

statistical programming assignment 2

by Laurice Keogh

Submission date: 10-May-2024 01:28PM (UTC+0100)

Submission ID: 233121415

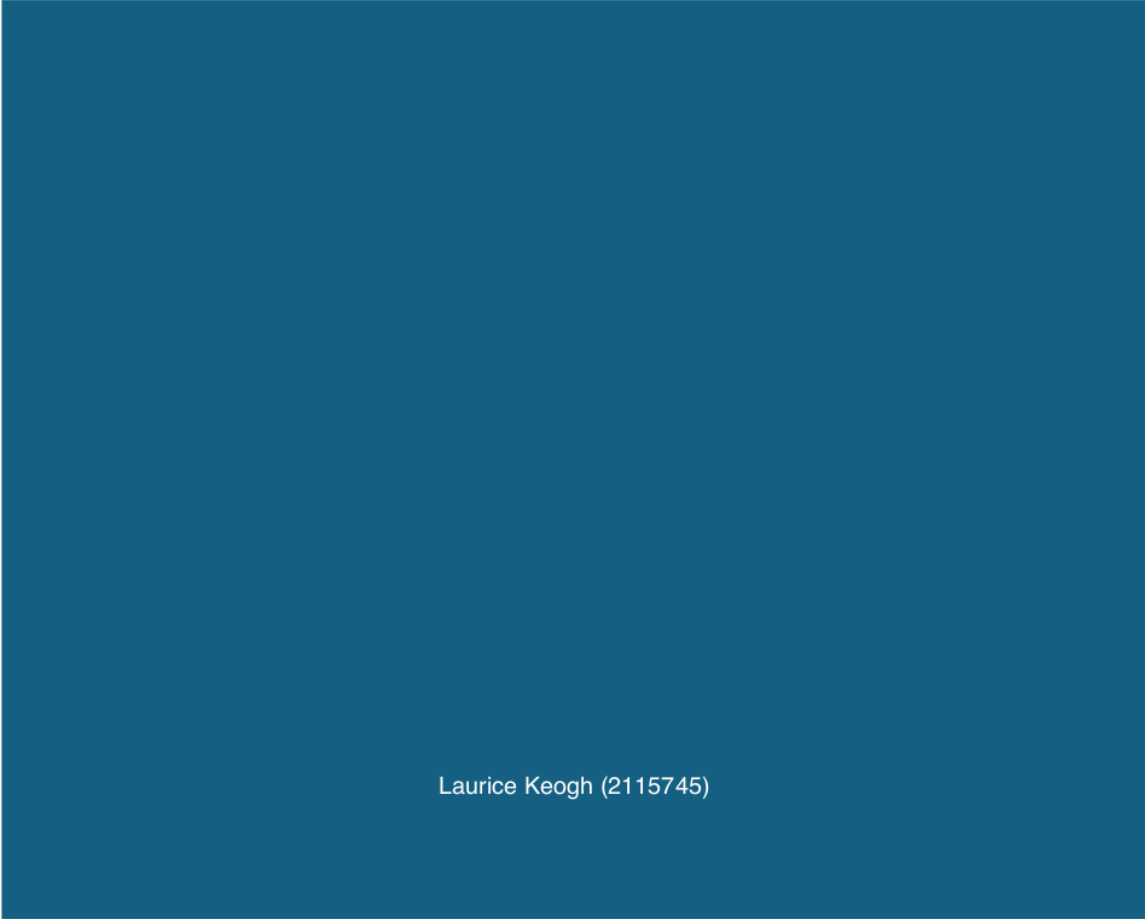
File name: 35959_Laurice_Keogh_statistical_programming_assignment_2_565887_150937502.docx (863.1K)

Word count: 1180

Character count: 5846



STATISTICAL PROGRAMMING COURSEWORK 2 - DOCUMENTATION



Laurice Keogh (2115745)

