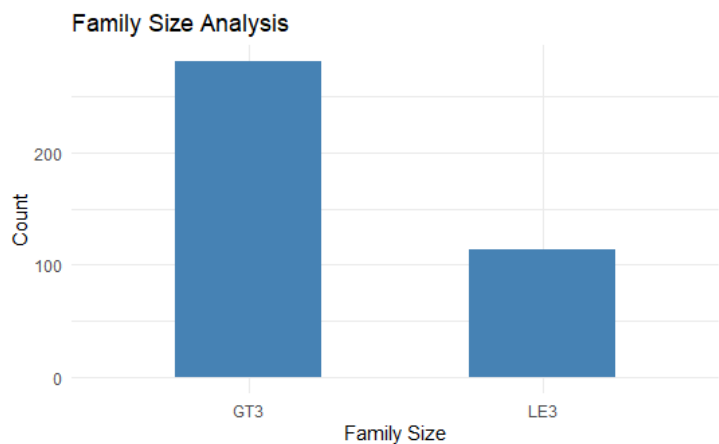
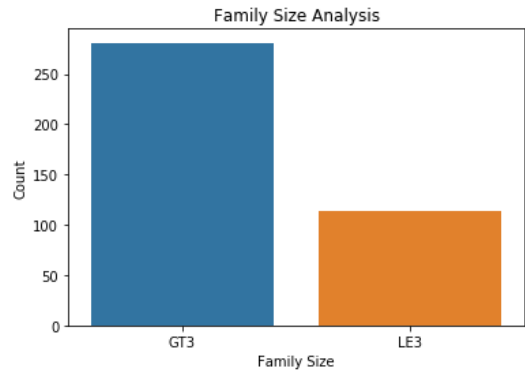
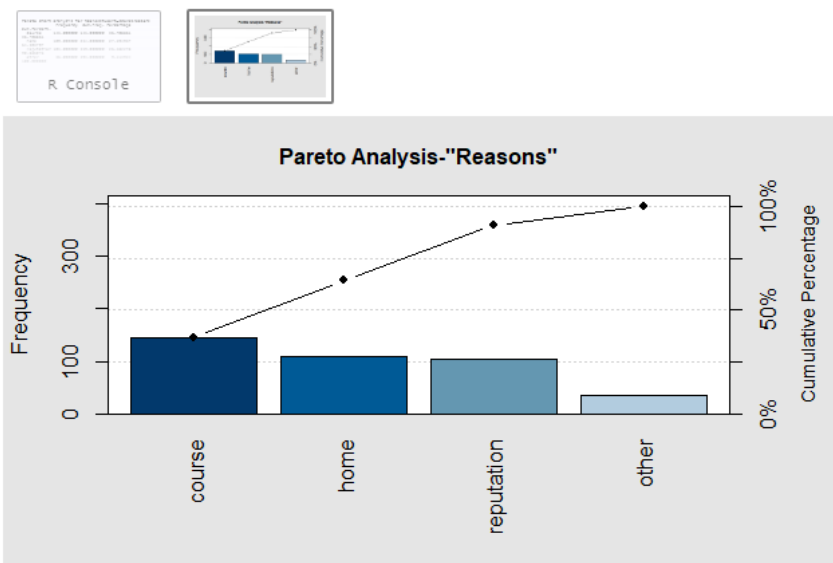
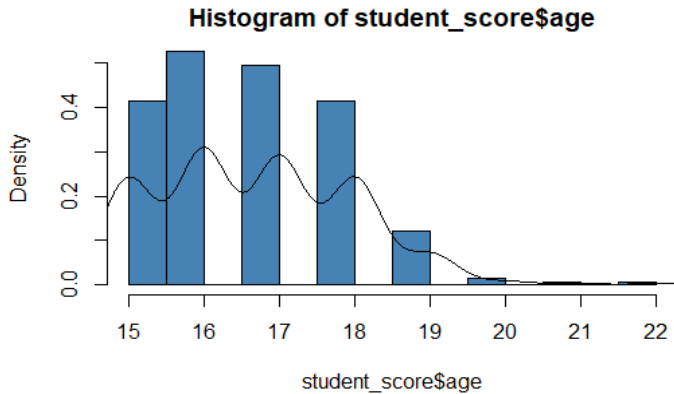
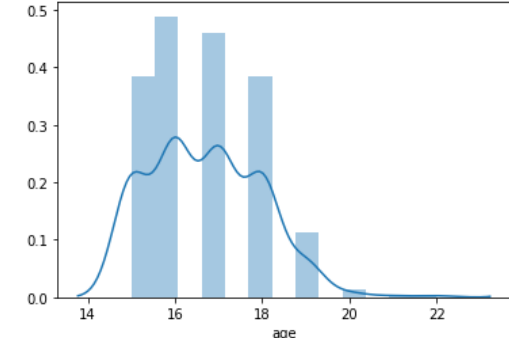
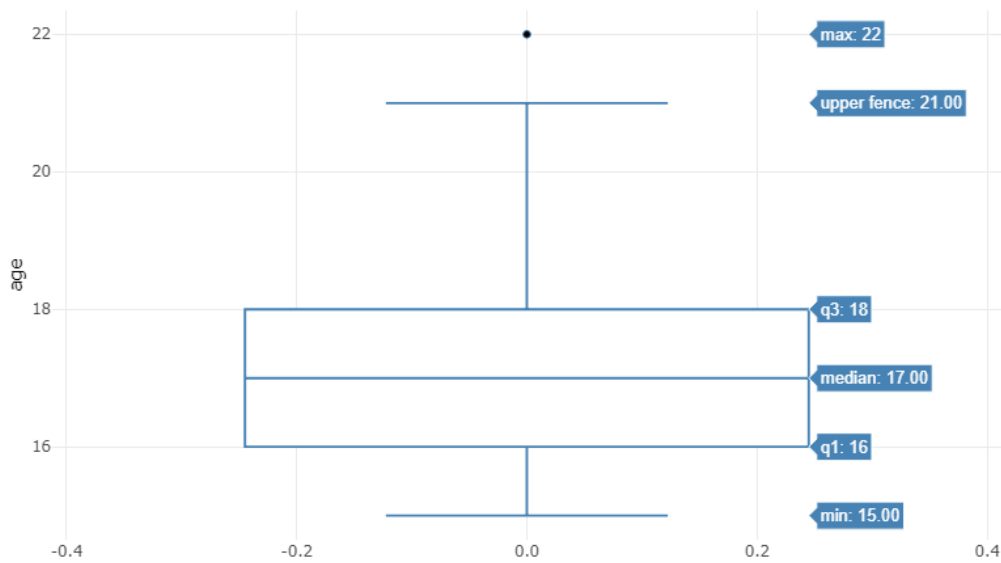
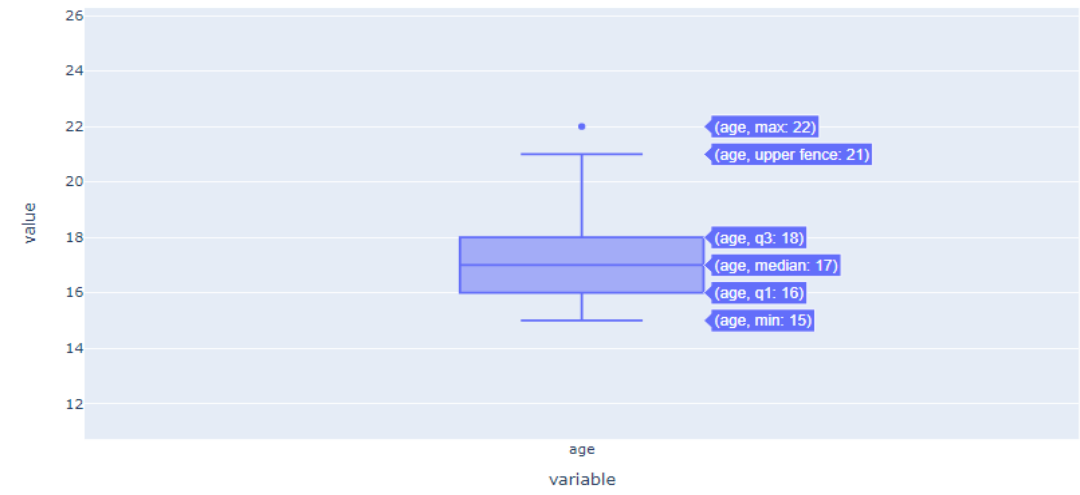


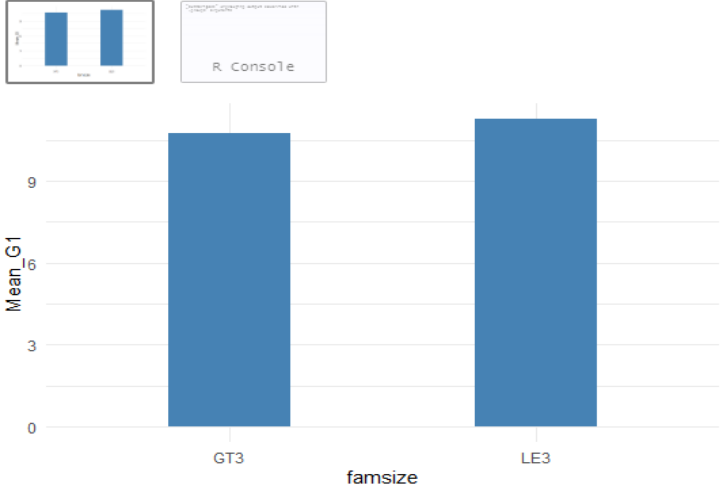
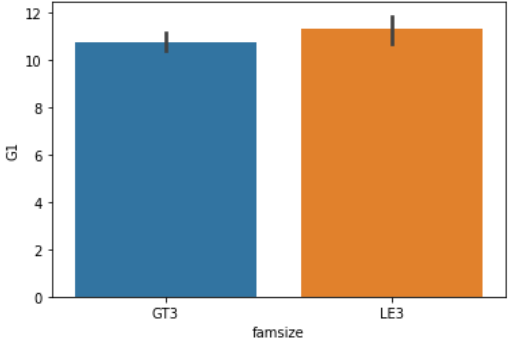
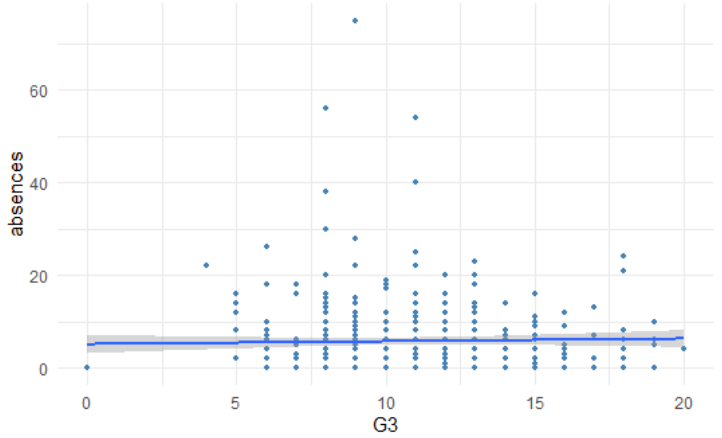
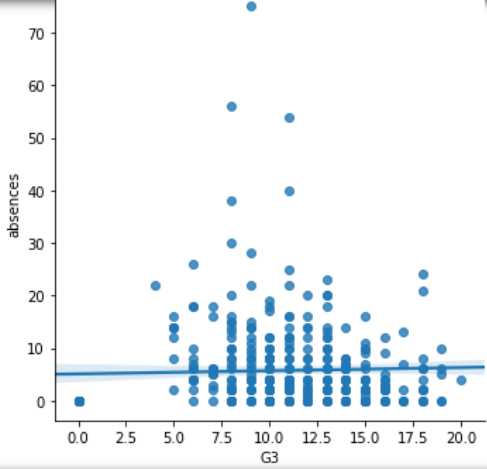


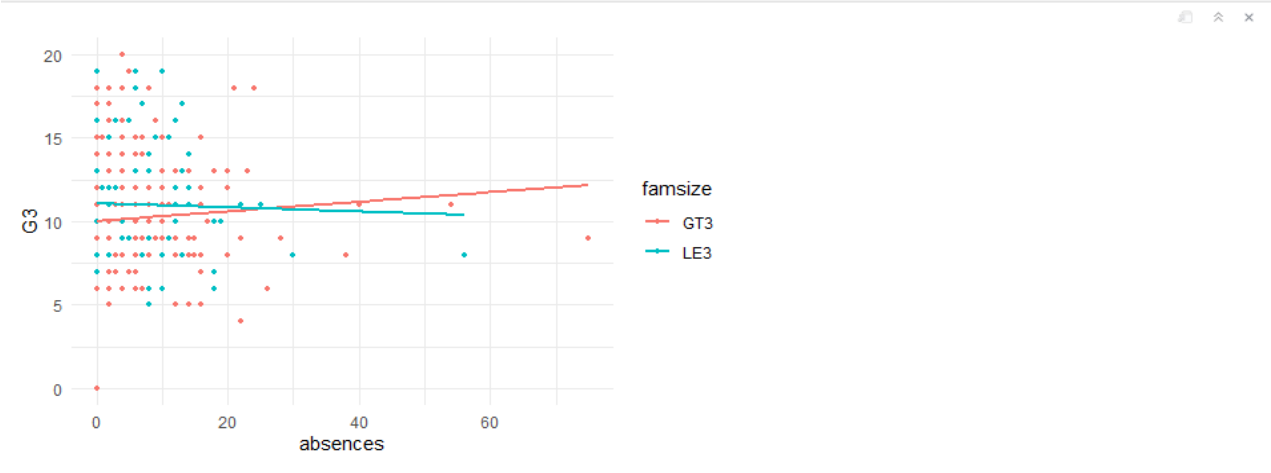
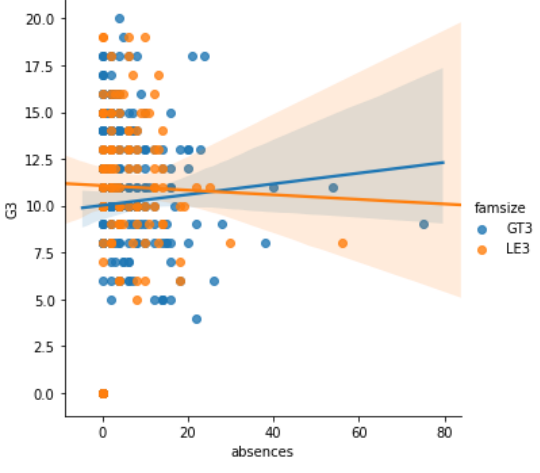
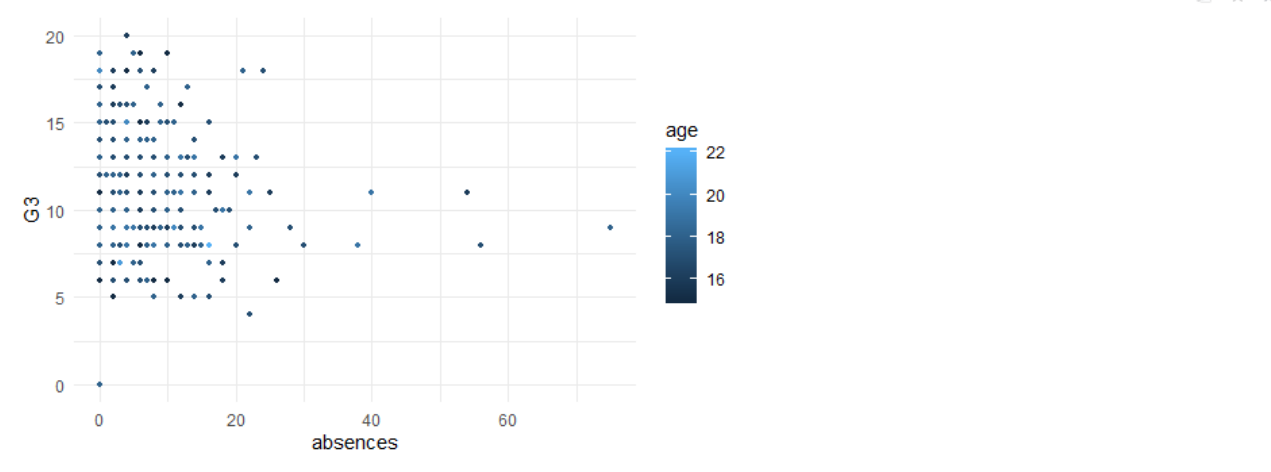
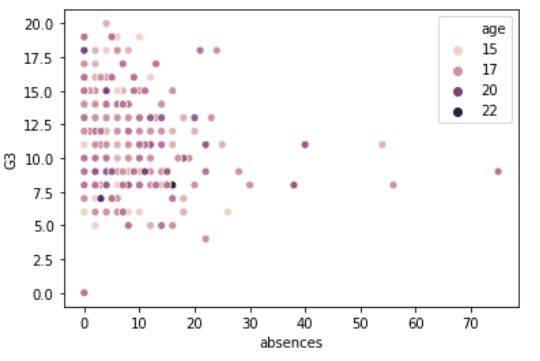
Function	R	Python
Univariate Analysis – Categorical Column		
1. Find unique values 2. Count of each category 3. Top category	<p>Bar chart</p> <pre>ggplot(student_score, aes(x=famsize)) + geom_bar(stat='count', fill="steelblue", width = 0.5) + labs(x="Family Size", y="Count", title="Family Size Analysis") + theme_minimal()</pre> <p>Bar Plot when applied to the raw data without any transformation, finds the frequency of the values in the categorical column.</p> <pre>##{r fig.height=3.5} #Find unique values #Count of each category #Top category  ggplot(student_score, aes(x=famsize)) + geom_bar(stat='count', fill="steelblue", width = 0.5) + labs(x="Family Size", y="Count", title="Family Size Analysis") + theme_minimal()</pre> 	<p>Count plot</p> <pre>sns.countplot(student_score['famsize']); plt.xlabel("Family Size"); plt.ylabel("Count"); plt.title("Family Size Analysis");</pre> <p>Count Plot counts the frequency of the values in the categorical column.</p> <pre>#Find unique values #Count of each category #Top category  sns.countplot(student_score['famsize']); plt.xlabel("Family Size"); plt.ylabel("Count"); plt.title("Family Size Analysis");</pre> 
