[Cover Page]

Contents

1. Task 1: Visualising Outliers

"The objective of this task is to Practice visualising outliers in a dataset using various plots"

Dataset: Use the "airquality" dataset available in R.

Dataset Preview

(Importing necessary libraries to perform this task)

library(ggplot2)

(loading and previewing the "air quality dataset)

data("airquality")

print(airquality)

Output

```
> print(airquality)
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
3      12     149 12.6   74     5   3
4      18     313 11.5   62     5   4
5      NA      NA 14.3   56     5   5
6      28      NA 14.9   66     5   6
7      23     299  8.6   65     5   7
8      19      99 13.8   59     5   8
9       8      19 20.1   61     5   9
10     NA     194  8.6   69     5  10
11      7      NA  6.9   74     5  11
12     16     256  9.7   69     5  12
13     11     290  9.2   66     5  13
14     14     274 10.9   68     5  14
15     18      65 13.2   58     5  15
16     14     334 11.5   64     5  16
17     34     307 12.0   66     5  17
18      6      78 18.4   57     5  18
19     30     322 11.5   68     5  19
20     11      44  9.7   62     5  20
21      1       8  9.7   59     5  21
22     11     320 16.6   73     5  22
23      4      25  9.7   61     5  23
24     32      92 12.0   61     5  24
25     NA      66 16.6   57     5  25
26     NA     266 14.9   58     5  26
27     NA      NA  8.0   57     5  27
28     23      13 12.0   67     5  28
29     45     252 14.9   81     5  29
30    115     223  5.7   79     5  30
31     37     279  7.4   76     5  31
32     NA     286  8.6   78     6   1
33     NA     287  9.7   74     6   2
34     NA     242 16.1   67     6   3
35     NA     186  9.2   84     6   4
36     NA     220  8.6   85     6   5
37     NA     264 14.3   79     6   6
38     29     127  9.7   82     6   7
39     NA     273  6.9   87     6   8
40     71     291 13.8   90     6   9
41     39     323 11.5   87     6  10
42     NA     259 10.9   93     6  11
43     NA     250  9.2   92     6  12
44     23     148  8.0   82     6  13
45     NA     332 13.8   80     6  14
46     NA     322 11.5   79     6  15
47     21     191 14.9   77     6  16
48     37     284 20.7   72     6  17
49     20      37  9.2   65     6  18
50     12     120 11.5   73     6  19
51     13     137 10.3   76     6  20
52     NA     150  6.3   77     6  21
53     NA      59  1.7   76     6  22
54     NA      91  4.6   76     6  23
55     NA     250  6.3   76     6  24
56     NA     135  8.0   75     6  25
57     NA     127  8.0   78     6  26
58     NA      47 10.3   73     6  27
59     NA      98 11.5   80     6  28
60     NA      31 14.9   77     6  29
61     NA     138  8.0   83     6  30
62    135     269  4.1   84     7   1
63     49     248  9.2   85     7   2
64     32     236  9.2   81     7   3
65     NA     101 10.9   84     7   4
66     64     175  4.6   83     7   5
67     40     314 10.9   83     7   6
68     77     276  5.1   88     7   7
69     97     267  6.3   92     7   8
70     97     272  5.7   92     7   9
71     85     175  7.4   89     7  10
72     NA     139  8.6   82     7  11
73     10     264 14.3   73     7  12
74     27     175 14.9   81     7  13
75     NA     291 14.9   91     7  14
76      7      48 14.3   80     7  15
77     48     260  6.9   81     7  16
78     35     274 10.3   82     7  17
```

119	NA	153	5.7	88	8	27
120	76	203	9.7	97	8	28
121	118	225	2.3	94	8	29
122	84	237	6.3	96	8	30
123	85	188	6.3	94	8	31
124	96	167	6.9	91	9	1
125	78	197	5.1	92	9	2
126	73	183	2.8	93	9	3
127	91	189	4.6	93	9	4
128	47	95	7.4	87	9	5
129	32	92	15.5	84	9	6
130	20	252	10.9	80	9	7
131	23	220	10.3	78	9	8
132	21	230	10.9	75	9	9
133	24	259	9.7	73	9	10
134	44	236	14.9	81	9	11
135	21	259	15.5	76	9	12
136	28	238	6.3	77	9	13
137	9	24	10.9	71	9	14
138	13	112	11.5	71	9	15
139	46	237	6.9	78	9	16
140	18	224	13.8	67	9	17
141	13	27	10.3	76	9	18
142	24	238	10.3	68	9	19
143	16	201	8.0	82	9	20
144	13	238	12.6	64	9	21
145	23	14	9.2	71	9	22
146	36	139	10.3	81	9	23
147	7	49	10.3	69	9	24
148	14	20	16.6	63	9	25
149	30	193	6.9	70	9	26
150	NA	145	13.2	77	9	27
151	14	191	14.3	75	9	28
152	18	131	8.0	76	9	29
153	20	223	11.5	68	9	30

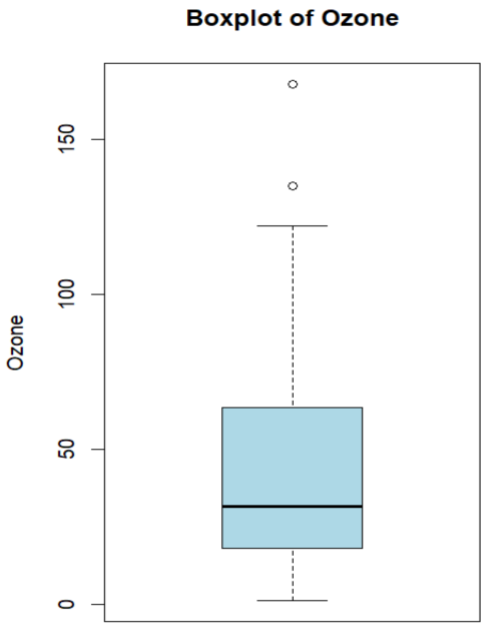
79	61	285	6.3	84	7	18
80	79	187	5.1	87	7	19
81	63	220	11.5	85	7	20
82	16	7	6.9	74	7	21
83	NA	258	9.7	81	7	22
84	NA	295	11.5	82	7	23
85	80	294	8.6	86	7	24
86	108	223	8.0	85	7	25
87	20	81	8.6	82	7	26
88	52	82	12.0	86	7	27
89	82	213	7.4	88	7	28
90	50	275	7.4	86	7	29
91	64	253	7.4	83	7	30
92	59	254	9.2	81	7	31
93	39	83	6.9	81	8	1
94	9	24	13.8	81	8	2
95	16	77	7.4	82	8	3
96	78	NA	6.9	86	8	4
97	35	NA	7.4	85	8	5
98	66	NA	4.6	87	8	6
99	122	255	4.0	89	8	7
100	89	229	10.3	90	8	8
101	110	207	8.0	90	8	9
102	NA	222	8.6	92	8	10
103	NA	137	11.5	86	8	11
104	44	192	11.5	86	8	12
105	28	273	11.5	82	8	13
106	65	157	9.7	80	8	14
107	NA	64	11.5	79	8	15
108	22	71	10.3	77	8	16
109	59	51	6.3	79	8	17
110	23	115	7.4	76	8	18
111	31	244	10.9	78	8	19
112	44	190	10.3	78	8	20
113	21	259	15.5	77	8	21
114	9	36	14.3	72	8	22
115	NA	255	12.6	75	8	23
116	45	212	9.7	79	8	24
117	168	238	3.4	81	8	25
118	73	215	8.0	86	8	26

1.1 Create a boxplot for the "Ozone" variable.

Code:

```
boxplot(airquality$Ozone, main="Boxplot of Ozone", ylab="Ozone", col="lightblue", na.rm=TRUE)
```

Output:

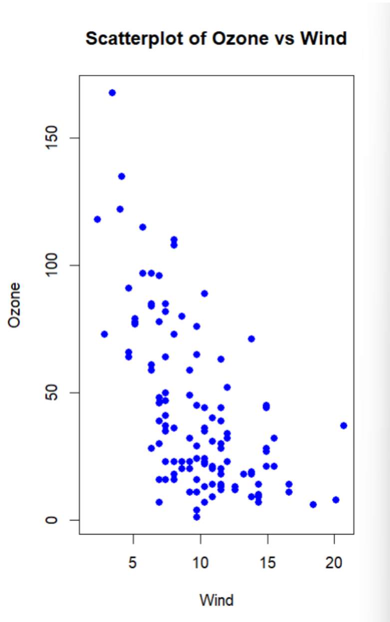


1.2 Create a scatterplot of “Ozone” against “Wind”.

Code:

```
plot(airquality$Wind, airquality$Ozone, main="Scatterplot of Ozone vs Wind",
xlab="Wind", ylab="Ozone", pch=19, col="blue", na.rm=TRUE)
```

Output:

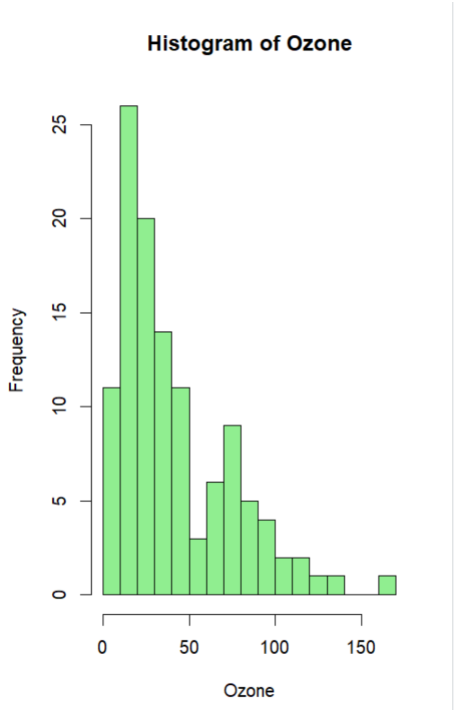


1.3 Create a histogram for the “Ozone” variable.

Code:

```
hist(airquality$Ozone, main="Histogram of Ozone", xlab="Ozone", col="lightgreen",
breaks=20, na.rm=TRUE)
```

Output:



1.4 Create a density Plot for the "Ozone" variable. Discuss the findings and identify potential outliers in the dataset.

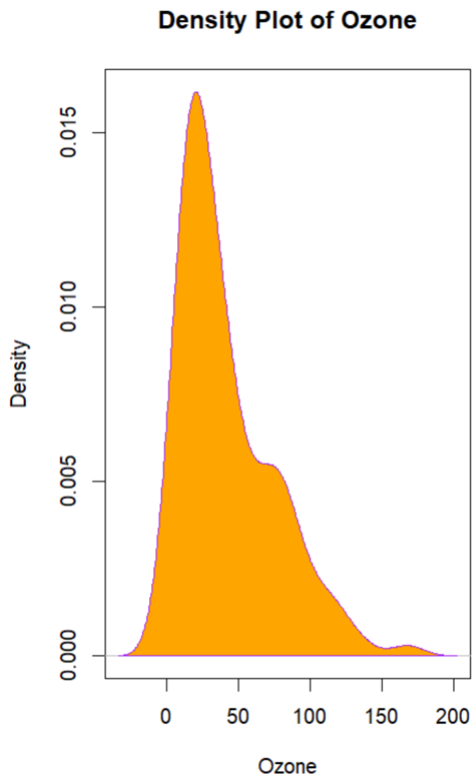
Code:

```
ozone_density <- density(airquality$Ozone, na.rm=TRUE)

plot(ozone_density, main="Density Plot of Ozone", xlab="Ozone", ylab="Density", col="purple")

polygon(ozone_density, col="orange", border="purple")
```

Output:



2. Task 1.1: Detecting Outliers

“The objective of this task is to apply different outlier detection methodsto the same dataset and compare the results.“

Dataset: Use the "Ozone" variable from the "airquality" dataset.

