Solo Group 10 - Project 2 - Karthik Ragunath Ananda Kumar

Solution 1 (a)

Code and Output

Custom utility function for reading CSV

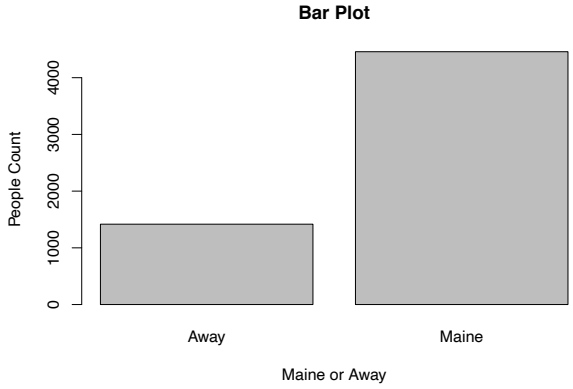
```
read_csv_func = function(x){
  df = read.csv(x, header=TRUE) # Read CSV
  return (df)
}
```

```
roadrace_df = read_csv_func("/Users/karthik_ragunath/Desktop/Stats/roadrace.csv")
# Function call to read csv
```

Custom utility function for plotting bar plots

```
plot_bar_plots = function(df, column_name, main_title, x_lab, y_lab){
  column_names_list = names(df)
  # get column list

  column_index = match(column_name, column_names_list)
  # get the index of where the column is present in the dataframe

  barplot(table(df[,column_index]), main=main_title, xlab=x_lab, ylab=y_lab)
  # Plot the barplot by using the column index
  # of the dataframe to extract the feature to plot
}
```

```
plot_bar_plots(df=roadrace_df, column_name="Maine", main_title="Bar Plot",
               x_lab="Maine or Away", y_lab="People Count")
```




```
# Use the custom utility function we have written to plot the bar graph
```

Custom Fucntion to extract rows based on some condition

```
extract_rows_based_on_condition = function(df, column_name, condition){
  columns_list = names(df) # Getting list of column names in dataframe

  column_index = match(column_name, columns_list)
  # Getting index of column where are looking for from the column list

  df_constraint = df[df[,column_index] == condition,]
  # Applying constraint to get rows which are matching a condition
  return (df_constraint)
}
```

```
maine_group_df = extract_rows_based_on_condition(df=roadrace_df,
                                                 column_name="Maine",
                                                 condition="Maine")
away_group_df = extract_rows_based_on_condition(df=roadrace_df,
                                                column_name="Maine",
                                                condition="Away")
percentage_from_maine = as.double(nrow(maine_group_df) / nrow(roadrace_df))
percentage_from_away = as.double(nrow(away_group_df) / nrow(roadrace_df))
difference_in_percentage = (percentage_from_maine - percentage_from_away) * 100
```

```
cat("% Maine: ", percentage_from_maine, sep='')
```

```
## % Maine: 0.7588085
```

```
cat("- %Away: ", percentage_from_away, sep='')
```

```
## - %Away: 0.2411915
```

```
cat("- %Difference between Maine and Away: ", difference_in_percentage, sep='')
```

```
## - %Difference between Maine and Away: 51.7617
```

Conclusion:

From the bar plots, it is clear that number of runners from Maine who are participating in this road race is much greater than people away from Maine. To be precise, 75.8% belong to Maine and 24.1% belong to Away and there is a difference in percentage of 51.76%

Solution 1 (b)