4. Percentage of each category	<p>Pareto chart</p> <pre>pareto.chart(table(student_score\$reason), main = 'Pareto Analysis-"Reasons"')</pre> <pre>##{r fig.height=4} #Percentage of each category pareto.chart(table(student_score\$reason), main = 'Pareto Analysis-"Reasons"')</pre> 	<p>Pareto Chart is not in-built in Python. Percentage of the values are to be calculated using the frequency. The percentage is to be added as a separate column. A bar plot and a line plot can be plotted (as sub plots or overlay) to achieve the similar version of Pareto in R.</p> <p>Alternatively, user defined functions can also be written.</p>

Function	R	Python
Univariate Analysis – Numerical Column		
5. Density distribution	<p><b>Histogram</b></p> <pre>hist(student_score\$age, col="steelblue", freq = FALSE) lines(density(student_score\$age), col='black',lwd=1.5)</pre> <p>Plotting only the histogram does not give the density distribution. Using ggplot + geom_hist() also gives the similar plot, with more flexibility to change the bin size, etc.</p> <pre>{r fig.height=3.5} #Density distribution hist(student_score\$age, col="steelblue", freq = FALSE) lines(density(student_score\$age), col='black',lwd=1.5)</pre>  <p>The histogram shows the density distribution of age. The x-axis is labeled 'student_score\$age' and ranges from 15 to 22. The y-axis is labeled 'Density' and ranges from 0.0 to 0.4. The bars are steelblue, and a black density curve is overlaid.</p>	<p><b>Histogram</b></p> <pre>sns.distplot(student_score['age']);</pre> <p>Plotting histogram using the matplotlib library (.hist) function, does not give the density distribution.</p> <pre>sns.distplot(student_score['age']);</pre>  <p>The histogram shows the density distribution of age. The x-axis is labeled 'age' and ranges from 14 to 22. The y-axis ranges from 0.0 to 0.5. The bars are steelblue, and a black density curve is overlaid.</p>