(importing necessary libraries)

library(dplyr)

library(outliers)

(loading dataset and extracting ozone variable)

data("airquality")

ozone <- airquality$Ozone

2.1 Detect outliers using the z-score method with a threshold of 2.

Z-score method:

The Z-score method identifies outliers by measuring how far each data point is from the mean, in terms of standard deviations. If a data point's Z-score is very high or very low, it is considered an outlier. This method is useful when the data is normally distributed.

Code:

ozone_z <- (ozone - mean(ozone, na.rm=TRUE)) / sd(ozone, na.rm=TRUE)

z_score_outliers <- ozone[abs(ozone_z) > 2]

cat("Z-score outliers:\n")

print(z_score_outliers)

Output:

```
Z-score outliers:
> print(z_score_outliers)
[1] 115 135 122 110 168 118
```

Outlier:

115, 135, 122, 110, 168, 118

1. This method found several outliers: 115, 135, 122, 110, 168, and 118.
2. It identifies outliers based on how many standard deviations a data point is from the mean.
3. It tends to find more outliers because it looks at how far data points are from the average.
4. It might be affected by data that is not evenly spread out.

2.2 Detect outliers using the IQR method with a multiplier of 1.5.

IQR method:

The Interquartile Range (IQR) method identifies outliers by looking at the spread of the middle 50% of the data. The IQR is the range between the first quartile (25th percentile) and the third quartile (75th percentile). Any data point that falls below the first quartile minus 1.5 times the IQR or above the third quartile plus 1.5 times the IQR is considered an outlier. This method is robust because it is not influenced by extreme values.

Code:

```
Q1 <- quantile(ozone, 0.25, na.rm=TRUE)

Q3 <- quantile(ozone, 0.75, na.rm=TRUE)

IQR <- Q3 - Q1

lower_bound <- Q1 - 1.5 * IQR

upper_bound <- Q3 + 1.5 * IQR

iqr_outliers <- ozone[ozone < lower_bound | ozone > upper_bound]

cat("IQR outliers:\n")

print(iqr_outliers)
```

Output:

```
IQR outliers:
> print(iqr_outliers)
[1] 135 168
```

Outliers:

135, 168

1. This method found 135 and 168 as outliers.
2. It uses the interquartile range (IQR) to find outliers, which is the range of the middle 50% of the data.
3. It is good for finding outliers in data that is not evenly spread out and is not affected by very high or low values.
4. The results are the same as Tukey's Fences since both methods are similar.

2.3 Detect the outlier using Turkey's fences.

Turkey's Fences:
Tukey's Fences is a method similar to the IQR method but is based on the concept of whiskers in a boxplot. The fences are set at 1.5 times the IQR for mild outliers and 3 times the IQR for extreme outliers. Any data point outside these fences is considered an outlier. This method provides a clear visual representation of outliers in the data.

Code:

tukey_fences <- boxplot.stats(ozone)$out

cat("Tukey's fences outliers:\n")

print(tukey_fences)

Output:

```
Tukey's fences outliers:
> print(tukey_fences)
[1] 135 168
> 
```

Outliers:

135, 168

1. It also found 135 and 168 as outliers, just like the IQR method.
2. This shows that both methods are reliable for finding significant outliers.

2.4 Detect outliers using Grubbs' test.

Grubb's Test:

Grubbs' Test is a statistical test used to detect outliers by testing the hypothesis that the maximum (or minimum) value in a dataset is an outlier. For instance, if Grubbs' Test identifies 168 as an outlier, it means that this value is significantly different from the other values in the dataset, making it an outlier. This method is particularly useful for small sample sizes and for detecting a single outlier in a dataset.

Code:

```
grubbs_test <- grubbs.test(ozone, opposite=FALSE, two.sided=TRUE)

grubbs_outliers <- if (grubbs_test$p.value < 0.05) grubbs_test$alternative else NULL

cat("Grubbs' test outliers:\n")

print(grubbs_outliers)
```

Output:

```
Grubbs' test outliers:
> print(grubbs_outliers)
[1] "highest value 168 is an outlier"
>
```

Outliers:

168

1. It found 168 as an outlier.
2. This test looks for the most extreme value and is good for finding a single outlier.
3. It works best when the data is normally distributed (bell-shaped).

(comparing the detected outliers across different methods)

1. The Z-score method found the most outliers, even mild ones. It’s good if you need to spot many unusual values.

2. The IQR and Tukey's Fences methods only found the most extreme values (135 and 168). These methods are good for data that is not evenly spread out.

3. Grubbs' test identified the single most extreme outlier (168). It’s useful if you’re only looking for the most unusual value in normally distributed data.

3. Task 1.2: Handling Outliers

"The objective of this task is to practice various outlier handling techniques on a dataset with known outliers."

Dataset: Create a synthetic dataset with a normal distribution and introduce a few outliers.

3.1 Create a synthetic dataset with 100 normally distributed data points with a mean of 50 and a standard deviation 10.

Code:

```
set.seed(42) # For reproducibility

data <- rnorm(100, mean=50, sd=10)

print(data)
```

Output:

```
> print(data)
  [1] 63.70958 44.35302 53.63128 56.32863 54.04268
  [6] 48.93875 65.11522 49.05341 70.18424 49.37286
 [11] 63.04870 72.86645 36.11139 47.21211 48.66679
 [16] 56.35950 47.15747 23.43545 25.59533 63.20113
 [21] 46.93361 32.18692 48.28083 62.14675 68.95193
 [26] 45.69531 47.42731 32.36837 54.60097 43.60005
 [31] 54.55450 57.04837 60.35104 43.91074 55.04955
 [36] 32.82991 42.15541 41.49092 25.85792 50.36123
 [41] 52.05999 46.38943 57.58163 42.73295 36.31719
 [46] 54.32818 41.88607 64.44101 45.68554 56.55648
 [51] 53.21925 42.16161 65.75728 56.42899 50.89761
 [56] 52.76551 56.79289 50.89833 20.06910 52.84883
 [61] 46.32765 51.85231 55.81824 63.99737 42.72708
 [66] 63.02543 53.35848 60.38506 59.20729 57.20878
 [71] 39.56881 49.09814 56.23518 40.46477 44.57171
 [76] 55.80996 57.68179 54.63768 41.14224 39.00219
 [81] 65.12707 52.57921 50.88440 48.79103 38.05671
 [86] 56.11997 47.82860 48.17243 59.33346 58.21773
 [91] 63.92116 45.23826 56.50349 63.91110 38.89211
 [96] 41.39207 38.68261 35.40786 50.79983 56.53204
> 
```

3.2 Introduce 5 outliers to the dataset.

Code:

```
data_with_outliers <- c(data, 100, 105, 110, 120, 130)

print(data_with_outliers)
```

Output:

```
> print(data_with_outliers)
  [1]  63.70958  44.35302
  [3]  53.63128  56.32863
  [5]  54.04268  48.93875
  [7]  65.11522  49.05341
  [9]  70.18424  49.37286
 [11]  63.04870  72.86645
 [13]  36.11139  47.21211
 [15]  48.66679  56.35950
 [17]  47.15747  23.43545
 [19]  25.59533  63.20113
 [21]  46.93361  32.18692
 [23]  48.28083  62.14675
 [25]  68.95193  45.69531
 [27]  47.42731  32.36837
 [29]  54.60097  43.60005
 [31]  54.55450  57.04837
 [33]  60.35104  43.91074
 [35]  55.04955  32.82991
 [37]  42.15541  41.49092
 [39]  25.85792  50.36123
 [41]  52.05999  46.38943
 [43]  57.58163  42.73295
 [45]  36.31719  54.32818
 [47]  41.88607  64.44101
 [49]  45.68554  56.55648
 [51]  53.21925  42.16161
 [53]  65.75728  56.42899
 [55]  50.89761  52.76551
 [57]  56.79289  50.89833
 [59]  20.06910  52.84883
 [61]  46.32765  51.85231
 [63]  55.81824  63.99737
 [65]  42.72708  63.02543
 [67]  53.25848  60.28506
 [71]  39.56881  49.09814
 [73]  56.23518  40.46477
 [75]  44.57171  55.80996
 [77]  57.68179  54.63768
 [79]  41.14224  39.00219
 [81]  65.12707  52.57921
 [83]  50.88440  48.79103
 [85]  38.05671  56.11997
 [87]  47.82860  48.17243
 [89]  59.33346  58.21773
 [91]  63.92116  45.23826
 [93]  56.50349  63.91110
 [95]  38.89211  41.39207
 [97]  38.68261  35.40786
 [99]  50.79983  56.53204
[101] 100.00000 105.00000
[103] 110.00000 120.00000
[105] 130.00000
```

3.3 Apply Winsorisation with the 5th and 95th percentiles to handle the outliers.

Winsorisation:

Replaces extreme outliers with values near a specific percentile, making the outliers less extreme.

Pros:

Reduces the impact of outliers without removing any data points.

Cons:

Can distort the data if there are many extreme values. The original data distribution is somewhat kept but not entirely.