Contents

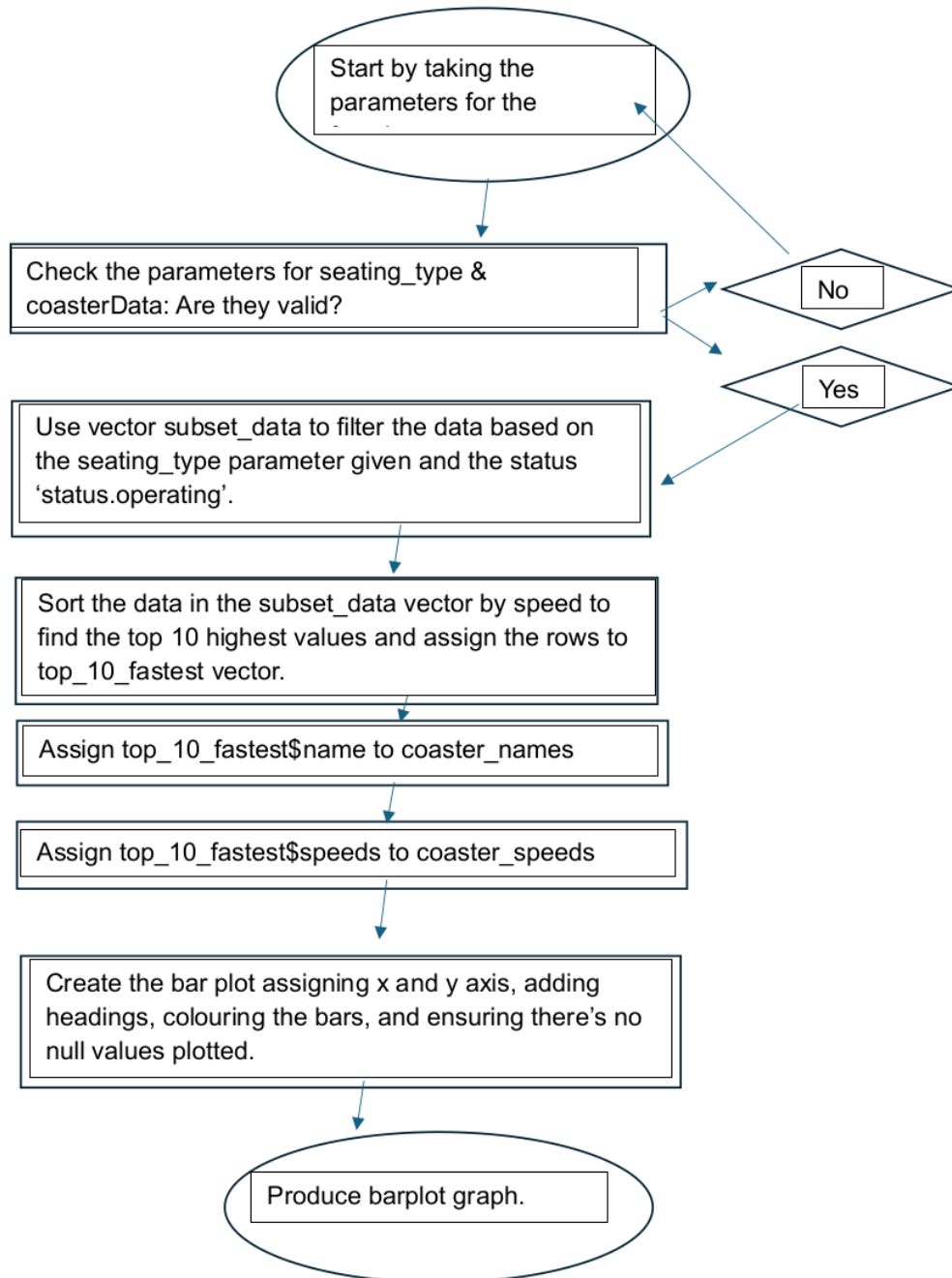
Introduction	1
Code Flow Charts and Explanation	2
Task 3 Flow Chart	2
Filtering applied excel	3
Bar plot graph	3
Explanation	3
Script	4
Task 4 Flow Chart	4
R results	6
Excel results	6
Filtering applied Excel	6
Explanation	6
Script	7
Task 5 Flow Chart	7
Explanation	9
Script	9
Working directory	10
Main script	10
Reflection	10
References	11

Introduction

The following report documents the processes involved in created task 3, 4, and 5. This includes a code explanation, flow charts and methods of checking the results produced. It concludes with a reflection on the challenges faced when creating the code.