6. Five point summary	<p><b>Boxplot</b></p> <pre>ggplot(data=student_score, aes(y=age)) + geom_boxplot(color="steelblue") + theme_minimal()</pre> <p>Without using the plotly library, the boxplot will be static. The values will not be displayed on 'mouse over'.</p> <pre>{r fig.height=2, fig.width=2} #Five point Summary b = ggplot(data=student_score, aes(y=age)) + geom_boxplot(color="steelblue") + theme_minimal() ggplotly(b)</pre>  <p>The boxplot shows the five-point summary of age. The x-axis is labeled 'age' and ranges from -0.4 to 0.4. The y-axis ranges from 16 to 22. The box is steelblue. Labels on the right indicate: max: 22, upper fence: 21.00, q3: 18, median: 17.00, q1: 16, and min: 15.00.</p>	<p><b>Boxplot</b></p> <pre>plt.boxplot(student_score['age']);</pre> <p>Without using the plotly library, the boxplot will be static. The values will not be displayed on 'mouse over'. Using the seaborn library, the boxplot will be horizontally oriented by default.</p> <pre>#Five point Summary px.box(student_score['age'])  #Alternate ways sns.boxplot(student_score['age'], orient="v"); plt.boxplot(student_score['age']);</pre>  <p>The boxplot shows the five-point summary of age. The x-axis is labeled 'age variable' and ranges from 12 to 26. The y-axis is labeled 'value' and ranges from 12 to 26. The box is steelblue. Labels on the right indicate: (age, max: 22), (age, upper fence: 21), (age, q3: 18), (age, median: 17), (age, q1: 16), and (age, min: 15).</p>

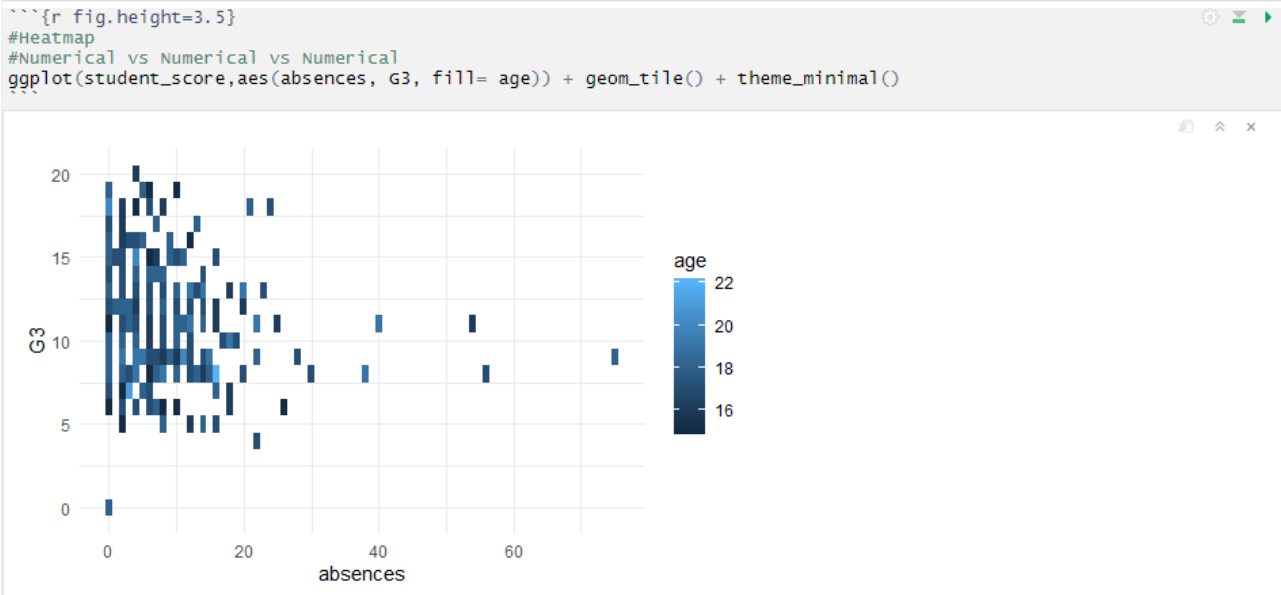
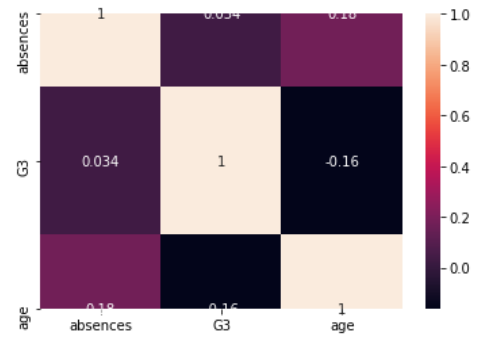
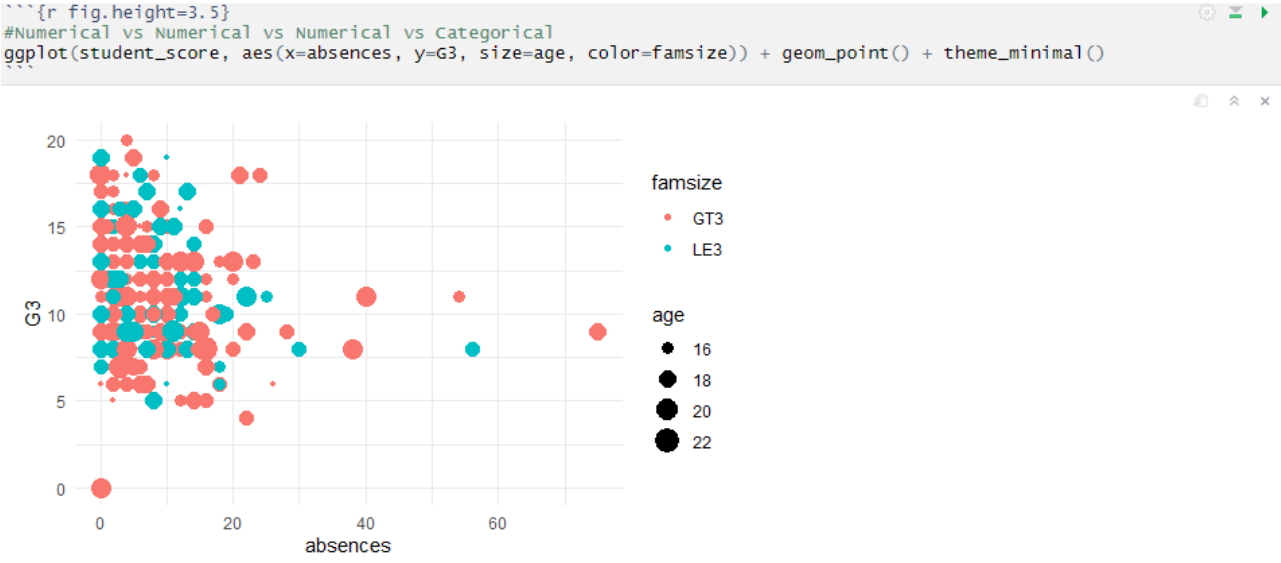
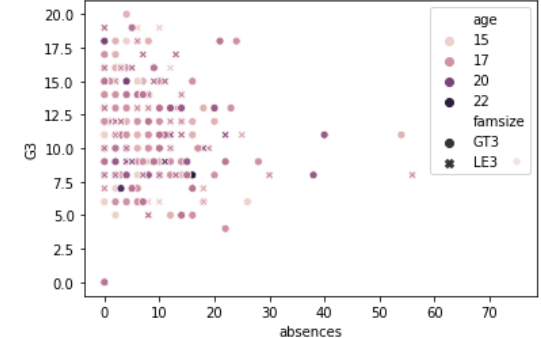
Function	R	Python
Univariate Analysis – Text Column		
7. Word Cloud – text data	<p>Word Cloud</p> <pre>wordcloud(words_freq_imp\$words, words_freq_imp\$Freq, min.freq = 50, colors = brewer.pal(8,"Dark2"), random.order = FALSE, random.color = TRUE)</pre> <p>Initial transformation to be done on the text data and stopwords removed, before word cloud is built.