Code:

```
library(DescTools)

winsorized_data <- Winsorize(data_with_outliers, probs=c(0.05, 0.95))

print(winsorized_data)
```

Output:

```
> print(winsorized_data)
  [1] 63.70958 44.35302 53.63128 56.32863 54.04268 48.93875
  [7] 65.11522 49.05341 70.18424 49.37286 63.04870 72.33001
 [13] 36.11139 47.21211 48.66679 56.35950 47.15747 32.46068
 [19] 32.46068 63.20113 46.93361 32.46068 48.28083 62.14675
 [25] 68.95193 45.69531 47.42731 32.46068 54.60097 43.60005
 [31] 54.55450 57.04837 60.35104 43.91074 55.04955 32.82991
 [37] 42.15541 41.49092 32.46068 50.36123 52.05999 46.38943
 [43] 57.58163 42.73295 36.31719 54.32818 41.88607 64.44101
 [49] 45.68554 56.55648 53.21925 42.16161 65.75728 56.42899
 [55] 50.89761 52.76551 56.79289 50.89833 32.46068 52.84883
 [61] 46.32765 51.85231 55.81824 63.99737 42.72708 63.02543
 [67] 53.35848 60.38506 59.20729 57.20878 39.56881 49.09814
 [73] 56.23518 40.46477 44.57171 55.80996 57.68179 54.63768
 [79] 41.14224 39.00219 65.12707 52.57921 50.88440 48.79103
 [85] 38.05671 56.11997 47.82860 48.17243 59.33346 58.21773
 [91] 63.92116 45.23826 56.50349 63.91110 38.89211 41.39207
 [97] 38.68261 35.40786 50.79983 56.53204 72.33001 72.33001
[103] 72.33001 72.33001 72.33001
```

3.4 Apply trimming by removing the top and bottom 5% of the data.

Trimming:

Excludes the most extreme values, resulting in a smaller but cleaner dataset.

Pros:

Completely removes extreme outliers, so they don't affect the analysis.

Cons:

Loses data points, which can reduce the statistical power of your analysis. Not good for small datasets or if the extreme values are important.

Code:

```
trimmed_data <- data_with_outliers

lower_bound <- quantile(data_with_outliers, 0.05)

upper_bound <- quantile(data_with_outliers, 0.95)

trimmed_data <- trimmed_data[trimmed_data >= lower_bound & trimmed_data <=
upper_bound]

print(trimmed_data)
```

Output:

```
> print(trimmed_data)
 [1] 63.70958 44.35302 53.63128 56.32863 54.04268 48.93875
 [7] 65.11522 49.05341 70.18424 49.37286 63.04870 36.11139
[13] 47.21211 48.66679 56.35950 47.15747 63.20113 46.93361
[19] 48.28083 62.14675 68.95193 45.69531 47.42731 54.60097
[25] 43.60005 54.55450 57.04837 60.35104 43.91074 55.04955
[31] 32.82991 42.15541 41.49092 50.36123 52.05999 46.38943
[37] 57.58163 42.73295 36.31719 54.32818 41.88607 64.44101
[43] 45.68554 56.55648 53.21925 42.16161 65.75728 56.42899
[49] 50.89761 52.76551 56.79289 50.89833 52.84883 46.32765
[55] 51.85231 55.81824 63.99737 42.72708 63.02543 53.35848
[61] 60.38506 59.20729 57.20878 39.56881 49.09814 56.23518
[67] 40.46477 44.57171 55.80996 57.68179 54.63768 41.14224
[73] 39.00219 65.12707 52.57921 50.88440 48.79103 38.05671
[79] 56.11997 47.82860 48.17243 59.33346 58.21773 63.92116
[85] 45.23826 56.50349 63.91110 38.89211 41.39207 38.68261
[91] 35.40786 50.79983 56.53204
```

3.5 Apply mean imputation to the outliers.

Mean imputation:

Imputed data can introduce bias if the outliers are important to the analysis.

Pros:

Easy to do and keeps the size of the dataset the same.

Cons:

Can give misleading results by making the data look more uniform. Replaces outliers with the average value, which might not be accurate.

Code:

```
mean_value <- mean(data_with_outliers)

iqr_bounds <- quantile(data_with_outliers, c(0.25, 0.75))

iqr_value <- IQR(data_with_outliers)

lower_bound <- iqr_bounds[1] - 1.5 * iqr_value

upper_bound <- iqr_bounds[2] + 1.5 * iqr_value

mean_imputed_data <- data_with_outliers

mean_imputed_data[mean_imputed_data < lower_bound | mean_imputed_data > upper_bound] <- mean_value

print(mean_imputed_data)
```

Output:

```
> print(mean_imputed_data)
  [1] 63.70958 44.35302 53.63128 56.32863 54.04268 48.93875
  [7] 65.11522 49.05341 70.18424 49.37286 63.04870 72.86645
 [13] 36.11139 47.21211 48.66679 56.35950 47.15747 53.30966
 [19] 25.59533 63.20113 46.93361 32.18692 48.28083 62.14675
 [25] 68.95193 45.69531 47.42731 32.36837 54.60097 43.60005
 [31] 54.55450 57.04837 60.35104 43.91074 55.04955 32.82991
 [37] 42.15541 41.49092 25.85792 50.36123 52.05999 46.38943
 [43] 57.58163 42.73295 36.31719 54.32818 41.88607 64.44101
 [49] 45.68554 56.55648 53.21925 42.16161 65.75728 56.42899
 [55] 50.89761 52.76551 56.79289 50.89833 53.30966 52.84883
 [61] 46.32765 51.85231 55.81824 63.99737 42.72708 63.02543
 [67] 53.35848 60.38506 59.20729 57.20878 39.56881 49.09814
 [73] 56.23518 40.46477 44.57171 55.80996 57.68179 54.63768
 [79] 41.14224 39.00219 65.12707 52.57921 50.88440 48.79103
 [85] 38.05671 56.11997 47.82860 48.17243 59.33346 58.21773
 [91] 63.92116 45.23826 56.50349 63.91110 38.89211 41.39207
 [97] 38.68261 35.40786 50.79983 56.53204 53.30966 53.30966
[103] 53.30966 53.30966 53.30966
```

3.6 Apply a logarithmic transformation to the dataset.

Logarithmic Transformation:

Compresses the data range, reducing the impact of large values, but changes the original scale of the data.

Pros:

Can make the data more normally distributed and stabilize variance. Helpful for data that is skewed to the right.

Cons:

Only works for positive data. Cannot be applied to zero or negative values.

Code:

```
log_transformed_data <- log(data_with_outliers)

print(log_transformed_data)
```

Output:

```
> print(log_transformed_data)
  [1] 4.154335 3.792181 3.982133 4.031203 3.989774 3.890570
  [7] 4.176158 3.892910 4.251124 3.899401 4.143907 4.288628
 [13] 3.586608 3.854650 3.884997 4.031751 3.853492 3.154250
 [19] 3.242410 4.146322 3.848734 3.471560 3.877035 4.129498
 [25] 4.233410 3.821996 3.859198 3.477182 4.000052 3.775058
 [31] 3.999200 4.043900 4.100178 3.782159 4.008234 3.491340
 [37] 3.741363 3.725475 3.252617 3.919222 3.952397 3.837072
 [43] 4.053204 3.754970 3.592291 3.995043 3.734953 4.165750
 [49] 3.821782 4.035240 3.974420 3.741510 4.185970 4.032983
 [55] 3.929816 3.965858 4.039411 3.929830 2.999181 3.967436
 [61] 3.835739 3.948399 4.022101 4.158842 3.754833 4.143538
 [67] 3.977033 4.100742 4.081045 4.046707 3.678041 3.893821
 [73] 4.029543 3.700432 3.797099 4.021952 4.054941 4.000724
 [79] 3.717035 3.663618 4.176340 3.962321 3.929556 3.887547
 [85] 3.639077 4.027492 3.867624 3.874787 4.083173 4.064190
 [91] 4.157651 3.811943 4.034302 4.157493 3.660791 3.723089
 [97] 3.655390 3.566934 3.927893 4.034808 4.605170 4.653960
[103] 4.700480 4.787492 4.867534
```

4. Task 2: Analysing a different dataset

"Choose a new dataset to perform EDA. You can use any dataset or select one from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php)."

Using "Wine Quality dataset from the UCI Machine Learning Repository

4.1 Load your dataset into R

Code:

(importing necessary libraries)

```
library(tidyverse)

library(ggplot2)

library(DT)

library(caret)
```

(Loading the dataset)

```
wine <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv", sep = ";")
```

(Displaying the head)

```
datatable(head(wine))
```

Output:

Show 10 entries

Search:

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlori
1	7.4	0.7	0	1.9	
2	7.8	0.88	0	2.6	
3	7.8	0.76	0.04	2.3	
4	11.2	0.28	0.56	1.9	
5	7.4	0.7	0	1.9	
6	7.4	0.66	0	1.8	

Showing 1 to 6 of 6 entries

Previous 1 Next

4.2 Perform data cleansing and preprocessing.

Code:

(Checking for missing values)

sum(is.na(wine))

summary(wine)

Output:

```
> sum(is.na(wine))
[1] 0
> # Summary statistics
> summary(wine)
 fixed.acidity   volatile.acidity  citric.acid   
 Min.   : 4.60   Min.   :0.1200   Min.   :0.000  
 1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090  
 Median : 7.90   Median :0.5200   Median :0.260  
 Mean   : 8.32   Mean   :0.5278   Mean   :0.271  
 3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420  
 Max.   :15.90   Max.   :1.5800   Max.   :1.000  
 residual.sugar     chlorides       free.sulfur.dioxide
 Min.   : 0.900   Min.   :0.01200   Min.   : 1.00      
 1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00      
 Median : 2.200   Median :0.07900   Median :14.00      
 Mean   : 2.539   Mean   :0.08747   Mean   :15.87      
 3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00      
 Max.   :15.500   Max.   :0.61100   Max.   :72.00      
 total.sulfur.dioxide    density             pH       
 Min.   :  6.00       Min.   :0.9901   Min.   :2.740  
 1st Qu.: 22.00       1st Qu.:0.9956   1st Qu.:3.210  
 Median : 38.00       Median :0.9968   Median :3.310  
 Mean   : 46.47       Mean   :0.9967   Mean   :3.311  
 3rd Qu.: 62.00       3rd Qu.:0.9978   3rd Qu.:3.400  
 Max.   :289.00       Max.   :1.0037   Max.   :4.010  
   sulphates         alcohol          quality     
 Min.   :0.3300   Min.   : 8.40   Min.   :3.000  
 1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000  
 Median :0.6200   Median :10.20   Median :6.000  
 Mean   :0.6581   Mean   :10.42   Mean   :5.636  
 3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000  
 Max.   :2.0000   Max.   :14.90   Max.   :8.000  
> 
```