Code Flow Charts and Explanation

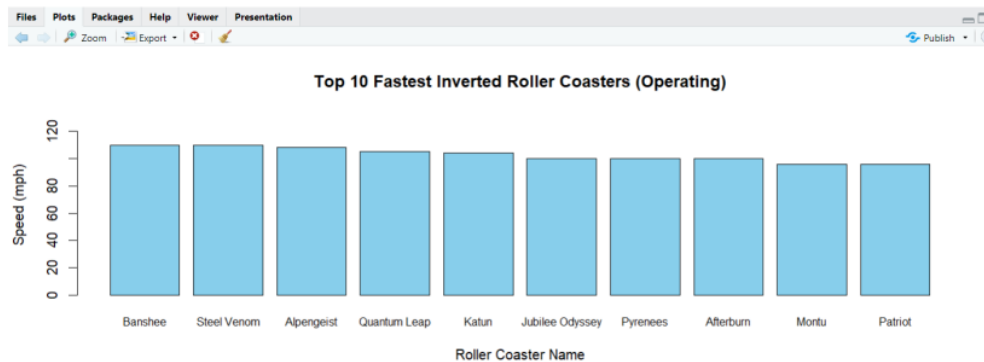
Task 3 Flow Chart



Filtering applied excel

1	name	material	seating	speed	height	length	num_inv	manufa	park	status
8	Banshee	Steel	Inverted	110	51	1260	7	B&M	Kings Island	status.operating
25	Steel Venom	na	Inverted	110	56	192	0	Intamin	Valleyfair	status.operating
39	Alpengeist	Steel	Inverted	108	60		6	B&M	Busch Gardens	status.operating
46	Quantum Leap	Steel	Inverted	105	58	367	3	Vekoma	Sochi-Park	status.operating
51	Katun	Steel	Inverted	104	50	1200	6	B&M	Mirabilandia	status.operating
53	Jubilee Odyssey	Steel	Inverted	100	51	891	5	Vekoma	Fantasy Island	status.operating
56	Pyrenees	Steel	Inverted	100	45	1234	6	B&M	Parque Esplanade	status.operating
72	Afterburn	Steel	Inverted	100	34	900	6	B&M	Carowinds	status.operating
131	Montu	Steel	Inverted	96	46	1214	7	B&M	Busch Gardens	status.operating
133	Patriot	Steel	Inverted	96	46	939	4	B&M	Worlds of Fun	status.operating

Bar plot graph



Explanation

The code works by creating a function which accepts the seating type given from the main script and the coasterData. It checks whether the parameters given are valid and if they are the code continues running. A vector is used to store the data which conforms to the parameters given for seating_type and it also only accepts rows which also have the status 'status.operating'. The data is sorted to give the top 10 highest speed by using the – for descending and head for starting at the beginning, and put into another vector. The names of the fastest coasters are put in one vector whilst the speeds of the fastest coaster are put in another. This is essential to bar plotting. The bar plot is designed with the main heading and axis headings being labelled, and the bars being coloured, whilst also ensuring there are no nulls in the data collected because this could skew the results. The main title of the bar is changeable based on what seating type the user wants due to calling on seating_type instead of specifying a type. An extra 20% space is allocated on the y axis for a clearer visualization.

The data shown on the bar chart is checked with the raw data by using filters in excel such as having only 'Inverted' coasters, only 'status.operating' and ordering the speeds from highest to lowest. The actions performed in the R script have returned the expected result.