</p> <pre>```{r fig.height=4} #word CLOUD wc = read.csv("netflix_titles.csv") wc\$description = gsub("[^a-z 0-9]", " ", wc\$description) wc\$description = as.character(wc\$description) row_words = strsplit(wc\$description, " ") words = unlist(row_words) words_freq = table(words) words_freq = as.data.frame(words_freq) words_freq = words_freq %&gt;% arrange(-Freq) stop_words = stopwords() words_freq_imp = words_freq %&gt;% filter(!words %in% stop_words)  wordcloud(words_freq_imp\$words, words_freq_imp\$Freq, min.freq = 50, colors = brewer.pal(8,"Dark2"), random.order = FALSE, random.color = TRUE) ```</pre> 	<p>Word Cloud</p> <pre>WordCloud(stopwords=stopwords, background_color='white').generate(str(wc['description']))</pre> <p>‘Stopwords’ are the words that do not carry any meaning. They are to be removed before word cloud is built.</p> <pre>: wc = pd.read_csv("netflix_titles.csv")  : #word CLOUD stopwords = set(STOPWORDS) wordcloud = WordCloud(stopwords=stopwords, background_color='white').generate(str(wc['description'])) plt.imshow(wordcloud) plt.axis("off") plt.tight_layout(pad = 0) plt.show()</pre> 

Function	R	Python
Bivariate Analysis		
8. Categorical vs Numerical	<div>Barplot</div> <div><pre>student_score %&gt;% group_by(famsize) %&gt;% summarise(Mean_G1=mean(G1, na.rm=T)) %&gt;% ggplot(aes(famsize,Mean_G1)) + geom_bar(stat='identity',fill='steelblue', width=0.4) + theme_minimal()</pre></div> <div>Data has to be transformed and then fed into the bar chart function to plot the mean of the numerical column vs the categorical column.</div> <div><pre>##{r fig.height=3.5} #Categorical vs Numerical student_score %&gt;% group_by(famsize) %&gt;% summarise(Mean_G1=mean(G1, na.rm=T)) %&gt;% ggplot(aes(famsize,Mean_G1)) + geom_bar(stat='identity',fill='steelblue', width=0.4) + theme_minimal()</pre></div> <div></div>	<div>Barplot</div> <div><pre>sns.barplot(student_score['famsize'], student_score['G1']);</pre></div> <div>The Bar Plot in Python automatically calculates the mean of the numerical column. It is not required to transform the data. The black line on top of each bar indicates the 'confidence interval'. Bar Plots can also be used to calculate the 'sum' or any other statistic using the 'estimator' parameter and the confidence interval values will change accordingly.