4.3 Conduct univariate, bivariate and multivariate analysis.

Univariate Analysis

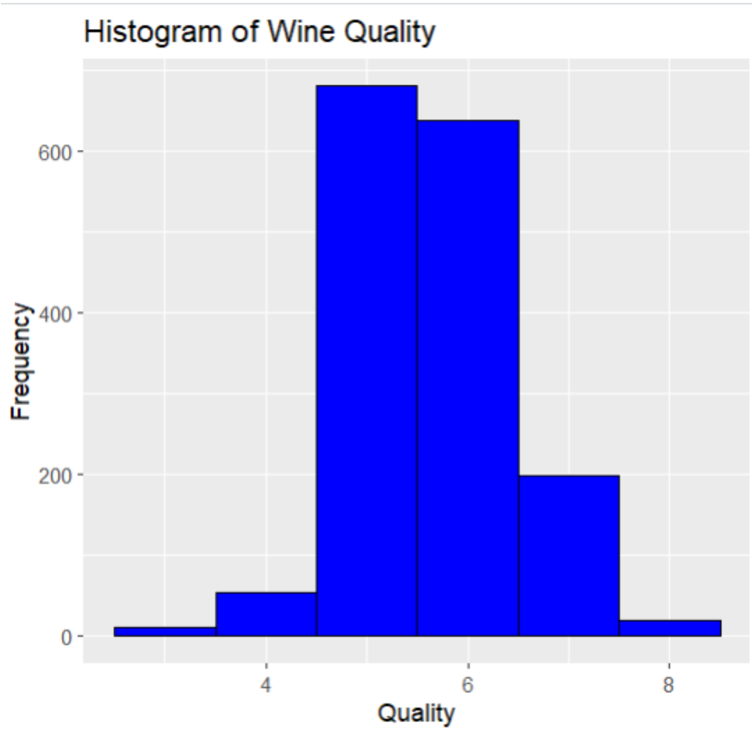
Definition: This type of analysis focuses on one single variable. It helps us understand the basic characteristics of that variable, like its distribution, central tendency, and spread.

(Histogram for wine quantity)

Code:

```
ggplot(wine, aes(x = quality)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Histogram of Wine Quality", x = "Quality", y = "Frequency")
```

Output:

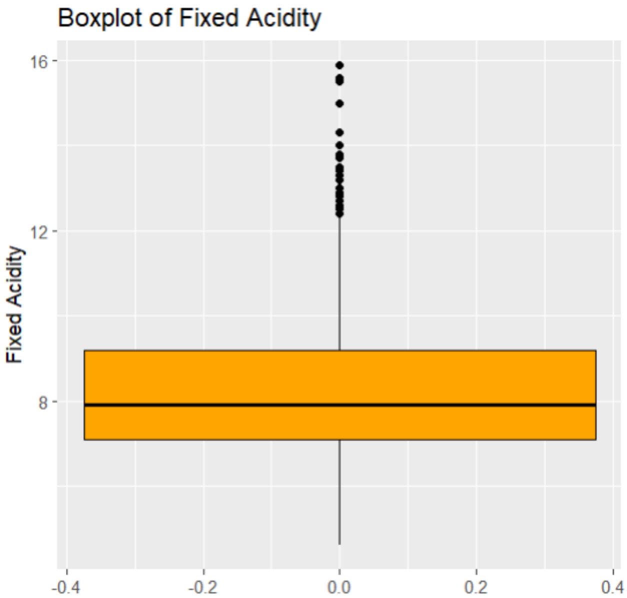


(Boxplot for fixed acidity)

Code:

```
ggplot(wine, aes(y = `fixed.acidity`)) +
  geom_boxplot(fill = "orange", color = "black") +
  labs(title = "Boxplot of Fixed Acidity", y = "Fixed Acidity")
```

Output:



Bivariate Analysis

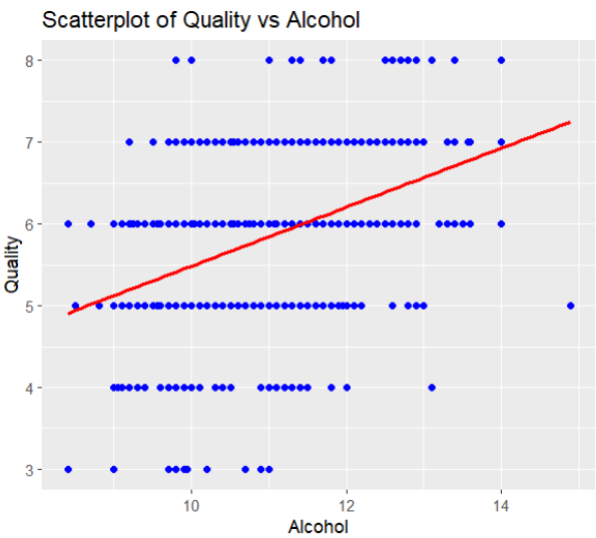
Definition: This type of analysis examines the relationship between two variables. It helps us see if and how one variable affects or is related to another variable.

(Scatterplot for quantity vs alcohol)

Code:

```
ggplot(wine, aes(x = alcohol, y = quality)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Scatterplot of Quality vs Alcohol", x = "Alcohol", y = "Quality")
```

Output:

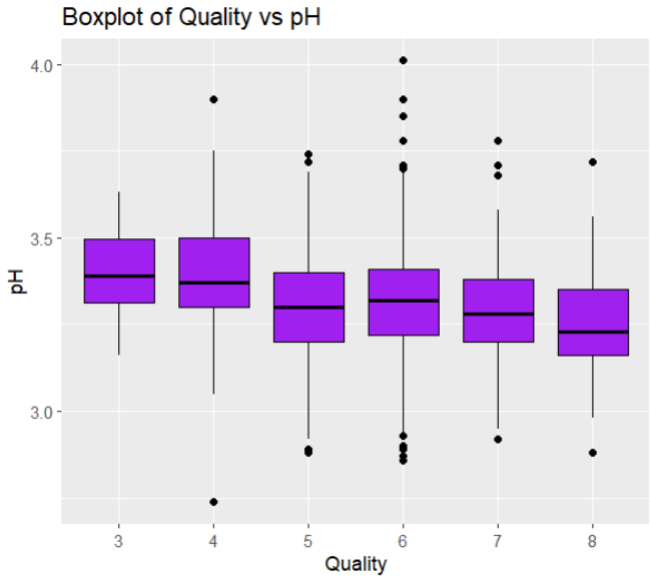


(Boxplot for quantity vs PH)

Code:

```
ggplot(wine, aes(x = factor(quality), y = pH)) +
  geom_boxplot(fill = "purple", color = "black") +
  labs(title = "Boxplot of Quality vs pH", x = "Quality", y = "pH")
```

Output:



Multivariate Analysis

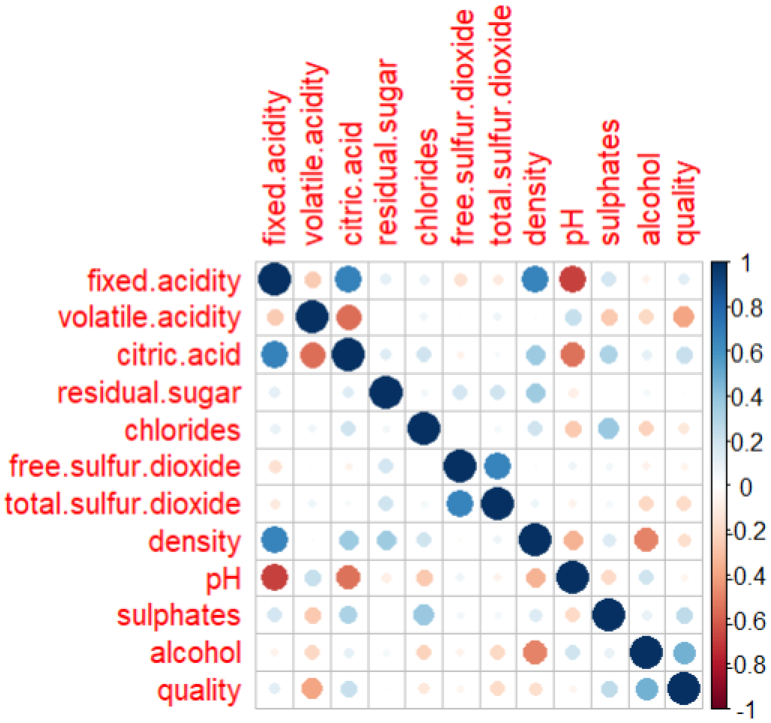
Definition: This type of analysis looks at more than two variables at the same time. It helps us understand complex relationships and interactions between multiple variables.

(Correlation matrix)

Code:

```
cor_matrix <- cor(wine)
library(corrplot)
corrplot(cor_matrix, method = "circle")
```

Output:

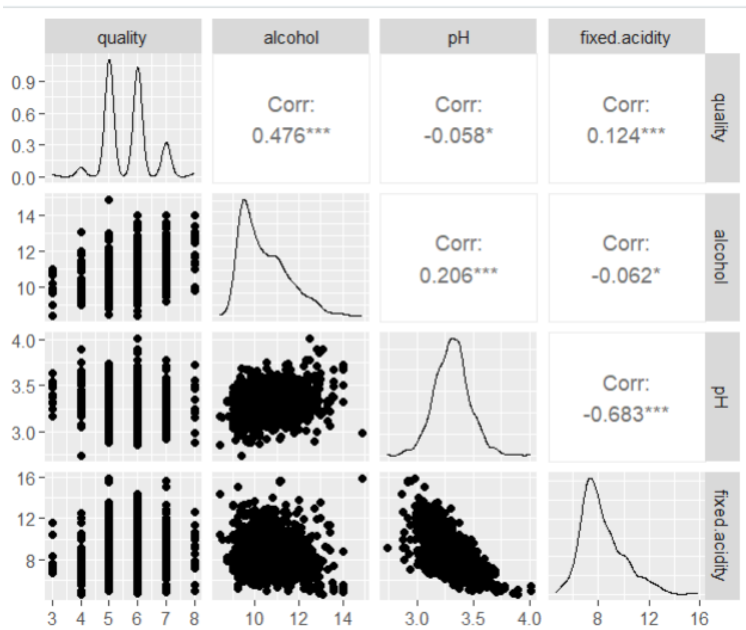


(Pair plot)

Code:

```
library(GGally)
ggpairs(wine[, c("quality", "alcohol", "pH", "fixed.acidity")])
```

Output:



4.4 Detect and handle outliers.

(Detecting/Identifying outliersusing IQR method for alcohol)

