object

sales\_ts <- ts(data$Sales, start = c(year(min(data$Date)), month(min(data$Date))), frequency = 12)

# Plot original time series

plot.ts(sales\_ts, main = "Australian Drug Sales Time Series", ylab = "Sales", xlab = "Time", col = "blue")

# Decompose time series (additive)

decomposed <- decompose(sales\_ts, type = "additive")

plot(decomposed)

# Test for stationarity using Augmented Dickey-Fuller test

adf.test(sales\_ts)

# Differencing to make it stationary (if needed)

diff\_ts <- diff(sales\_ts)

plot.ts(diff\_ts, main = "Differenced Time Series")

# ACF and PACF plots

acf(diff\_ts, main = "ACF of Differenced Series")

pacf(diff\_ts, main = "PACF of Differenced Series")

# Fit ARIMA model

fit <- auto.arima(sales\_ts)

summary(fit)

# Forecast next 12 months

forecasted <- forecast(fit, h = 12)

plot(forecasted)

# Optional: ggplot2 line plot

ggplot(data, aes(x = Date, y = Sales)) +

geom\_line(color = "darkgreen") +

labs(title = "Australian Drug Sales Over Time", x = "Date", y = "Sales") +

theme\_minimal()

12.  
# Load required libraries

library(ggplot2)

library(dplyr)

# Sample dataset (replace with actual dataset from JMP Pro)

# Create a synthetic dataset with 3 groups

set.seed(123)

data <- data.frame(

Group = rep(c("A", "B", "C"), each = 20),

Score = c(rnorm(20, mean = 50, sd = 10),

rnorm(20, mean = 55, sd = 10),

rnorm(20, mean = 60, sd = 10))

)

# View first few rows

head(data)

# ----------------------------

# 1. One-way ANOVA Test

# ----------------------------

anova\_result <- aov(Score ~ Group, data = data)

summary(anova\_result)

# ----------------------------

# 2. Mann-Whitney U Test (Wilcoxon rank-sum)

# ----------------------------

# Pairwise Mann-Whitney U test (between each group pair)

pairwise.wilcox.test(data$Score, data$Group, p.adjust.method = "bonferroni")

# ----------------------------

# 3. Kruskal-Wallis Test

# ----------------------------

kruskal\_result <- kruskal.test(Score ~ Group, data = data)

kruskal\_result

# ----------------------------

# 4. Boxplot for visual comparison

# ----------------------------

ggplot(data, aes(x = Group, y = Score, fill = Group)) +

geom\_boxplot() +

labs(title = "Score Comparison Across Groups", y = "Score") +

theme\_minimal()