</div> <div><pre>#Categorical vs Numerical sns.barplot(student_score['famsize'], student_score['G1']);</pre></div> <div></div>
9. Numerical vs Numerical	<div>Scatter plot</div> <div><pre>ggplot(student_score, aes(x=G3, y=absences)) + geom_point(color='steelblue', size=1) + geom_smooth(method="lm") + theme_minimal()</pre></div> <div>The regression line is achieved using the geom_smooth. Changing the value of the 'method' parameter changes the type of fit.</div> <div><pre>##{r fig.height=3.5} #Numerical vs Numerical ggplot(student_score, aes(x=G3, y=absences)) + geom_point(color='steelblue', size=1) + geom_smooth(method="lm") + theme_minimal()</pre></div> <div></div>	<div>Scatter plot</div> <div><pre>sns.lmplot(x='G3', y='absences', data=student_score);</pre></div> <div>The scatterplot gives only the scatter plot. The lmplot gives the scatter plot along with the regression line fit.</div> <div><pre>#Numerical vs Numerical sns.lmplot(x='G3', y='absences', data=student_score); #Alternate method sns.scatterplot(student_score['G3'], student_score['absences']); #does not produce the slope</pre></div> <div></div>

Function	R	Python
10. Date vs Numerical Column – Trending data – Timeline vs numerical data	<p><u>Line Chart</u></p> <p><b><code>student_score %&gt;% ggplot(aes(Dt,absences)) + geom_line(color='steelblue') + theme_classic()</code></b></p> <p>Like the other charts, categorical or numerical variables can be included to group the y-values.</p> <pre>##{r fig.height=3.5} #Line Chart - Trending data - timeline vs numerical data student_score\$Dt = as.Date(student_score\$D, "%d-%m-%Y") student_score %&gt;% ggplot(aes(Dt,absences)) + geom_line(color='steelblue') + theme_classic()</pre>	<p><u>Line Chart</u></p> <p><b><code>sns.lineplot(student_score['D'], student_score['absences']);</code></b></p> <p>Like the other charts, categorical or numerical variables can be included (as hue) to group the y-values.