Code:

```
Q1 <- quantile(wine$alcohol, 0.25)

Q3 <- quantile(wine$alcohol, 0.75)

IQR <- Q3 - Q1

lower_bound <- Q1 - 1.5 * IQR

upper_bound <- Q3 + 1.5 * IQR

alcohol_outliers <- wine[wine$alcohol < lower_bound | wine$alcohol > upper_bound,
]

print(alcohol_outliers)
```

Output

```
> print(alcohol_outliers)
 [1] fixed.acidity        volatile.acidity
 [3] citric.acid          residual.sugar
 [5] chlorides            free.sulfur.dioxide
 [7] total.sulfur.dioxide density
 [9] pH                   sulphates
[11] alcohol              quality
<0 rows> (or 0-length row.names)
```

(Handling Outliers using winsorisation)

Code:

```
wine$alcohol <- Winsorize(wine$alcohol, probs = c(0.05, 0.95))

print(wine$alcohol)
```

Output:

```
> print(wine$alcohol)
  [1]  9.4  9.8  9.8  9.8  9.4  9.4  9.4 10.0  9.5
 [10] 10.5  9.2 10.5  9.9  9.2  9.2  9.2 10.5  9.3
 [19]  9.2  9.2  9.4  9.7  9.5  9.4  9.7  9.3  9.5
 [28]  9.5  9.4  9.8 10.1 10.6  9.8  9.4  9.2  9.6
 [37] 10.8  9.7  9.8 10.5 10.5  9.3 10.5 10.3  9.5
 [46] 12.5  9.2  9.5  9.2  9.2  9.2  9.4  9.4  9.4
 [55] 10.2  9.5  9.6  9.4 10.0  9.4  9.2  9.3  9.5
 [64]  9.8 10.9 10.9  9.6 10.7 10.7 10.5  9.5  9.5
 [73]  9.5  9.2  9.6 10.5 10.5 10.7 10.1  9.2  9.2
 [82]  9.4  9.2  9.4 10.3 10.1  9.9  9.6  9.5  9.2
 [91]  9.5  9.9  9.8  9.6 10.5 12.5 10.7  9.2  9.8
[100]  9.2 10.2 10.4  9.2  9.2  9.4  9.2  9.3  9.3
[109]  9.6  9.3  9.5  9.8  9.8  9.7  9.5 10.5 10.0
[118]  9.4 10.9  9.2  9.2 10.9  9.2  9.5  9.5  9.4
[127] 10.9 10.9 10.5  9.4  9.4 12.5 12.5  9.8  9.9
```

```
[136]  9.6  9.5  9.2  9.5  9.5  9.6  9.5 12.5  9.4
[145] 12.5  9.4 10.0  9.3 10.2 10.5 10.3  9.4 10.1
[154] 10.1 10.5 10.5 10.5 10.5  9.3  9.3  9.6  9.2
[163] 10.0  9.4  9.4  9.5 10.2  9.2 10.4  9.5  9.2
[172]  9.2  9.2 11.5  9.5  9.5  9.5 10.5  9.6  9.5
[181]  9.5  9.3  9.3  9.3  9.3  9.7  9.2  9.7  9.5
[190]  9.5  9.4  9.8  9.5  9.7  9.7  9.4 10.2 10.1
[199] 12.5 11.4 10.3  9.3  9.5  9.2  9.2 10.8 10.8
[208]  9.3  9.4 10.5 12.4 10.0 10.2 10.1  9.8 10.5
[217] 11.0  9.2  9.7  9.5  9.4  9.4  9.5 10.0 10.4
[226] 10.5  9.5  9.8 10.5 11.0 12.2  9.9  9.6 11.0
[235]  9.2  9.2  9.2  9.2  9.2  9.2  9.3 10.9  9.8
[244]  9.2  9.2  9.9  9.5  9.3  9.8  9.9 10.0  9.9
[253] 10.5  9.5  9.9  9.3  9.2  9.2  9.4 10.5  9.3
[262]  9.4 10.0  9.3 10.9 10.2  9.8 12.5  9.4 10.1
[271] 10.7 10.1 10.1  9.4  9.4 10.7  9.4 10.1 12.5
[280] 10.5  9.3  9.9  9.2 10.5  9.8  9.8 10.3 10.3
[289] 10.6  9.2 10.6 10.5 10.3 10.1  9.5  9.5  9.9
[298]  9.6  9.7  9.6 10.7 10.1 10.0  9.5  9.2  9.3
[307]  9.4  9.5  9.5  9.5  9.3  9.4  9.5  9.4 11.0
[316] 11.0 10.1 10.4 11.5 10.4 11.5  9.7  9.3  9.5
[325]  9.2  9.2 11.5 11.5  9.7  9.5 12.5 12.5  9.4
[334] 11.0 11.7 12.2 12.5 10.3 11.5  9.8  9.2 11.3
[343]  9.8  9.8 10.7  9.9 12.3 12.0 10.0  9.4  9.9
[352]  9.4  9.3 12.5 11.9 12.5 11.0 11.7 10.4  9.8
[361]  9.4  9.9 10.0 10.2 10.0 11.8 10.0  9.2  9.4
[370] 12.0  9.9  9.2 10.6  9.2 10.8 11.8 11.0 12.0
[379] 12.5 10.8  9.4 10.0  9.4  9.4  9.2  9.7  9.2
[388]  9.6  9.2 10.0 12.5 10.0  9.5  9.2  9.9 12.5
[397]  9.9 11.0 11.0  9.4  9.9 10.8 10.5 10.5  9.2
[406] 10.1 10.8 10.8 11.3  9.6  9.5  9.5  9.3 11.7
[415]  9.5  9.3 11.7 10.5 10.4  9.9 11.8 12.3 10.9
[424] 11.0 10.9 12.3 11.4 10.6  9.3 10.4 11.0  9.2
[433] 12.5  9.5  9.9  9.5 10.2 11.2  9.9  9.3  9.8
[442] 11.3 11.2 11.6 12.5 10.1 10.5 11.2 10.2 10.8
[451] 10.8  9.2 10.0 11.2 11.1 12.5 10.3  9.6 11.2
[460]  9.2 11.3  9.3 11.8  9.2  9.2  9.7 11.5 12.5
[469]  9.2  9.8 10.6 11.4 10.4 10.6  9.4 10.2  9.7
[478] 11.0 10.2 10.1  9.2 11.7  9.4  9.4 12.5 10.0
[487] 10.0 10.0 10.8 10.2 10.6 12.5 12.5 11.6 12.1

[496] 11.0  9.2 11.1 11.0 11.6  9.2 12.0 12.0 10.9
[505] 10.8 12.5 10.8  9.5 10.2 11.4  9.5 10.2  9.7
[514] 11.8 11.8  9.3 11.9  9.2 11.7 11.0 10.0  9.2
[523]  9.8  9.4  9.5  9.9 11.0 11.4  9.2  9.4 10.3
[532] 10.3 10.3 12.5 10.0 10.3  9.4 10.7 12.0 11.2
[541]  9.6 11.0  9.9 11.0  9.2  9.2  9.5 10.7 10.4
[550]  9.4  9.5 10.0 10.0 11.5 11.1 11.1 11.7 11.1
[559] 11.7 12.5 11.4  9.2  9.2 10.1 12.5 11.4  9.2
[568]  9.2 10.7 11.7 11.0 11.7 10.4  9.6 10.0 10.2
[577] 10.0  9.5  9.8  9.8  9.6  9.6  9.2  9.9 10.7
[586]  9.6 10.6  9.3 12.5 10.5  9.7 11.5  9.7  9.2
[595]  9.5  9.3  9.3 10.0  9.8  9.3 10.0  9.2  9.3
[604]  9.2  9.2  9.2 12.2 10.5 10.4 12.5  9.2  9.4
[613] 10.0  9.8 10.2  9.7  9.7  9.8 10.2  9.3  9.4
[622]  9.4  9.5 12.1 10.2 10.2  9.2  9.2  9.3  9.3
[631]  9.3  9.5 10.5 11.3  9.5  9.7  9.4  9.4 10.2
[640] 10.3  9.4  9.5  9.4  9.5  9.4 10.1 10.1 11.0
[649] 11.2 11.3  9.6 11.2 12.5 12.0  9.5  9.4  9.6
[658] 10.5  9.6 11.0  9.6  9.2  9.6 10.2 10.2  9.7
[667]  9.5  9.2 11.0  9.2 10.0  9.5  9.5  9.5  9.3
[676] 10.2  9.3  9.9 10.0  9.6  9.2 10.2  9.8 11.3
[685]  9.4 11.3  9.2  9.7  9.4  9.4 10.7  9.8  9.2
[694]  9.4  9.4 12.5  9.5  9.5  9.7 10.8 10.1  9.5
[703]  9.4  9.6  9.7  9.9 10.0 10.5 11.6 10.0 10.1
[712]  9.5  9.4  9.4  9.8  9.2  9.4 10.0  9.6  9.5
[721]  9.6  9.2 10.0  9.5 11.2 10.4 11.1  9.5  9.5
[730] 12.5  9.6 11.5  9.6  9.5  9.3  9.5  9.5  9.3
[739]  9.2  9.3 11.5  9.5  9.2 10.0  9.5  9.5  9.2
[748]  9.4  9.6  9.5  9.5  9.5  9.4  9.5  9.2 10.7
[757] 11.2  9.8  9.8  9.2  9.7  9.6 10.0  9.6  9.5
[766]  9.5  9.4  9.5  9.7  9.6  9.7  9.4  9.4  9.5
[775]  9.5 10.0 10.3 10.3 10.5  9.8  9.4  9.8 10.0
[784]  9.8  9.8  9.5  9.5 10.1 10.1  9.3  9.7  9.6
[793]  9.7 10.8 12.5 10.2  9.6 10.8 10.7 10.7  9.4
[802] 10.0 12.5  9.6  9.9 12.5 12.5 12.5  9.2 10.3
[811] 10.5 10.9 10.8 11.4 11.3 10.8 10.5 11.9  9.4
[820]  9.6  9.7 12.5  9.8  9.8 10.3 10.7 11.0 10.7
[829] 12.5 11.1 10.9 11.1  9.9  9.9  9.4  9.3 11.7
[838] 11.7 11.2 10.0 12.1 10.3 10.9  9.4 10.6  9.8
[847]  9.8  9.9  9.8  9.8  9.5  9.5  9.7 10.9 10.9
```

4.5 Perform feature transformation if needed.

(Performing Logarithmic Transformation)