Code and Outputs

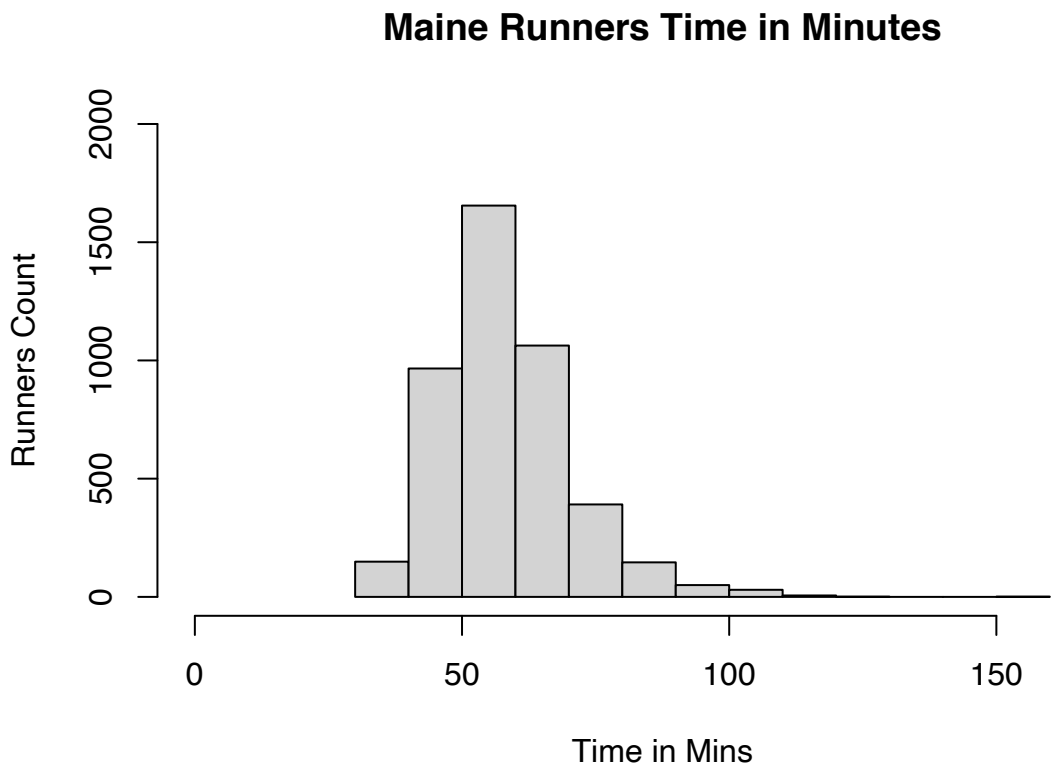
```
maine_group_df = extract_rows_based_on_condition(df=roadrace_df,
                                                 column_name="Maine",
                                                 condition="Maine")
# extract rows based on condition using utility function we have  created.
# The utility function code was already explained in previous question.

away_group_df = extract_rows_based_on_condition(df=roadrace_df,
                                                column_name="Maine",
                                                condition="Away")
# Repeat the same to extract data from away column Maine
#which contains string "Away"
```

Custom utility function to plot histograms

```
plot_histogram = function(df, column_name, main_title, x_lab, y_lab,
                          apply_lims=FALSE, custom_x_lim=c(0,100),
                          custom_y_lim=c(0,1000)){
  columns_list = names(df)
  # Get the list of columns in dataframe

  column_index = match(column_name, columns_list)
  # Get the column index of the column name we are looking for

  if(apply_lims == FALSE)
  {
    hist(df[,column_index], main=main_title, xlab=x_lab, ylab=y_lab)
    # Plot the histogram from extracted column data
  }
  else
  {
    hist(df[,column_index], main=main_title, xlab=x_lab, ylab=y_lab,
         xlim=custom_x_lim, ylim=custom_y_lim)
    # Plot the histogram from extracted column data
  }
}
```