</p> <pre>#Line chart - Trending data - timeline vs numerical data plt.figure(figsize=(10,5)) sns.lineplot('D','absences', data=student_score);  positions = (50, 150, 250, 350); labels = ("Jan 1971", "Apr 1971", "Jul 1971", "Oct 1971"); plt.xticks(positions, labels);</pre>
Multivariate Analysis		
11. Categorical vs Categorical vs Numerical	<p><u>Grouped Bar plot</u></p> <p><b><code>student_score %&gt;% group_by(health, famsize) %&gt;% summarise(Mean_G1=mean(G1, na.rm=T)) %&gt;% ggplot(aes(health,Mean_G1, fill=famsize)) + geom_bar(stat='identity', width=0.4, position="dodge") + theme_minimal()</code></b></p> <p>The value of the 'position' parameter determines whether the plot is a grouped bar plot or a stacked bar plot or a 100% stacked bar plot.</p> <pre>##{r fig.height=3.5} #Categorical vs Categorical vs Numerical student_score %&gt;% group_by(health, famsize) %&gt;% summarise(Mean_G1=mean(G1, na.rm=T)) %&gt;% ggplot(aes(health,Mean_G1, fill=famsize)) + geom_bar(stat='identity', width=0.4, position="dodge") + theme_minimal()</pre>	<p><u>Grouped Bar Plot</u></p> <p><b><code>sns.barplot(student_score['health'], student_score['G1'], hue=student_score['famsize']);</code></b></p> <p>The barplot function with the hue parameter outputs a grouped barplot.</p> <pre>#Categorical vs Categorical vs Numerical sns.barplot(student_score['health'], student_score['G1'], hue=student_score['famsize']);</pre>

Function	R	Python
12. Categorical vs Numerical vs Numerical	<p><b>Multiple Scatter Plot</b>  <code>ggplot(student_score, aes(x=absences, y=G3, color=famsize)) + geom_point(size=1) + geom_smooth(method="lm", se=F) + theme_minimal()</code>            The color parameter can be substituted the value of a categorical column. Data is automatically grouped by the categorical column.