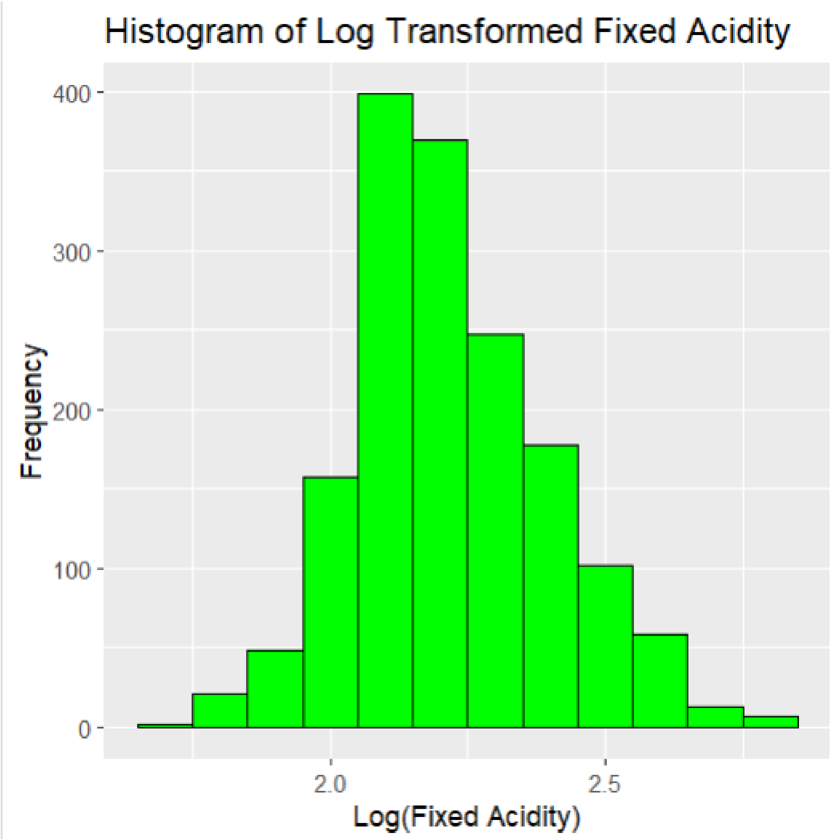
Code:

```
wine$log_fixed_acidity <- log(wine$`fixed.acidity` + 1) # Adding 1 to avoid log(0)
```

(Checking distribution after performing the Logarithmic Transformation)

```
ggplot(wine, aes(x = log_fixed_acidity)) +
  geom_histogram(binwidth = 0.1, fill = "green", color = "black") +
  labs(title = "Histogram of Log Transformed Fixed Acidity", x = "Log(Fixed Acidity)", y
= "Frequency")
```

Output:



4.6 Execute the other relevant EDA tasks.

(Summary statistic after cleansing and transformation)

Code:

summary(wine)

Output:

```
 fixed.acidity    volatile.acidity
 Min.   : 4.60   Min.   :0.1200
 1st Qu.: 7.10   1st Qu.:0.3900
 Median : 7.90   Median :0.5200
 Mean   : 8.32   Mean   :0.5278
 3rd Qu.: 9.20   3rd Qu.:0.6400
 Max.   :15.90   Max.   :1.5800
  citric.acid     residual.sugar
 Min.   :0.000   Min.   : 0.900
 1st Qu.:0.090   1st Qu.: 1.900
 Median :0.260   Median : 2.200
 Mean   :0.271   Mean   : 2.539
 3rd Qu.:0.420   3rd Qu.: 2.600
 Max.   :1.000   Max.   :15.500
   chlorides
 Min.   :0.01200
 1st Qu.:0.07000
 Median :0.07900
 Mean   :0.08747
 3rd Qu.:0.09000
 Max.   :0.61100
 free.sulfur.dioxide
 Min.   : 1.00
 1st Qu.: 7.00
 Median :14.00
 Mean   :15.87
 3rd Qu.:21.00
 Max.   :72.00
 total.sulfur.dioxide
 Min.   :  6.00
 1st Qu.: 22.00
 Median : 38.00
 Mean   : 46.47
 3rd Qu.: 62.00
 Max.   :289.00

    density             pH
 Min.   :0.9901   Min.   :2.740
 1st Qu.:0.9956   1st Qu.:3.210
 Median :0.9968   Median :3.310
 Mean   :0.9967   Mean   :3.311
 3rd Qu.:0.9978   3rd Qu.:3.400
 Max.   :1.0037   Max.   :4.010
   sulphates         alcohol
 Min.   :0.3300   Min.   : 9.20
 1st Qu.:0.5500   1st Qu.: 9.50
 Median :0.6200   Median :10.20
 Mean   :0.6581   Mean   :10.41
 3rd Qu.:0.7300   3rd Qu.:11.10
 Max.   :2.0000   Max.   :12.50
    quality       log_fixed_acidity
 Min.   :3.000   Min.   :1.723
 1st Qu.:5.000   1st Qu.:2.092
 Median :6.000   Median :2.186
 Mean   :5.636   Mean   :2.216
 3rd Qu.:6.000   3rd Qu.:2.322
 Max.   :8.000   Max.   :2.827
```

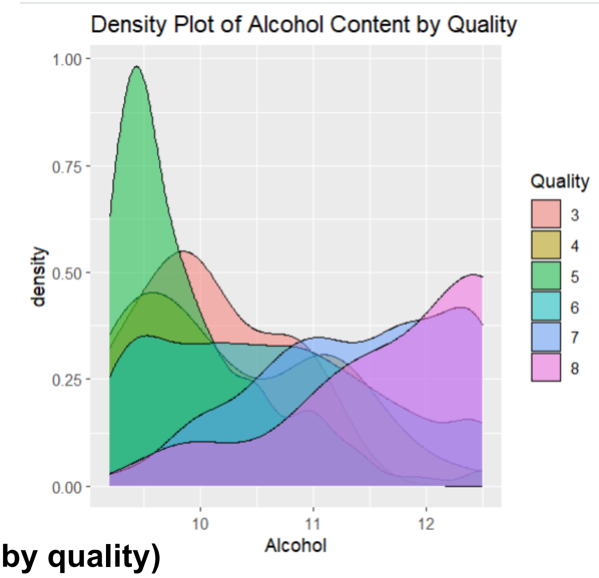
(Some additional Visualizations)

(Density plot for alcohol content by quality)

Code:

```
ggplot(wine, aes(x = alcohol, fill = factor(quality))) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Alcohol Content by Quality", x = "Alcohol", fill = "Quality")
```

Output:

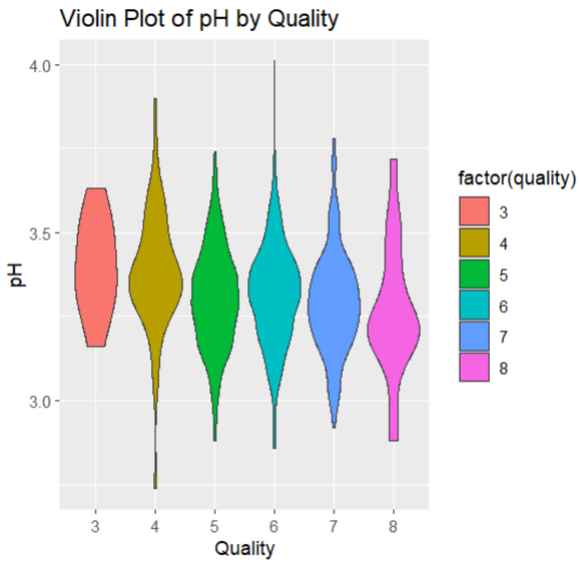


(Violin plot for pH by quality)

Code:

```
ggplot(wine, aes(x = factor(quality), y = pH, fill = factor(quality))) +
  geom_violin() +
  labs(title = "Violin Plot of pH by Quality", x = "Quality", y = "pH")
```

Output:



4.7 Document the findings and observations.

Findings and Observations

1. Data Cleaning:
- No missing values were found in the dataset.
- Data types were appropriate for analysis.

2. Univariate Analysis:
- The wine quality scores range from 3 to 8, with most scores around 5 and 6.
- Fixed acidity shows some outliers.

3. Bivariate Analysis:
- There is a positive correlation between alcohol content and wine quality.
- pH values do not vary significantly across different quality levels.

4. Multivariate Analysis:
- Strong correlations observed between certain variables, such as density and residual sugar.
- Pair plots help visualize relationships between multiple variables simultaneously.

5. Outlier Detection and Handling:
- Outliers were detected in the alcohol content using the IQR method.
- Winsorisation was applied to handle these outliers, reducing their influence on the analysis.

6. Feature Transformation:
- Log transformation was applied to fixed acidity to address skewness.

7. Additional Insights:
- Density plots and violin plots provided deeper insights into the distribution and relationship of variables by quality.

5. Task 3: Categorical variable exploration.

Find a dataset with categorical variables (e.g., the Titanic dataset) and perform EDA, including explore the relationships between these variables using appropriate visualisation techniques, such

as bar plots, stacked bar plots, and mosaic plots.

<u>Loading the titanic dataset</u>

Code:

```
library(tidyverse)

library(ggplot2)

library(ggthemes)

library(titanic)

data("titanic_train")
```

(Displaying the head)

```
datatable(head(titanic_train))
```

Output"

Show 10 entries Search:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 2
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17
[illegible]	[illegible]	[illegible]	Heikkinen, [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	STON

Showing 1 to 6 of 6 entries

Previous 1 Next

Data Cleansing and preprocessing

(Check for missing values)

Code:

sum(is.na(titanic_train))

Output:

```
> sum(is.na(titanic_train))
[1] 177
```

(Summary statistics)

Code:

summary(titanic_train)

Output:

```
> summary(titanic_train)
  PassengerId       Survived          Pclass     
 Min.   :  1.0   Min.   :0.0000   Min.   :1.000  
 1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000  
 Median :446.0   Median :0.0000   Median :3.000  
 Mean   :446.0   Mean   :0.3838   Mean   :2.309  
 3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000  
 Max.   :891.0   Max.   :1.0000   Max.   :3.000  
                                                 
     Name               Sex                 Age       
 Length:891         Length:891         Min.   : 0.42  
 Class :character   Class :character   1st Qu.:20.12  
 Mode  :character   Mode  :character   Median :28.00  
                                       Mean   :29.70  
                                       3rd Qu.:38.00  
                                       Max.   :80.00  
                                       NA's   :177    
     SibSp           Parch           Ticket         
 Min.   :0.000   Min.   :0.0000   Length:891        
 1st Qu.:0.000   1st Qu.:0.0000   Class :character  
 Median :0.000   Median :0.0000   Mode  :character  
 Mean   :0.523   Mean   :0.3816                     
 3rd Qu.:1.000   3rd Qu.:0.0000                     
 Max.   :8.000   Max.   :6.0000                     
                                                    
      Fare           Cabin             Embarked        
 Min.   :  0.00   Length:891         Length:891        
 1st Qu.:  7.91   Class :character   Class :character  
 Median : 14.45   Mode  :character   Mode  :character  
 Mean   : 32.20                                        
 3rd Qu.: 31.00                                        
 Max.   :512.33                                        
```