Script

```
# Step 3: Display comparison of top 10 fastest inverted roller coasters
displayComparison <- function(seating_type, coasterData) {

  # Subset coasterData for roller coasters of the specified seating type that
  # are operating
  subset_data <- coasterData[coasterData$seating_type == seating_type
                             & coasterData$status == "status.operating", ]

  # Extract the top 10 fastest roller coasters from the subset, ordered by speed
  top_10_fastest <- head(subset_data[order(-subset_data$speed), ], 10)

  # Extract roller coaster names
  coaster_names <- top_10_fastest$name

  # Extract roller coaster speeds
  coaster_speeds <- top_10_fastest$speed

  # Create a barplot displaying the speeds of the top 10 fastest roller coasters
  barplot(coaster_speeds,

          # Set the main title of the plot
          main = paste("Top 10 Fastest", seating_type, "Roller Coasters
                        (Operating)"),

          # Set the label for the x-axis
          xlab = "Roller Coaster Name",

          # Set the label for the y-axis
          ylab = "Speed (mph)",

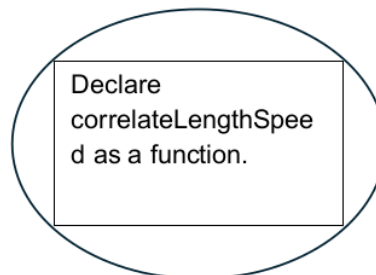
          # Set the color of the bars to skyblue
          col = "skyblue",

          # Set the y-axis limits dynamically based on the maximum speed,
          # with 20% additional space
          ylim = c(0, max(coaster_speeds, na.rm = TRUE) * 1.2),

          # Set the names for the bars to coaster_names
          names.arg = coaster_names, cex.names = 0.8)

}
```

Task 4 Flow Chart



```
graph TD; Start(( )) --> Step1[Store data from coasterData in a vector sit_down_data, where the data is filtered for seating type "sit down" and only completed rows of data, avoiding nulls.]; Step1 --> Step2[Apply the built-in function mean() to sit_down_data$height and store in mean_height vector.]; Step2 --> Step3[Apply the built-in function mean() to sit_down_data$length and store in mean_length vector.]; Step3 --> Step4[Apply the built-in function cor() to sit_down_data$length and sit_down_data$speed to then store in the correlation vector.]; Step4 --> Step5[Store mean height, mean length, and the correlation between length and speed in the data frame lengthSpeed.]; Step5 --> End(( ))
```

Store data from coasterData in a vector sit_down_data, where the data is filtered for seating type "sit down" and only completed rows of data, avoiding nulls.

Apply the built-in function mean() to sit_down_data\$height and store in mean_height vector.

Apply the built-in function mean() to sit_down_data\$length and store in mean_length vector.

Apply the built-in function cor() to sit_down_data\$length and sit_down_data\$speed to then store in the correlation vector.

Store mean height, mean length, and the correlation between length and speed in the data frame lengthSpeed.

Return data frame
lengthSpeed.

R results

	mean_height	mean_length	correlation
1	28.83574	640.3419	0.6582279

Excel results

mean height	26.73529412
mean length	606.1474627
correlation	0.664872903

Filtering applied Excel

1	name	material	seating	speed	height	length	num_inv	manuf	park	status	
2	Goudurix	Steel	Sit Down	75	37	950	7	Vekoma	Parc Aster	status.operating	n
4	Alucinakis	Steel	Sit Down	30	8	250	0	Zamperla	Terra MÃ-t	status.operating	n
5	Anaconda	Wooden	Sit Down	85	35	1200	0	William J. C	Walygator	status.operating	c
6	Azteka	Steel	Sit Down	55	17	500	0	Soquet	Le Pal	status.operating	
9	Big Thunde	Steel	Sit Down	60	22	1500	0	Vekoma	Disneylanc	status.operating	
10	EqWalizer	Steel	Sit Down	76	36	285	3	Vekoma	Walibi RhÃ	status.operating	
11	Calamity N	Steel	Sit Down	48	14	785	0	Vekoma	Walibi Belg	status.operating	
13	Cobra	Steel	Sit Down	76	36	285	3	Vekoma	Walibi Belg	status.operating	
14	Coccinelle	Steel	Sit Down	36	8	360	0	Zierer	Walibi RhÃ	status.operating	
17	Course de	Steel	Sit Down	65	15	450	0	Schwarzko	Nigloland	status.relocated	
21	Eurosat	Steel	Sit Down	60	26	877	0	Mack	Europa Pa	status.retracked	
22	Expedition	Steel	Sit Down	120	53	1220	0	Intamin	Holiday Pa	status.operating	
23	Le Grand c	Steel	Sit Down	50	12	380	0	Soquet	Fraispertu	status.operating	
24	Indiana Jor	Steel	Sit Down	58	18	566	1	Intamin	Disneylanc	status.operating	
26	Cop Car Cl	Steel	Sit Down	60	16	620	2	Intamin	Movie Park	status.closed.definitely	