</p> 	<p><b>Multiple Scatter Plot</b>  <code>sns.lmplot(x='absences', y='G3', data=student_score, hue='famsize');</code>            'Hue' parameter is used for the purpose of grouping the 'y – axis' values. This parameter can take categorical or numerical values. It can take more than one values too.</p> 
13. Numerical vs Numerical vs Numerical	<p><b>Scatter Plot</b>  <code>ggplot(student_score, aes(x=absences, y=G3, color=age)) + geom_point(size=1) + theme_minimal()</code>            The same 'color' parameter as seen above can be used to substitute numerical values.</p> 	<p><b>Scatter Plot</b>  <code>sns.scatterplot(x='absences', y='G3', data=student_score, hue='age');</code>            'Hue' parameter takes numerical values in this case, to group the values of 'y-axis'.</p> 



Function	R	Python
14. Numerical vs Numerical vs Numerical	<p><b>Heatmap</b></p> <pre>ggplot(student_score,aes(absences, G3, fill= age)) + geom_tile() + theme_minimal()</pre> <p>Heatmap can also be produced using the 'heatmap' function. Though the chart shows the visual co-relation of the three numerical variables, it does not give the numerical value of co-relation.</p> 	<p><b>Heatmap</b></p> <pre>sns.heatmap(student_score[['absences','G3','age']].corr(), annot=True);</pre> <p>Heatmap is used to find correlation between two or more numeric columns.</p> <pre>#Heatmap #Numerical vs Numerical vs Numerical sns.heatmap(student_score[['absences','G3','age']].corr(), annot=True);</pre> 
15. Numerical vs Numerical vs Numerical vs Categorical	<p><b>Scatter Plot</b></p> <pre>ggplot(student_score, aes(x=absences, y=G3, size=age, color=famsize)) + geom_point() + theme_minimal()</pre> <p>When the grouping of 'y-axis' to be done with more than one column, then we can use both 'color' and 'size' parameter. Though not mandatory, as a best practise, the 'color' is used with categorical values and 'size' parameter is used with numerical values.