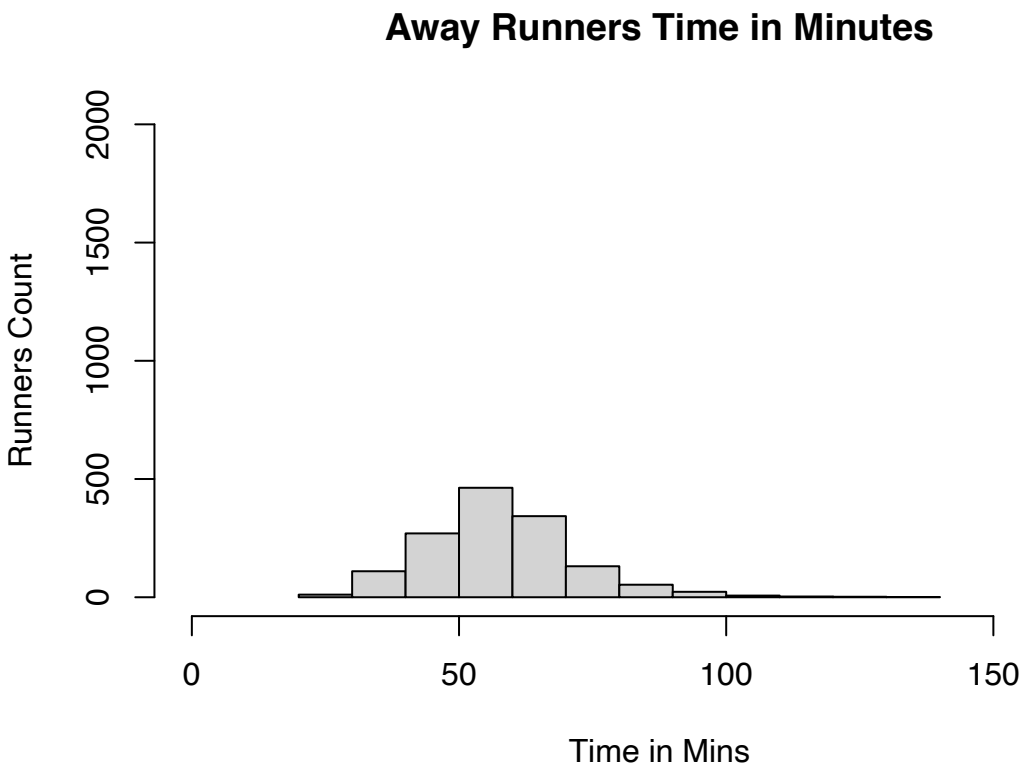
Plot histograms of finish times for runners from Maine using utility function we have created

```
plot_histogram(df=maine_group_df, column_name="Time..minutes.",
               main_title="Maine Runners Time in Minutes", x_lab="Time in Mins",
               y_lab="Runners Count", apply_lims=TRUE, custom_x_lim=c(0, 175),
               custom_y_lim=c(0,2000))
```



Plot histograms of finish times for runners from Away using utility function we have created

```
plot_histogram(df=away_group_df, column_name="Time..minutes.",
               main_title="Away Runners Time in Minutes", x_lab="Time in Mins",
               y_lab="Runners Count", apply_lims=TRUE, custom_x_lim=c(0, 175),
               custom_y_lim=c(0,2000))
```



Custom utility function to obtain summary statistics

```
stat_info = function(df, column_name){

  # Extract column data of interest
  columns_names_list = names(df)
  column_index = match(column_name, columns_names_list)
  x = df[, column_index]

  if(typeof(x) == "character") # if column data type is character,
                               # we typecast it to Double
  {
    x = as.double(x)
  }
  mean_x = mean(x) # Get the mean of data
  sd_x = sd(x) # Get the Standard Deviation
  # range_x = max(x) - min(x)
  range_x = range(x) # Get the range
  median_x = median(x) # Get the median
  iqr_x = IQR(x) # Get the IQR
  max_x = max(x) # Get the max
  min_x = min(x) # Get the min
  stats_list = c(mean_val=mean_x, sd_val=sd_x, range_val=range_x,
                 median_val=median_x, iqr_val=iqr_x, max_val=max_x,
```

```
                min_val=min_x)
  #create a list from summary statistics we obtained
  return (stats_list)
}
```

```
stats_data_maine = stat_info(df=maine_group_df, column_name="Time..minutes.")
# Using our custom utility function to obtain summary statistics
stats_data_maine
```

mean_val sd_val range_val1 range_val2 median_val iqr_val max_val 58.19514 12.18511 30.56700 152.16700 57.03350 14.24775 152.16700 min_val 30.56700

Verifying output of our custom utitlity function to obtain summary statistics with R's summary function

Custom utility function to extract specific column from dataframe

```
extract_column_func = function(df, column_name, keep_char=FALSE){
  column_names_list = names(df)
  column_index = match(column_name, column_names_list)
  column_data = df[, column_index]
  if(typeof(column_data) == "character" && keep_char==FALSE)
  {
    column_data = as.double(column_data)
  }
  return (column_data)
}
```

```
maine_grp_time_mins = extract_column_func(df=maine_group_df,
                                          column_name="Time..minutes.")
# Using our custom utility function to extract column data
stats_maine_summary = summary(maine_grp_time_mins)
stats_maine_summary # Printing summary statistics
```