Explanation

Data that meets the requirements of having only 'sit down' rides and only rows which have no nulls are put into a vector. From this the built in function mean() is used to find the mean height and store it in a vector, and the mean length and store it in a vector. The built in function [2] cor() is then used on the original vector to compare the correlation between the length and the speed and then stored in the correlation vector. It is required for these three vectors to be stored in a dataframe, and the column names have also been assigned. This is then returned as the data frame lengthSpeed.

To check these results the raw data was consulted. The data was filtered for only sit down coasters, and all mention of nulls/blanks removed. The average function was then used on the height and length to find the mean of both figures. The correlation

between the length and speed was calculated using the [3] corrl function. Each of these calculations were slightly incorrect meaning that something may have gone wrong in the calculations in R.

Script

```
# Step 4: Calculate and store the mean height, length, and correlation of length
# and speed for Sit Down roller coasters
correlateLengthSpeed <- function(coasterData) {

  # Filter coasterData for Sit Down roller coasters with no NA fields
  sit_down_data <- coasterData[coasterData$seating_type == "Sit Down"
                                & complete.cases(coasterData), ]

  # Calculate mean height
  mean_height <- mean(sit_down_data$height)

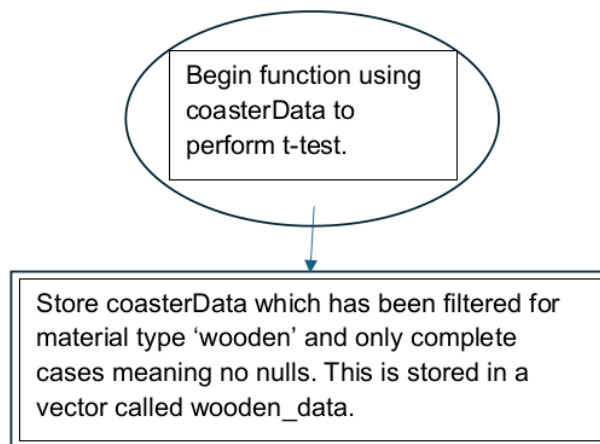
  # Calculate mean length
  mean_length <- mean(sit_down_data$length)

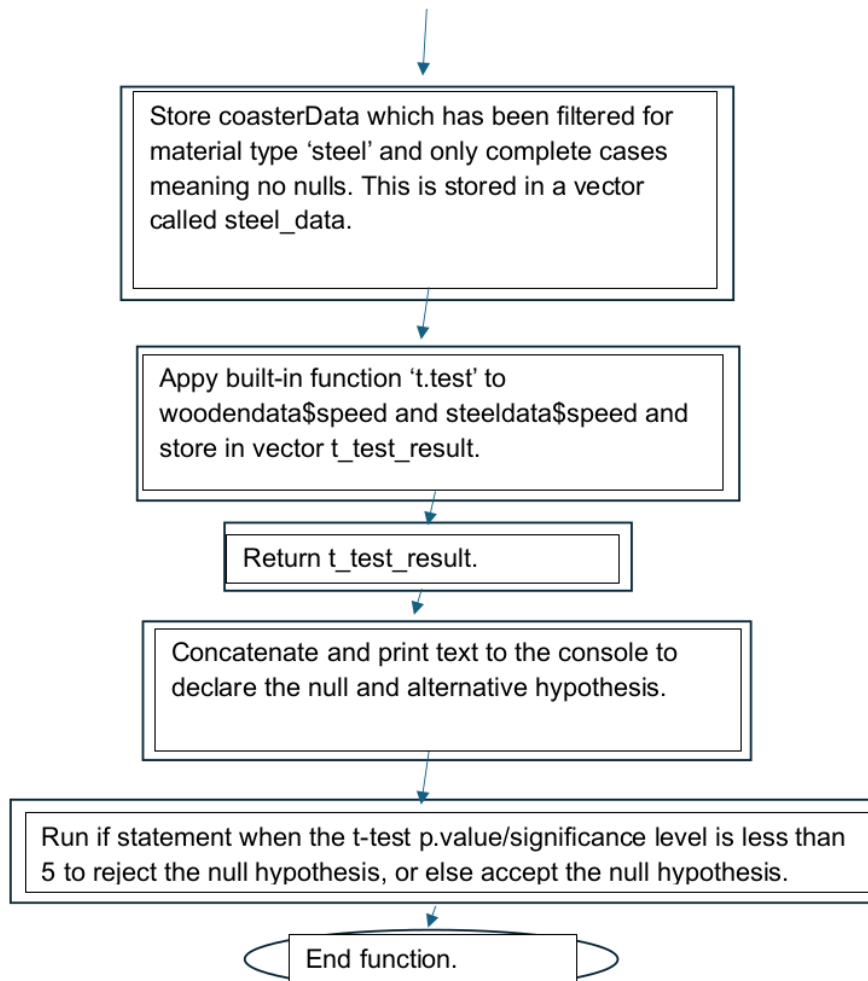
  # Calculate correlation coefficient between length and speed
  correlation <- cor(sit_down_data$length, sit_down_data$speed)

  # Create a data frame to store the results, and define the column names
  lengthSpeed <- data.frame(mean_height = mean_height,
                             mean_length = mean_length,
                             correlation = correlation)

  # Return the data frame
  return(lengthSpeed)
}
```

