

Assignment2

by Osvaldas Stanelis

Submission date: 09-May-2024 07:00PM (UTC+0100)

Submission ID: 232984679

File name: 10895_Osvaldas_Stanelis_Assignment2_565887_1233757514.docx (820.05K)

Word count: 878

Character count: 4835



Statistical Programming
ACCB5021
Osvaldas Stanelis
1802408

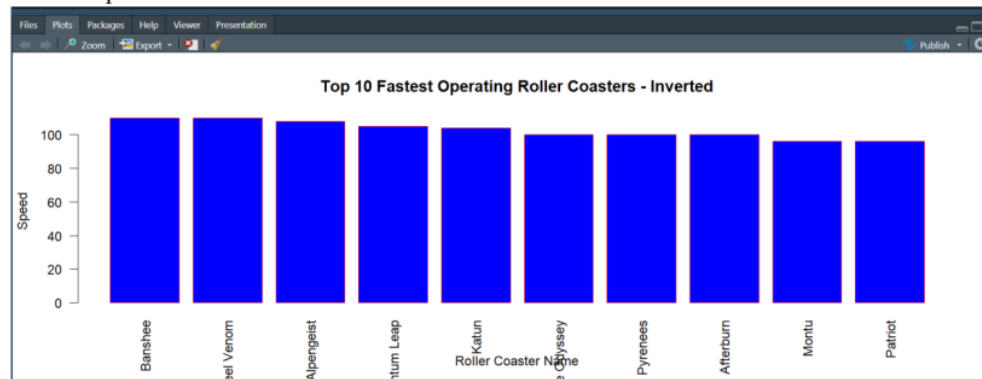
Roller Coaster

Task 2 – Documentation

Task 3 explanation.

The “displayComparison” function in the script is designed to generate a bar chart displaying the top ten fastest operating roller coasters for a given seating type. Here’s a breakdown of how each part of the function operates:

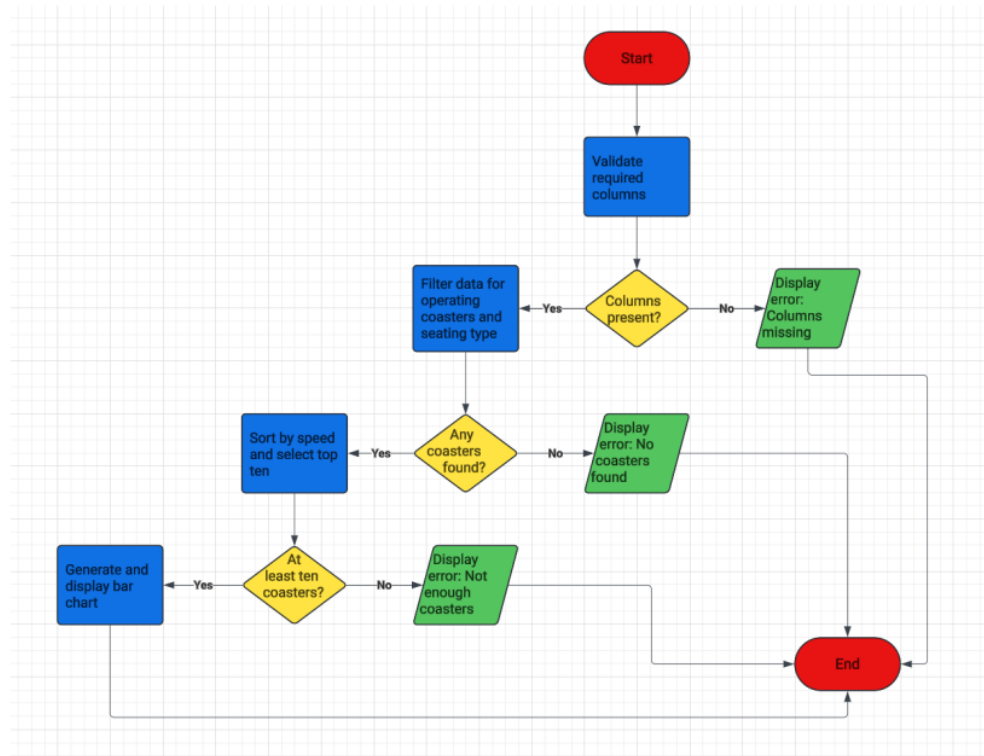
1. Validation of required columns. The function first checks if the necessary columns (name, speed, status, seating_type) exist in the coasterData dataset. This is crucial as the function needs these columns to filter and sort the data correctly. If any columns are missing, the function stops and returns an error message.
2. Data filtering. The function filters coasterData to include only those roller coasters that are both operating (status == “status.operating”) and match the specified seating type (seating_type == seating_type). This step ensures that the analysis is focused only on relevant data.
3. Verification of sufficient data. After filtering, the function checks if there are any coasters that meet the criteria. If no coasters are found, it stops and issues an error message indicating that no operating coasters were found for the specified seating type.
4. Sorting and selecting the “Top ten”. The filtered data is then sorted by speed in descending order. The function selects the top ten coasters from this sorted list. This step is where the function identifies the fastest coasters based on the speed attribute. It also checks if at least one coaster was found in the top ten list. If not, it issues an error that not enough coasters are available to display.
5. Bar chart is generated using the barplot function from R’s base plotting system. The speeds of the coasters are plotted with coaster names on the x-axis, and the chart is titled to reflect the top ten fastest coasters for the specified seating type. Visual aspects like the color and border of the bars are also set here.



```

67 ## TASK 3
68
69 # Function to display speed comparison of the top ten fastest operating roller coasters for a given seating type
70
71 displayComparison<-function(seating_type) {
72 # Validate the presence of required columns
73 requiredColumns<-c("name","speed","status","seating_type")
74 if (!all(requiredColumns %in% names(coasterData))) {
75   stop("One or more required columns are missing")
76 }
77
78 # Filter for operating coasters of the given seating type
79 relevantCoasters<-coasterData[coasterData$status == "status.operating" &
80   coasterData$seating_type == seating_type,]
81
82 # Check if there are enough coasters of the specified seating type
83 if (nrow(relevantCoasters) == 0) {
84   stop("No operating coasters found for the specified seating type")
85 }
86
87 # Sort by speed and select the top ten fastest
88 topCoasters<-relevantCoasters[order(-relevantCoasters$speed),][1:10,]
89
90 # Check if there are enough coasters to display
91 if (nrow(topCoasters) < 1) {
92   stop("Not enough coasters to display")
93 }
94
95 # Create a bar chart of the speeds
96 barplot(topCoasters$speed, names.arg=topCoasters$name, las=2,
97   main=paste("Top 10 Fastest Operating Roller Coasters -", seating_type),
98   xlab="Roller Coaster Name",
99   ylab="Speed",
100   col="blue",
101   border="red")
102 }
103
104 # Display comparison for 'Inverted' seating type
105 displayComparison("Inverted")

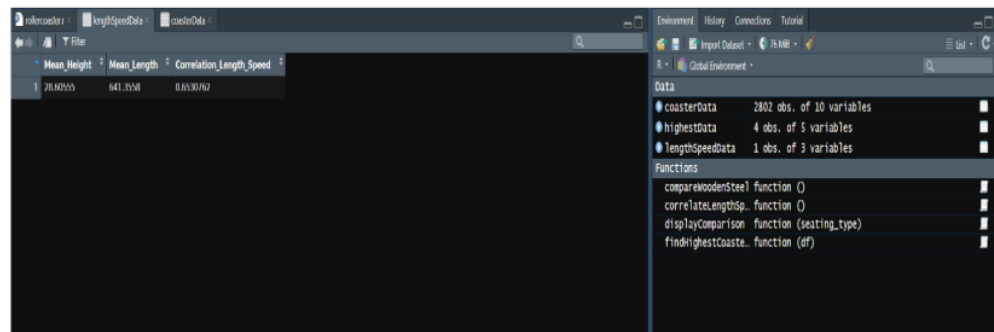
```



Task 4 explanation.

The function “correlateLengthSpeed” is designed to analyse roller coaster data by calculating the mean height, mean length, and the correlation between the length and speed of roller coasters specifically categorized as “Sit Down”. Here’s a breakdown of how function works:

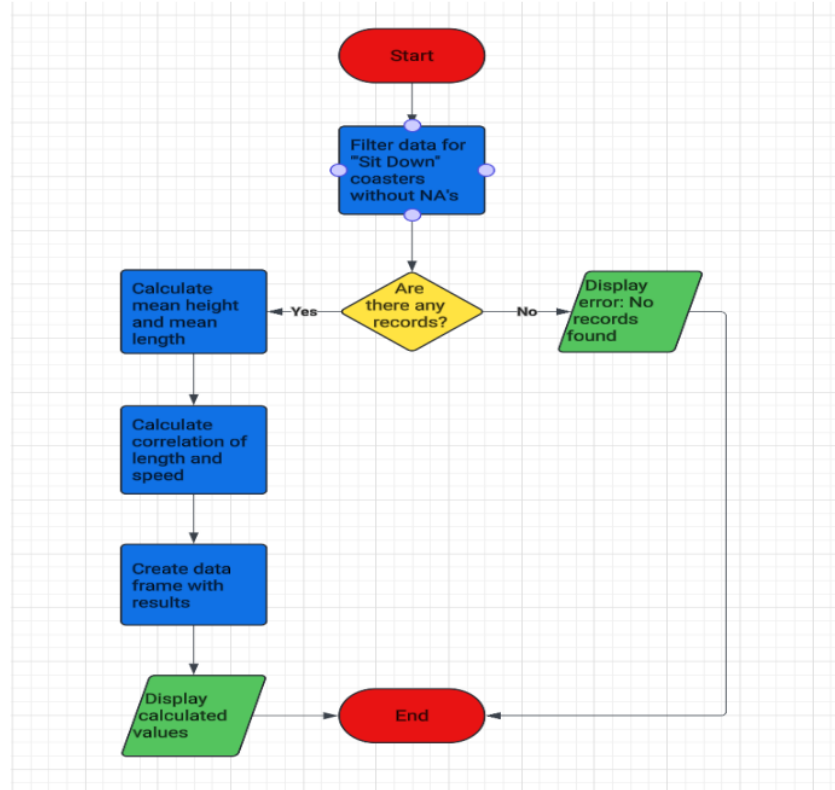
1. The function begins by filtering coasterData to include only roller coasters that are designated as “Sit Down” types and do not have missing values (NA) in the height, length, or speed columns. This ensures that the data used for calculations is complete and relevant to the analysis.
2. Validation of data after filtering, the function checks if the resulting dataset (“filteredData”) has any records left. If there are no records to analyse (“nrow(filteredData) == 0”), the function stops and issues an error message indicating no suitable records were found. This is crucial to avoid errors in subsequent calculations.
3. The function calculates the mean values for height and length using the “mean()” function. These statistics provide basic insights into the average dimensions of the roller coasters being analysed. Correlation coefficient: It calculates the Pearson correlation coefficient between length and speed using the “cor()” function. This statistic helps to understand the relationship between the length of the coasters and their speeds, indicating whether longer coasters tend to be faster or not.
4. The calculated statistics are then compiled into a new data frame named “lengthSpeed”. This data frame includes columns for mean height, mean length, and the correlation coefficient, neatly organizing the results for easy interpretation and presentation.
5. Finally, the function returns the “lengthSpeed” data frame, making the results available for further use or display.



```

107 ## TASK 4
108
109 # Function to calculate mean height, mean length, and correlation of length and speed
110 correlateLengthSpeed<-function() {
111 # Filter coasterData for "Sit Down" roller coasters with no NA in height, length, or speed
112 filteredData<-coasterData[coasterData$seating_type == "Sit Down" &
113                             !is.na(coasterData$height)&
114                             !is.na(coasterData$length)&
115                             !is.na(coasterData$speed),]
116
117 # Check if there are enough coasters to perform analysis
118 if (nrow(filteredData) == 0) {
119     stop("No suitable records found to perform correlation analysis")
120 }
121
122 # Calculate mean height and mean length
123 meanHeight<-mean(filteredData$height)
124 meanLength<-mean(filteredData$length)
125
126 # Calculate the correlation coefficient between length and speed
127 correlationCoefficient<-cor(filteredData$length, filteredData$speed)
128
129 # Create a data frame to store these values
130 lengthSpeed<-data.frame([
131     Mean_Height=meanHeight,
132     Mean_Length=meanLength,
133     Correlation_Length_Speed=correlationCoefficient])
134
135 # Return the data frame
136 return(lengthSpeed)
137 }
138
139 # Execute the function and store the result in a variable
140 lengthSpeedData<-correlateLengthSpeed()
141
142 # Print the resulting data frame
143 print(lengthSpeedData)
144

```



Task 5 explanation.

The function “compareWoodenSteel” is designed to perform a statistical T-test to compare the speeds of wooden and steel roller coasters, testing whether there is a significant difference between these two categories. Here’s a step-by-step explanation of how the function operates.

1. Data filtering for wooden coasters and steel coasters. Filters the “coasterData” for entries where the “material_type” is “Wooden” and there are no missing (NA) values in the speed column. Similarly filters the “coasterData” for entries where the “material_type” is “Steel” and there are no missing (NA) values in the speed column.
2. The function checks if there are enough data points (coasters) for both wooden and steel categories to perform a meaningful T-test. If either category has zero valid data points after filtering, the function stops and displays an error message indicating insufficient data.
3. Performs a two-sided T-test between the speed distributions of wooden and steel roller coasters using the “t.test()” function. This test determines if there is a statistically significant difference between the two means at a 95% confidence level.
4. Hypotheses articulation clearly states the null hypothesis (H0) and the alternative hypothesis (H1) regarding the mean speeds.

Test results display prints the results of the t-test, including the p-value, test statistic, confidence interval, and mean speeds of both groups.