Min. 1st Qu. Median Mean 3rd Qu. Max. 30.57 50.00 57.03 58.20 64.24 152.17

Obtaining summary statistics for finish times of Away Group Runners

```
stats_data_away = stat_info(df=away_group_df, column_name="Time..minutes.")
# Using custom utility function to extract summary finish time statistics for away group
stats_data_away
```

mean_val sd_val range_val1 range_val2 median_val iqr_val max_val 57.82181 13.83538 27.78200 133.71000 56.92000 15.67400 133.71000 min_val 27.78200

```
away_grp_time_mins = extract_column_func(df=away_group_df,
                                         column_name="Time..minutes.")
# Using summary function to get summary for finish time of away group
stats_away_summary = summary(away_grp_time_mins)
stats_away_summary
```

Min. 1st Qu. Median Mean 3rd Qu. Max. 27.78 49.15 56.92 57.82 64.83 133.71

Custom utility function to generate table with summary statistics

```
print_info_table_func = function(data_info_1, data_info_2, summary_1, summary_2,
                                 grp_name_1, grp_name_2, caption){
  df = data.frame(matrix(ncol = 4, nrow = 0))
  x = c("Index", "Groups", grp_name_1, grp_name_2)
  colnames(df) = x
  df[nrow(df) + 1,] = c("(i)", "Mean Val", data_info_1['mean_val'],
                        data_info_2['mean_val'])
  df[nrow(df) + 1,] = c("(ii)", "Median Val", data_info_1['median_val'],
                        data_info_2['median_val'])
  df[nrow(df) + 1,] = c("(iii)", "SD Val", data_info_1['sd_val'],
                        data_info_2['sd_val'])
  df[nrow(df) + 1,] = c("(iv)", "Range Val", paste(c(data_info_1['range_val1'],
                                                     data_info_1['range_val2']),
                                                      collapse=' '),
                        paste(c(data_info_2['range_val1'],
                                data_info_2['range_val2']), collapse=' '))
  df[nrow(df) + 1,] = c("(v)", "IQR Val", data_info_1['iqr_val'],
                        data_info_2['iqr_val'])
  df[nrow(df) + 1,] = c("(vi)", "Quantile 1", summary_1['1st Qu.'],
                        summary_2['1st Qu.'])
  df[nrow(df) + 1,] = c("(vii)", "Quantile 3", summary_1['3rd Qu.'],
                        summary_2['3rd Qu.'])
  df[nrow(df) + 1,] = c("(viii)", "Max", data_info_1['max_val'],
                        data_info_2['max_val'])
  df[nrow(df) + 1,] = c("(ix)", "Min", data_info_1['min_val'],
                        data_info_2['min_val'])
  kable(df, caption=caption)
}
```

Printing out our summary statistics for finish times for runners from Maine and Away

```
print_info_table_func(data_info_1 = stats_data_maine, data_info_2 =
                        stats_data_away, summary_1 = stats_maine_summary,
                      summary_2 = stats_away_summary, grp_name_1 = "Maine",
                      grp_name_2 = "Away",
                      caption = "Maine and Away Runners Comparison")
```

Table 1: Maine and Away Runners Comparison

Index	Groups	Maine	Away
(i)	Mean Val	58.1951381785554	57.8218136908962
(ii)	Median Val	57.0335	56.92
(iii)	SD Val	12.1851105531497	13.8353842414778
(iv)	Range Val	30.567 152.167	27.782 133.71
(v)	IQR Val	14.24775	15.674

Index	Groups	Maine	Away
(vi)	Quantile 1	49.9955	49.153
(vii)	Quantile 3	64.24325	64.827
(viii)	Max	152.167	133.71
(ix)	Min	30.567	27.782

```
Conclusion:

From the histograms, it is clear that both the histograms are right
skewed as mean is greater than median.
Also from table, it is clear that runners from Maine have higher Min, Q1,
Median, Mean, Max finish times compared to runners from Away.
And the value of Q3,SD,IQR of finish times are higher for the runners from
Away group.
```

Solution 1 (c)

