|  |  |  |  |
| --- | --- | --- | --- |
| Date: | **30-07-2020** | Name: | **Varun G Shetty** |
| Course: | **Coursera** | USN: | **4AL17EC093** |
| Topic: | Basic Statestics | Semester & Section: | **6th & ‘B’** |
| GitHub Repository: | **Varunshetty4** |  |  |

**Report-**

R Language:

Importing the Packages:  Imagine we want to do some great plotting and we want to use ggplot2 for it. If we want to do so, we need to take 2 steps:

1. Install the package ggplot2 using install.packages("ggplot2")
2. Load the package ggplot2 using library(ggplot2) or require(ggplot2)

Dim(): The first value returned by dim() is the number of cases (rows) and the second value is the number of variables (columns).

Str(): Using the str() function we can look at the structure of a dataset. str() takes the name of the data set as its first argument. The output shows the variable names, their type, and the values of the first observations.

Levels():  You can see the levels of a factor variable by using the function levels().  you can select a specific variable using either $ or [,], If you need to check the variables in the data set, remember that you can always use the str() function in your console.

Recording variable: For instance, if we want to make a new variable that categorises people over age 18 as "adult"", we might enter: yourdata$newvariable[yourdata$age > 18] <- "adult" This assigns the value "adult" to the variable newvariable, for all cases where age is greater than 18.

Frequency Examining: Frequency tables show you how often a given value occurs. To look at a frequency table of the data in R, use the function table(). The top row of the output is the value, and the bottom row is the frequency of the value.

Making a Bargraph: a bar graph, using the function barplot(). The first argument of barplot() is a vector containing the heights of each bar. These heights correspond to the proportional frequencies of a desired measure in your data. You can obtain this information using the table() function.

The first argument of barplot() was a vector of the bar heights. Following this, we can add arguments to format the graph as necessary. For instance, barplot(height, argument1, argument2). Here we are going to add a label to the y axis using the argument ylab = "name here", and x axis labels to the bars using the argument names.arg = "vector of names here".

Histograms: It can be useful to plot frequencies as histograms to visualize the spread of our data. Let's make a histogram of dataset using the function hist(). The first argument of hist() is vector of values for which the histogram is desired. Following this, we can add arguments to format the graph as necessary. For instance, hist(variable, argument1, argument2).

In your script, write a code to produce a histogram of the number of carburetors in each car using the variable carb of the data set mtcars. Make the title of this histogram "Carburetors" by adding the argument main = "title\*" inside your hist() function.

In the same way as we added a title argument to hist(), we can change the scale of the y-axis through adding the argument ylim followed by the range we want (e.g. for a scale from 0 to 50, we would say ylim = c(0,50)). We can also label the x-axis using the argument xlab = "title", or change the colour of the bars to blue with the argument col = "blue".

Mean and Median: We can measure the mean and median of a variable using the functions mean() and median()

Mode: Sometimes it is useful to look at the the most frequent value in a data set, known as the 'mode'. R doesn't have a standard function for mode, but we can calculate the mode easily using the table() function

When you have a large data set, the output of table() might be too long to manually identify which value is the mode. In this case it can be useful to use the sort() function, which arranges a vector or factor into ascending order. (You can add the argument decreasing = TRUE to sort() if you want to arrange it in to descending order.)

Range: The range of a variable is the difference between the highest and lowest value. We can find these values using max() and min() on the variables of our choice.

Quratiles: You can calculate the quartiles in your data set using the function quantile(). The output of quantile() gives you the lowest value, first quartile, second quartile, third quartile and highest value. 25% of your data lies below the first quartile value, 50% lies below the second quartile, and 75% lies below the third quartile value.

IQR and Boxplot: To better visualise your data's quartiles you can create a boxplot using the function boxplot() (in the same way as you used hist() and barplot()). Similarly, you can calculate the interquartile range manually by subtracting the value of the third quartile from the value of the first quartile, or we can use the function IQR() on your variable of interest. Let's try out making a boxplot and calculating the interquartile range of the mtcars variable qsec.

In the boxplot you created you can see a circle above the boxplot. This indicates an outlier. We can calculate an outlier as a value 1.5 \* IQR above the third quartile, or 1.5 \* IQR below the first quartile. Let's try it out with the qsec variable from mtcars.

Standard Deviation: We can also measure the spread of data through the standard deviation. You can calculate these using the function sd()