(Data type conversion if necessary)

Code:

str(titanic_train)

Output:

```
> str(titanic_train)
'data.frame':   891 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. Jo
hn Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "F
utrelle, Mrs. Jacques Heath (Lily May Peel)" ...
 $ Sex        : chr  "male" "female" "female" "female" ...
 $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282"
"113803" ...
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : chr  "" "C85" "" "C123" ...
 $ Embarked   : chr  "S" "C" "S" "S" ...
```

(Handling missing values)

```
titanic_train <- titanic_train %>%
  drop_na()
sum(is.na(titanic_train))
```

Output:

```
> # Handling missing values
> titanic_train <- titanic_train %>%
+   drop_na()
> # Check for missing values
> sum(is.na(titanic_train))
[1] 0
>
```

Exploring relationship between these categorical variables

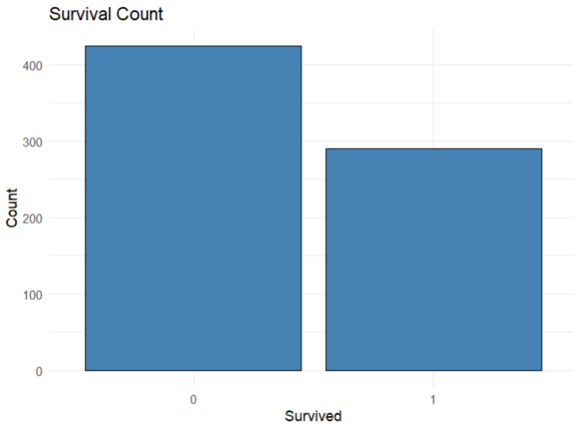
Univariate Analysis

(Bar plot for Survived)

Code:

```
ggplot(titanic_train, aes(x = factor(Survived))) +
  geom_bar(fill = "steelblue", color = "black") +
  labs(title = "Survival Count", x = "Survived", y = "Count") +
  theme_minimal()
```

Output:



(Bar plot for Pclass)

Code:

```
ggplot(titanic_train, aes(x = factor(Pclass))) +
  geom_bar(fill = "coral", color = "black") +
  labs(title = "Passenger Class Count", x = "Pclass", y = "Count") +
  theme_minimal()
```

Output:

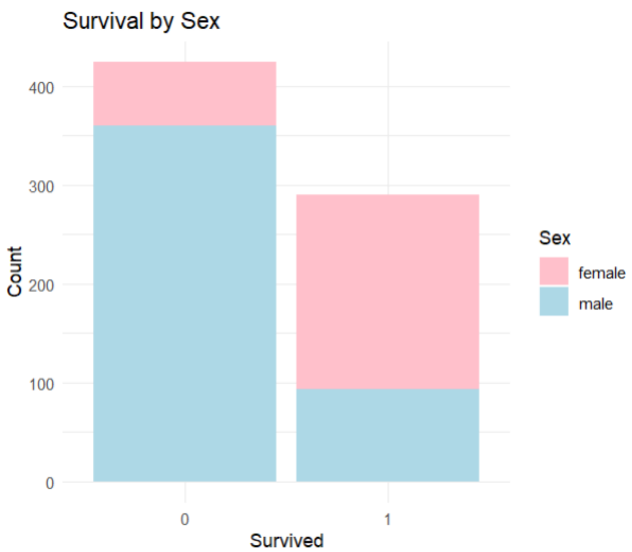
Bivariate Analysis

(Stacked bar plot for Survived by Sex)

Code:

```
ggplot(titanic_train, aes(x = factor(Survived), fill = factor(Sex))) +
  geom_bar(position = "stack") +
  labs(title = "Survival by Sex", x = "Survived", y = "Count", fill = "Sex") +
  scale_fill_manual(values = c("pink", "lightblue")) +
  theme_minimal()
```

Output:

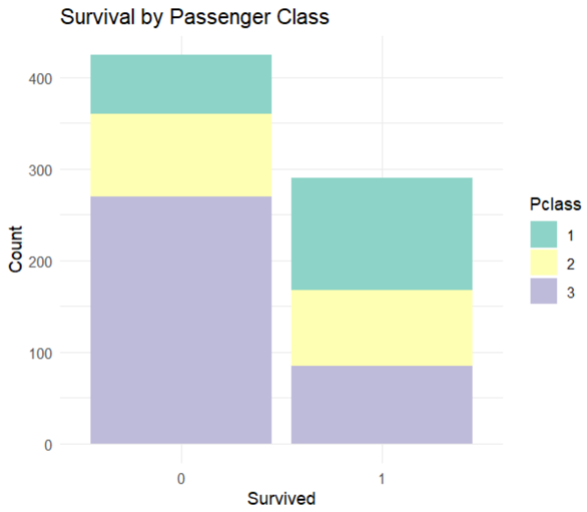


(Stacked bar plot for Survived by Pclass)

Code:

```
ggplot(titanic_train, aes(x = factor(Survived), fill = factor(Pclass))) +
  geom_bar(position = "stack") +
  labs(title = "Survival by Passenger Class", x = "Survived", y = "Count", fill =
"Pclass") +
  scale_fill_brewer(palette = "Set3") +
  theme_minimal()
```

Output:



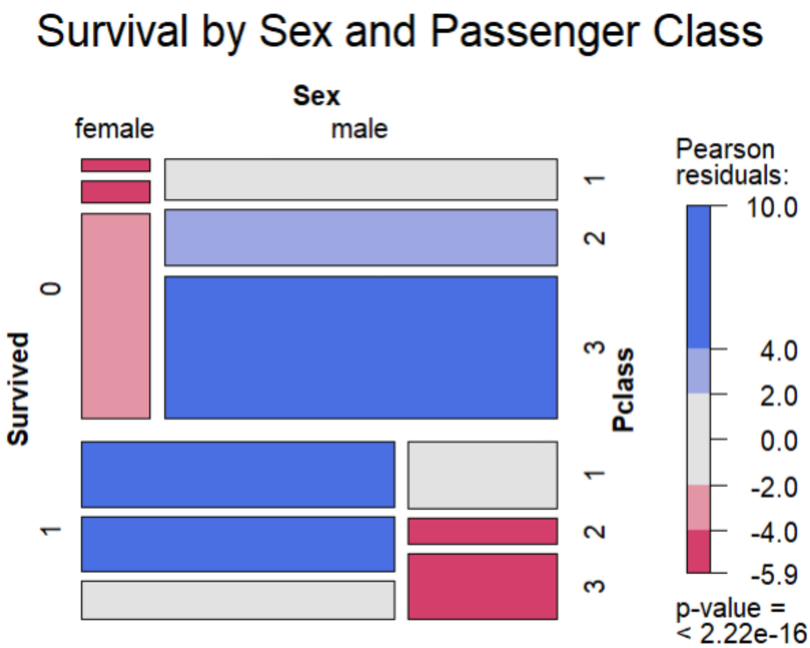
<u>**Multivariate Analysis**</u>

(Mosaic plot for Survived by Sex and Pclass)

Code:

```
library(vcd)
mosaic(~ Survived + Sex + Pclass, data = titanic_train,
       main = "Survival by Sex and Passenger Class",
       shade = TRUE,
       legend = TRUE)
```

Output:



6. Task 3.1: Customizing Visualizations

“Enhance the visualizations created in this tutorial by customizing them according to your preferences. You can use different colors, shapes, and themes for your plots. Explore the ggplot2 documentation (https://ggplot2.tidyverse.org/reference/) to learn more about available customization options.”

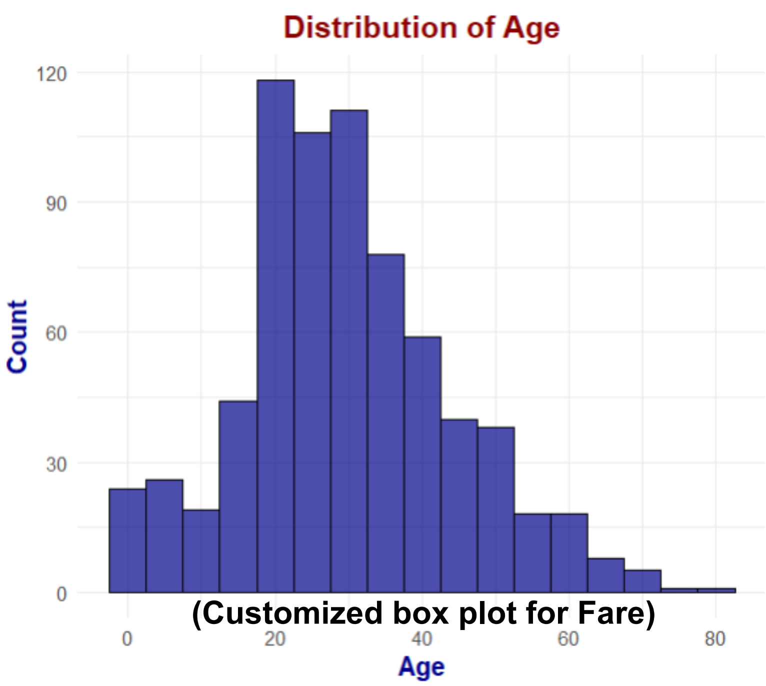
6.1 Modify the histogram and the boxplot int the univariate analysis section with new colours, fill, and themes.