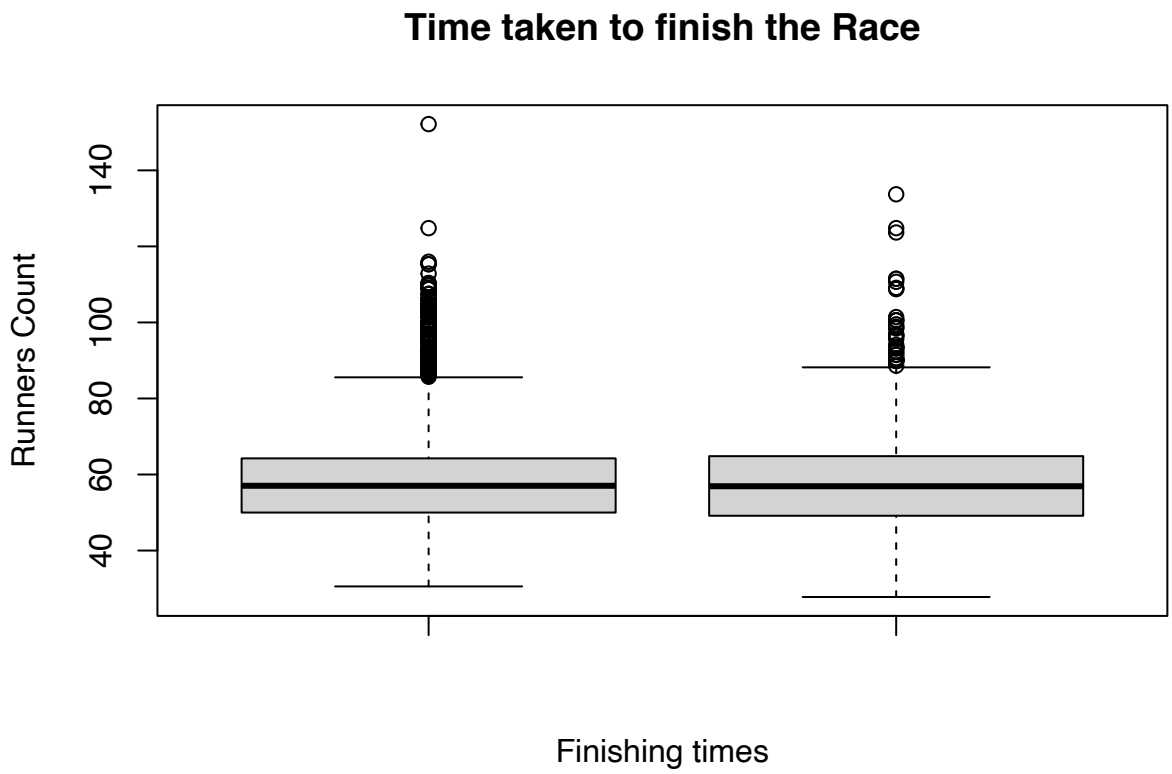
Utility function to plot side by side boxplots

```
side_by_side_box_plots = function(df_1, column_name_1, df_2, column_name_2,
                                  main_title, x_lab, y_lab, keep_char=FALSE){

  # Extarct column data for two features we are plotting box plots side by side
  column_list_1 = names(df_1)
  column_list_2 = names(df_2)
  column_index_1 = match(column_name_1, column_list_1)
  column_index_2 = match(column_name_2, column_list_2)
  column_data_1 = df_1[, column_index_1]
  column_data_2 = df_2[, column_index_2]

  # Typecast columns to double if our columns of our interests are of type character
  if(typeof(column_data_1) == "character" && keep_char==FALSE)
  {
    column_data_1 = as.double(column_data_1)
  }
  if(typeof(column_data_2) == "character" && keep_char==FALSE)
  {
    column_data_2 = as.double(column_data_2)
  }

  # Plot box plots side by side
  boxplot(column_data_1, column_data_2, main=main_title, xlab=x_lab, ylab=y_lab)
}
```

Plot box plots side by side for times to finish for runners from Maine and Away

```
side_by_side_box_plots(df_1=maine_group_df, column_name_1="Time..minutes.",
                       df_2=away_group_df, column_name_2="Time..minutes.",
```

```
                    main_title="Time taken to finish the Race",
                    x_lab="Finishing times", y_lab="Runners Count")
```



Solution 1 (d)

Extrarct runners ages for Male and Female genders using the custom utitity function we have created previously

```
male_group_df = extract_rows_based_on_condition(df=roadrace_df,
                                                column_name="Sex", condition="M")
female_group_df = extract_rows_based_on_condition(df=roadrace_df,
                                                  column_name="Sex",
                                                  condition="F")
```