Task 5 Flow Chart





Console output

```
Console Terminal Background Jobs
R 4.3.2 - ~/University/R/assignment 2/
> comparewoodenSteel(coasterData)

Welch Two Sample t-test

data: wooden_data$speed and steel_data$speed
t = 7.5433, df = 255.91, p-value = 8.002e-13
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 9.398466 16.039342
sample estimates:
mean of x mean of y
82.76471 70.04580

Hypotheses:
Null Hypothesis (H0): There is no significant difference in the mean speeds of wooden and steel roller coasters.
Alternative Hypothesis (H1): There is a significant difference in the mean speeds of wooden and steel roller coasters.

T-test results:
Conclusion: We reject the null hypothesis. There is a significant difference in the mean speeds of wooden and steel roller coasters.
> |
```

Explanation

The function is to compare wooden and steel roller coaster data using the t-test method. The `compareWoodenSteel` function is created using coaster data. Within this the rows that have no nulls and are material type wooden are put into a `wooden_data` vector. The rows with no nulls [2] (`complete.cases`) and material type steel are put into `steel_data` vector. The t-test built in function is applied to both sets of data on the speed and stored in a `t_test_vector`. The t-test data is then returned. The t-tests function is to prove/disprove hypothesis. The null hypothesis for this t-test is that there is no significant difference in the mean speeds. The alternative hypothesis is that there is a significant difference in the mean speeds. This t-test is tails 2. A two tailed test determines if there is a difference between the speeds in any direction. The hypothesis is concatenated to the console and an if else statement returns relevant text depending on the result of the p value in the t-test. In this case the null hypothesis is rejected.

This data can be double checked in excel by using the `t.test` function. The speeds of the wooden coasters must be in one arrangement and the steel in another, then the tails and type must be defined.