Conclusion based on P-value. Interprets the P-value to decide whether to reject or not reject the null hypothesis. If the P-value is less than or equal to 0.05, it concludes a significant difference (reject H0); otherwise, it concludes no significant difference (do not reject H0).

```
> # Call the function to perform the analysis
> compareWoodenSteel()
H0: There is no difference in the mean speeds of wooden and steel roller coasters.
H1: There is a difference in the mean speeds of wooden and steel roller coasters.

T-test results:

      welch Two Sample t-test

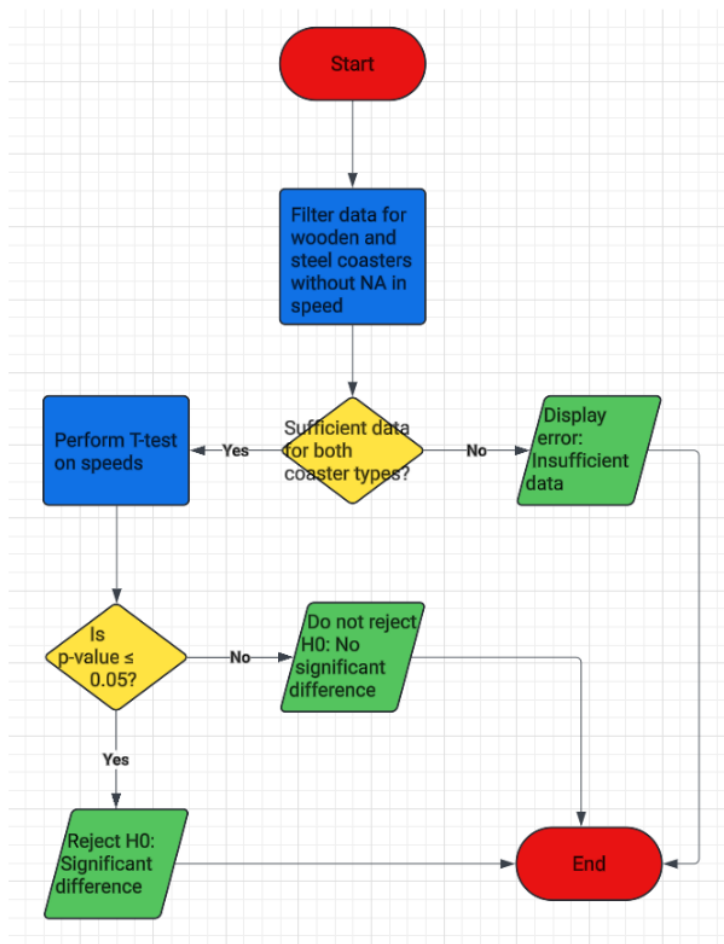
data:  woodenCoasters and steelCoasters
t = 8.2243, df = 277.44, p-value = 7.657e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 10.21077 16.63700
sample estimates:
mean of x mean of y
 82.30719  68.68330

Conclusion: Reject H0. There is a statistically significant difference in the speeds of wooden and steel roller coasters at the p = 0.05 level.
```

```

145 ## TASK 5
146
147 # Function to perform a t-test to compare the speeds of wooden and steel roller coasters
148 compareWoodenSteel<-function() {
149   # Filter coasterData for wooden and steel roller coasters with no NA in speed
150   woodenCoasters<-coasterData[coasterData$material_type == "wooden" &
151     !is.na(coasterData$speed), "speed"]
152   steelCoasters<-coasterData[coasterData$material_type == "steel" &
153     !is.na(coasterData$speed), "speed"]
154
155   # Check if there are enough coasters to perform analysis
156   if (length(woodenCoasters) == 0 || length(steelCoasters) == 0) {
157     stop("not enough data to perform t-test for one or both coaster types.")
158   }
159
160   # Perform the T-test
161   testResult<-t.test(woodenCoasters, steelCoasters,
162     alternative="two.sided", conf.level=0.95)
163
164   # Print the hypotheses and conclusion
165   cat("H0: There is no difference in the mean speeds of wooden and steel roller coasters.\n")
166   cat("H1: There is a difference in the mean speeds of wooden and steel roller coasters.\n\n")
167   cat("T-test results:\n")
168   print(testResult)
169
170   # Interpret the results
171   if (testResult$p.value <= 0.05) {
172     cat("\nConclusion: Reject H0. There is a statistically significant difference in the speeds of wooden and steel roller coasters at the p = 0.05 level.\n")
173   } else {
174     cat("\nConclusion: Do not reject H0. There is no statistically significant difference in the speeds of wooden and steel roller coasters at the p = 0.05 level.\n")
175   }
176 }
177
178 # Call the function to perform the analysis
179 compareWoodenSteel()

```



References

- [1] Dr. N. Whitehead, Class lecture, Topic: “Statistical Programming”, ACCB5021, University of Wales Trinity Saint David, Swansea, May 8, 2024.
- [2] “ChatGPT,” OpenAI. [Online]. Available: <https://chatgpt.com> [Accessed: May 8, 2024].

Assignment2

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8