Get the stats for age distributions of male runners using the custom utility function we have created

```
stats_male_age_info = stat_info(df=male_group_df, column_name="Age")
stats_male_age_info
```

mean_val sd_val range_val1 range_val2 median_val iqr_val max_val 40.44680 13.99289 9.00000 83.00000 41.00000 21.00000 83.00000 min_val 9.00000

Verify our stats with summary stats of R

```
male_grp_age = extract_column_func(df=male_group_df, column_name="Age")
male_grp_summary = summary(male_grp_age)
```

Get the stats for age distributions of female runners using the custom utility function we have created

```
stats_female_age_info = stat_info(df=female_group_df, column_name="Age")
stats_female_age_info
```

mean_val sd_val range_val1 range_val2 median_val iqr_val max_val 37.23653 12.26925 7.00000 86.00000 36.00000 18.00000 86.00000 min_val 7.00000

Verify our stats with summary stats of R

```
female_grp_age = extract_column_func(df=female_group_df, column_name="Age")
female_grp_summary = summary(female_grp_age)
```

Printing out our summary statistics for age distributions of male and female runners

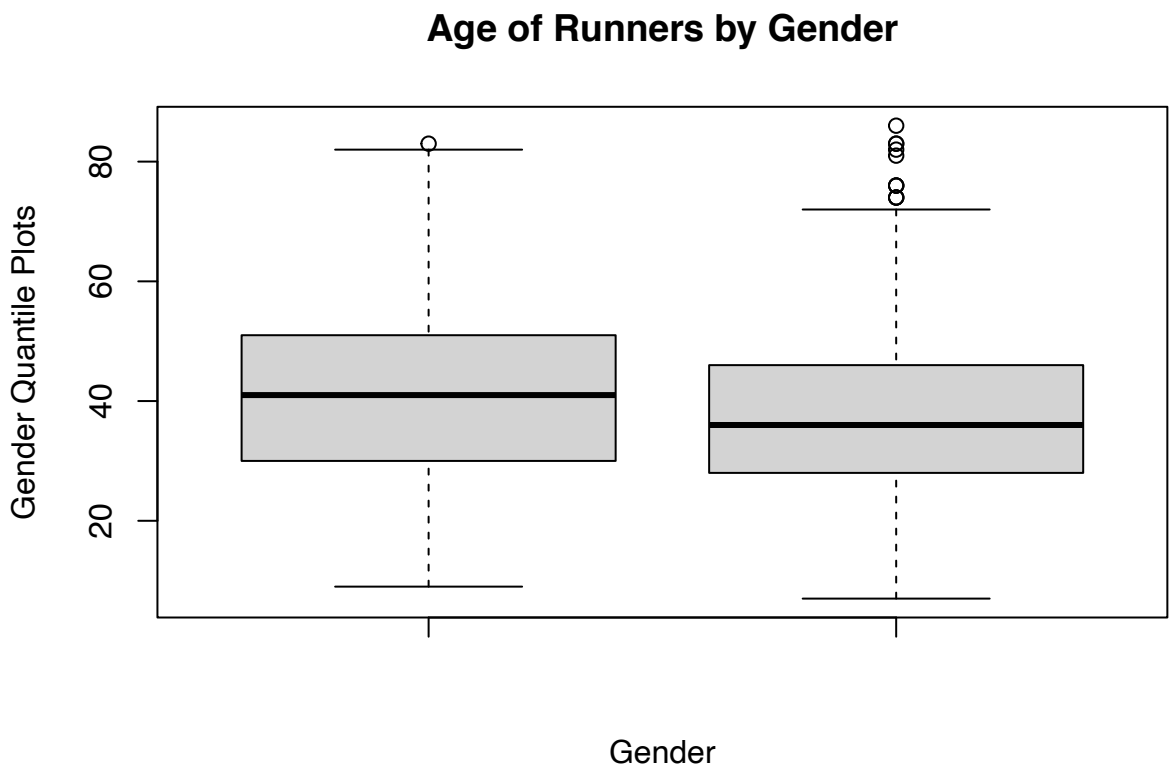
```
print_info_table_func(data_info_1 = stats_male_age_info,
                      data_info_2 = stats_female_age_info,
                      summary_1 = male_grp_summary,
                      summary_2 = female_grp_summary, grp_name_1 = "Male",
                      grp_name_2 = "Female",
                      caption = "Male and Female Runners Comparison")
```