Script

```
# Step 5: Compare wooden and steel roller coasters using t-test
compareWoodenSteel <- function(coasterData) {

  # Filter coasterData for wooden roller coasters with no NA fields
  wooden_data <- coasterData[coasterData$material_type == "Wooden"
                             & complete.cases(coasterData), ]
  # Filter coasterData for steel roller coasters with no NA fields
  steel_data <- coasterData[coasterData$material_type == "Steel"
                             & complete.cases(coasterData), ]

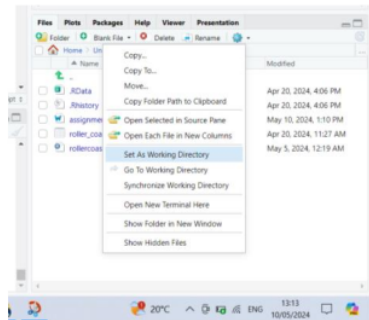
  # Perform independent t-test to compare the mean speeds of wooden and steel
  # roller coasters
  t_test_result <- t.test(wooden_data$speed, steel_data$speed)

  # Print the t-test results
  print(t_test_result)

  # Print hypotheses and conclusions
  cat("Hypotheses:\n")
  cat("Null Hypothesis (H0): There is no significant difference in the mean
      speeds of wooden and steel roller coasters.\n")
  cat("Alternative Hypothesis (H1): There is a significant difference in the
      mean speeds of wooden and steel roller coasters.\n\n")
  cat("T-test results:\n")

  # Check significance level
  if (t_test_result$p.value < 0.05) {
    cat("Conclusion: We reject the null hypothesis. There is a significant
        difference in the mean speeds of wooden and steel roller coasters.\n")
  } else {
    cat("Conclusion: We accept the null hypothesis. There is no significant
        difference in the mean speeds of wooden and steel roller coasters.\n")
  }
}
```

Working directory



```
> setwd("~/University/R/assignment 2")
```

Main script

```
### Main Script:
```

```
# Step 1: Load roller coaster data from CSV file  
coasterData <- loadDataFile()
```

```
# Step 2: Generate data frame for highest operating roller coasters  
highestData <- generateHighestData(coasterData)
```

```
# Step 3: Display comparison of top 10 fastest inverted roller coasters  
displayComparison("Inverted", coasterData)
```

```
# Step 4: Calculate and store the mean height, length, and correlation of length  
# and speed for Sit Down roller coasters  
lengthSpeed<-data.frame(correlateLengthSpeed(coasterData))
```

```
# Step 5: Compare wooden and steel roller coasters using t-test  
compareWoodenSteel(coasterData)
```

Reflection

During the creation of this code, I found getting the parameters working difficult. The seating type is defined in the main script so to get this working I had seating_type, coasterData in the creation of the function and then 'Inverted', coasterData in the main script. It wasn't obvious at first that seating_type and coasterData needed to be in the function creation not just the main script. I struggled to get the results in task 4 to match the results I was seeing in excel. Even though the numbers are close, it could indicate that another filter should have been applied. With more time I would investigate the discrepancies between the two. I did not effectively check if the output of task 5 was correct and a way that I could do this is by applying the [1] Shapiro test. The Shapiro test indicates whether the data is normally distributed or not which could be leading to some skew in the results. I learnt that it is appropriate to have all the working code first and then follow it with the main script for tidiness and ease of understanding. It also differentiates the creation of functions to the

running of functions. In task 3 I would remove the null data earlier on just in case the top 10 coasters include null data and then only 9 for example can be shown on the graph after excluding them afterwards.

ChatGPT was used as an aid whilst completing this assignment. It guided me to effective routes of ensuring that there were no nulls through either `na.rm` which removes from the area specified, or `complete.cases` which ensures there are no nulls in the row anywhere in any column. Other instances where ChatGPT acted as an aid was in explaining what each line of the bar plot was doing. ChatGPT was useful in checking that the code written matched all the requirements and if not offered suggestions although most of the time these suggestions were unnecessary upon deeper inspection of the specifications. AI is a useful tool, but it has its limitations and should be checked.

References

[1]

"What is shapiro test? How to perform it in R -," *ProjectPro*.

<https://www.projectpro.io/recipes/what-is-shapiro-test-perform-it-r>

[2]

ChatGPT, "ChatGPT - Latest News and Chat About AI," *chatgpt.com*, 2024.

<https://chatgpt.com/>

[3]

"CORREL function - Microsoft Support," *support.microsoft.com*.

<https://support.microsoft.com/en-gb/office/correl-function-995dcef7-0c0a-4bed-a3fb-239d7b68ca92#:~:text=The%20CORREL%20function%20returns%20the>

statistical programming assignment 2

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12