(Customized histogram for Age)

Code:

```
library(ggplot2)
ggplot(titanic_train, aes(x = Age)) +
  geom_histogram(binwidth = 5, fill = "darkblue", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Age", x = "Age", y = "Count") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "darkred", size = 14, face = "bold"),
    axis.title.x = element_text(color = "darkblue", size = 12, face = "bold"),
    axis.title.y = element_text(color = "darkblue", size = 12, face = "bold")
  )
```

Output:

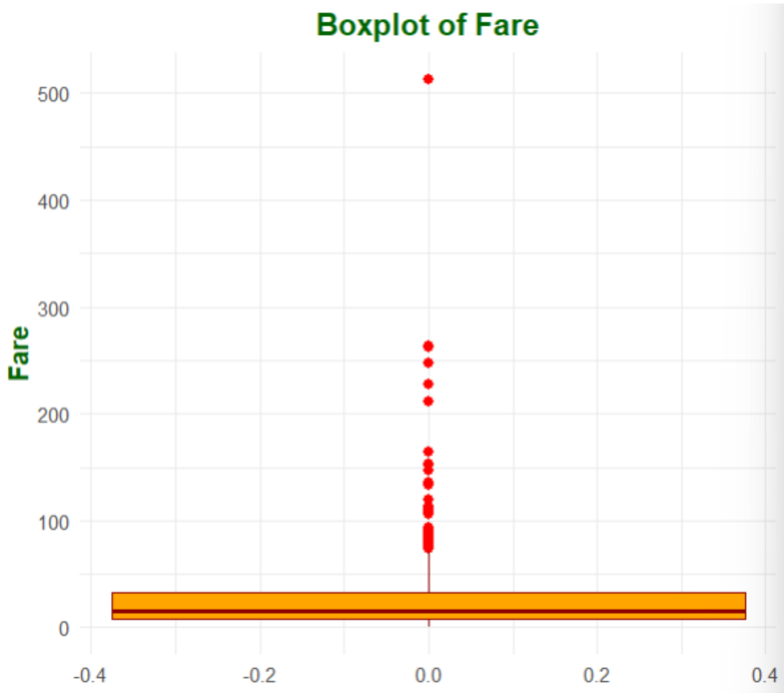


(Customized box plot for Fare)

Code:

```
ggplot(titanic_train, aes(y = Fare)) +
  geom_boxplot(fill = "orange", color = "darkred", outlier.colour = "red", outlier.shape
= 16, outlier.size = 2) +
  labs(title = "Boxplot of Fare", y = "Fare") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "darkgreen", size = 14, face = "bold"),
    axis.title.y = element_text(color = "darkgreen", size = 12, face = "bold")
  )
```

Output:



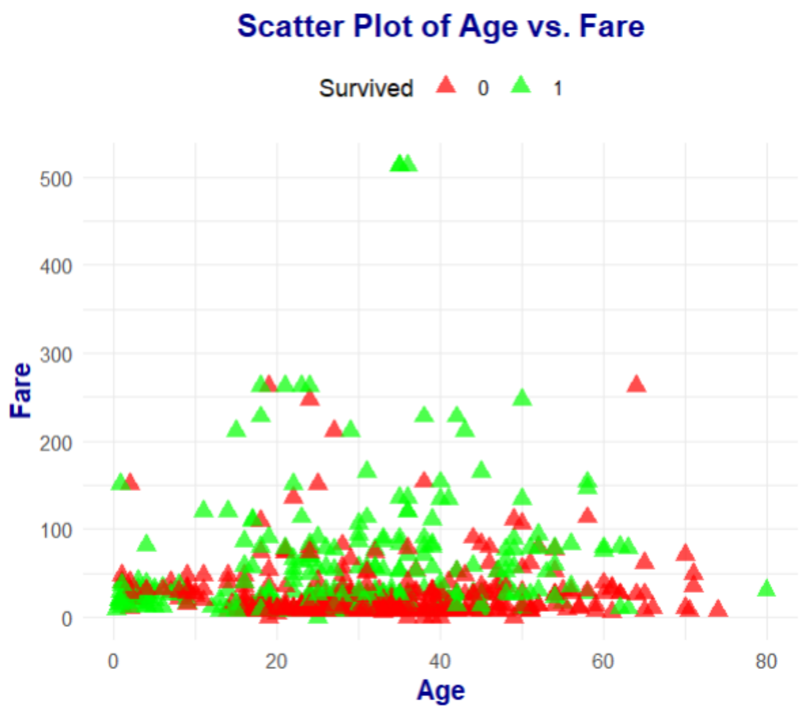
6.2 Customize scatter plot in the bivariate analysis section with different point shapes, sizes, and colours

(Customized scatter plot for Age vs Fare)

Code:

```
ggplot(titanic_train, aes(x = Age, y = Fare, color = factor(Survived))) +
  geom_point(shape = 17, size = 3, alpha = 0.7) +
  scale_color_manual(values = c("red", "green")) +
  labs(title = "Scatter Plot of Age vs. Fare", x = "Age", y = "Fare", color = "Survived") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "darkblue", size = 14, face = "bold"),
    axis.title.x = element_text(color = "darkblue", size = 12, face = "bold"),
    axis.title.y = element_text(color = "darkblue", size = 12, face = "bold"),
    legend.position = "top"
  )
```

Output:



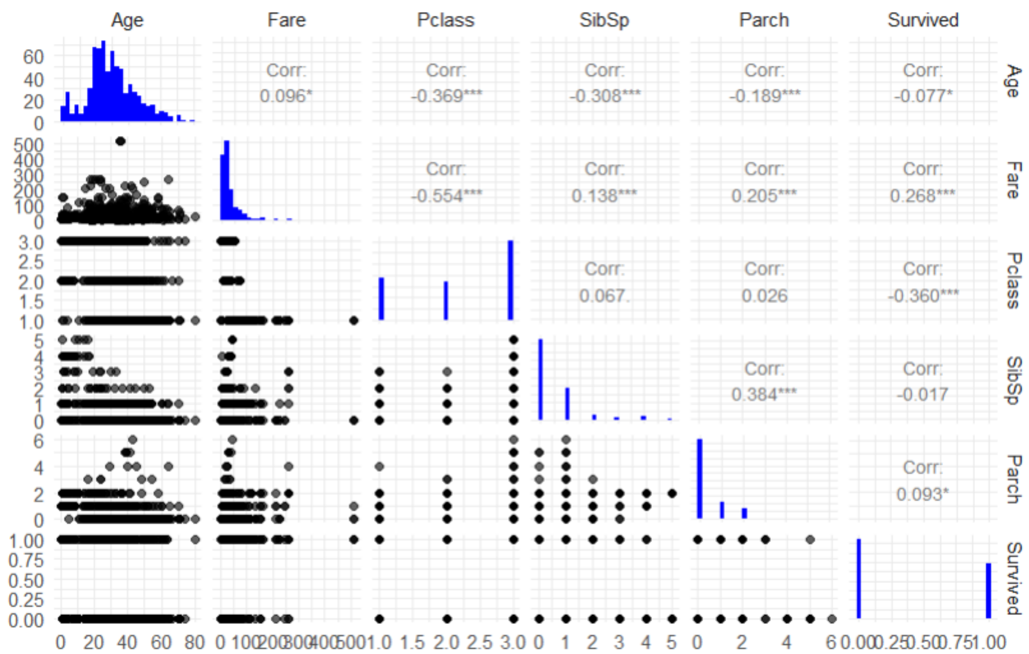
6.3 Change the appearance of the correlation matrix plot and heatmap in the multivariate analysis section

(Customized correlation matrix plot)

Code:

```
library(GGally)
numeric_vars <- titanic_train %>%
  select(Age, Fare, Pclass, SibSp, Parch, Survived)
ggpairs(numeric_vars,
        upper = list(continuous = wrap("cor", size = 3)),
        lower = list(continuous = wrap("points", alpha = 0.6)),
        diag = list(continuous = wrap("barDiag", fill = "blue")),
        axisLabels = "show") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "darkred", size = 14, face = "bold")
  )
```

Output:

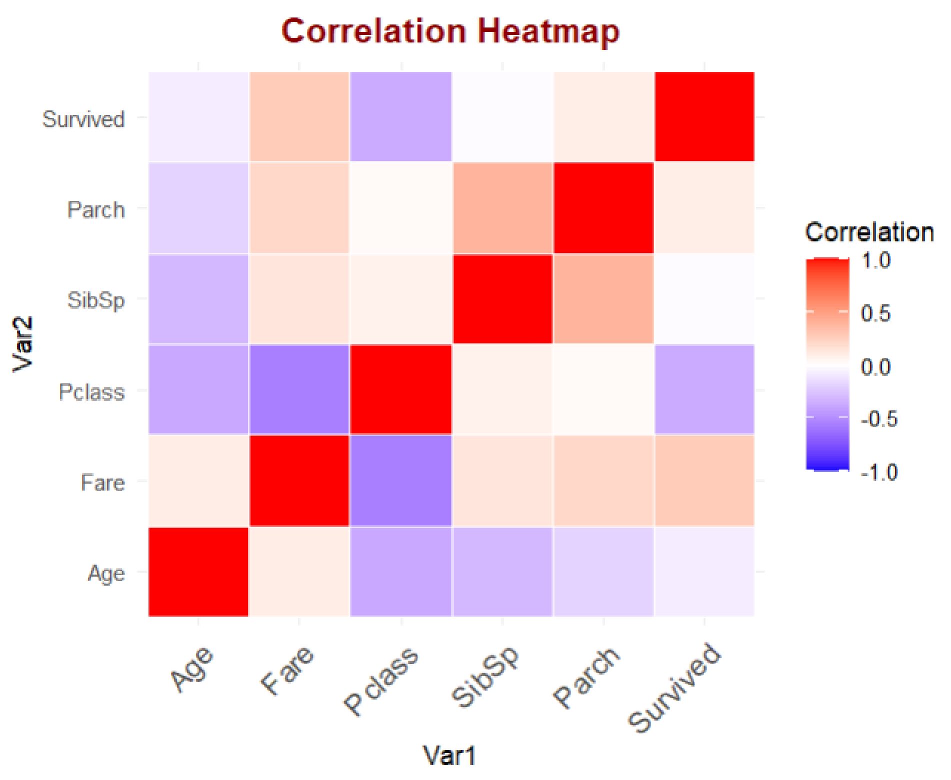


(Customized heatmap)

Code:

```
library(reshape2)
cor_matrix <- cor(numeric_vars, use = "complete.obs")
melted_cor_matrix <- melt(cor_matrix)
ggplot(data = melted_cor_matrix, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1), space = "Lab", name = "Correlation") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1),
    plot.title = element_text(hjust = 0.5, color = "darkred", size = 14, face = "bold")
  ) +
  labs(title = "Correlation Heatmap")
```

Output:



References:

https://archive.ics.uci.edu/datasets

https://www.w3schools.com/r/

https://ggplot2.tidyverse.org/

https://www.geeksforgeeks.org/data-visualization-in-r/

https://www.geeksforgeeks.org/exploratory-data-analysis-in-r-programming/

https://www.geeksforgeeks.org/data-cleaning-in-r/

https://dplyr.tidyverse.org/articles/dplyr.html

Word Count (excluding code, references, figures, tables, and appendices):

1114 words