</p> 	<p><b>Scatter Plot</b></p> <pre>sns.scatterplot(x='absences', y='G3', data=student_score, hue='age', style='famsize');</pre> <p>Though 'hue' parameter can take more than one values, 'style' is also one more way to represent grouped values.</p> <pre>#Numerical vs Numerical vs Numerical vs Categorical sns.scatterplot(x='absences', y='G3', data=student_score, hue='age', style='famsize');</pre> 

Function	R	Python
16. Faceting	<p>Faceting is done when y-values are to be grouped by more than two variables. More than one faceting variable can be used. Different chart types can also be produced.</p> <pre> {r fig.height=3.5} #Faceting student_score %&gt;% group_by(health, famsize, internet) %&gt;% summarise(Absences=mean(absences, na.rm=T)) %&gt;% ggplot(aes(health,Absences, fill=internet)) + geom_bar(stat='identity', width=0.4, position="dodge") + theme_classic() + facet_grid(~famsize) </pre> <p>The R faceted bar chart displays 'Absences' on the y-axis (0 to 9) against 'health' on the x-axis (1 to 5). The chart is faceted by 'famsize' into two panels: 'GT3' and 'LE3'. Within each health category, there are two bars representing 'internet' usage: 'no' (red) and 'yes' (teal). The 'GT3' panel shows higher absence values for health 1-5 compared to the 'LE3' panel.</p>	<p>The 'col' parameter is used for faceting in python. This is another way to group y-values, other than 'hue' and 'style'. Different chart types can also be produced.</p> <pre> #Faceting sns.catplot('health','absences', hue='internet', col='famsize', data=student_score, kind="bar"); </pre> <p>The Python faceted bar chart displays 'absences' on the y-axis (0 to 16) against 'health' on the x-axis (1 to 5). The chart is faceted by 'famsize' into two panels: 'famsize = GT3' and 'famsize = LE3'. Within each health category, there are two bars representing 'internet' usage: 'no' (blue) and 'yes' (orange). The 'GT3' panel shows higher absence values for health 1-5 compared to the 'LE3' panel.</p>