Table 2: Male and Female Runners Comparison

Index	Groups	Male	Female
(i)	Mean Val	40.4468012316115	37.236529989834
(ii)	Median Val	41	36
(iii)	SD Val	13.9928905091845	12.2692519361855
(iv)	Range Val	9 83	7 86
(v)	IQR Val	21	18
(vi)	Quantile 1	30	28
(vii)	Quantile 3	51	46
(viii)	Max	83	86
(ix)	Min	9	7

Plotting box plots for age distributions of male and female runners

```
side_by_side_box_plots(df_1=male_group_df, column_name_1="Age",
                       df_2=female_group_df, column_name_2="Age",
                       main_title="Age of Runners by Gender", x_lab="Gender",
                       y_lab="Gender Quantile Plots")
```




```
Conclusion:

From the output we observe that age distribtion of male runner have higher Q1,
mean, median, Q3, SD and IQR values compared to age distribution of female runners.
Also, the oldest and the youngest runners are both from female category.
```

Solution 2

Using our custom utility function to read the motorcycles dataset which conatins information on number of fatal accidents in each county of South Carolina

```
motorcycle_df = read_csv_func("/Users/karthik_ragunath/Desktop/Stats/motorcycle.csv")
```

Get the statistics on fatal motorcycle accidents using our custom utility function

```
accident_stats_info = stat_info(df=motorcycle_df,
                                column_name="Fatal.Motorcycle.Accidents")
accident_stats_info
```

mean_val sd_val range_val1 range_val2 median_val iqr_val max_val 17.02083 13.81256 0.00000 60.00000 13.50000 17.00000 60.00000 min_val 0.00000

Verify the statistics we obtained on fatal motorcycle accidents with R's summary function

```
accidents = extract_column_func(df=motorcycle_df,
                                column_name="Fatal.Motorcycle.Accidents")
accidents_summary = summary(accidents)
accidents_summary
```

Min. 1st Qu. Median Mean 3rd Qu. Max. 0.00 6.00 13.50 17.02 23.00 60.00

Custom utiltiy function to print summary statistics

```
print_info_table_induvidual_func = function(data_info_1, summary_1, grp_name, caption){
  df = data.frame(matrix(ncol = 3, nrow = 0))
  x = c("Index", "Groups", grp_name)
  colnames(df) = x
  df[nrow(df) + 1,] = c("(i)", "Mean Val", data_info_1['mean_val'])
  df[nrow(df) + 1,] = c("(ii)", "Median Val", data_info_1['median_val'])
  df[nrow(df) + 1,] = c("(iii)", "SD Val", data_info_1['sd_val'])
  df[nrow(df) + 1,] = c("(iv)", "Range Val", paste(c(data_info_1['range_val1'],
                                                     data_info_1['range_val2']),
                                                   collapse=' '))
  df[nrow(df) + 1,] = c("(v)", "IQR Val", data_info_1['iqr_val'])
  df[nrow(df) + 1,] = c("(vi)", "Quantile 1", summary_1['1st Qu.'])
  df[nrow(df) + 1,] = c("(vii)", "Quantile 3", summary_1['3rd Qu.'])
  df[nrow(df) + 1,] = c("(viii)", "Max", data_info_1['max_val'])
  df[nrow(df) + 1,] = c("(ix)", "Min", data_info_1['min_val'])
  kable(df, caption=caption)
}
```

Use our custom utility function to print out the statistics on fatal motorcycle accidents per county in Southern Carolina

```
print_info_table_induvidual_func(data_info_1 = accident_stats_info,
                                 summary_1 = accidents_summary,
                                 grp_name = "Accidents",
                                 caption = "Accident Summary Statistics")
```

Table 3: Accident Summary Statistics

Index	Groups	Accidents
(i)	Mean Val	17.0208333333333
(ii)	Median Val	13.5
(iii)	SD Val	13.8125591683852
(iv)	Range Val	0 60
(v)	IQR Val	17
(vi)	Quantile 1	6
(vii)	Quantile 3	23
(viii)	Max	60
(ix)	Min	0

Utility function to plot boxplots

```
box_plots = function(df, column_name, main_title, x_lab, y_lab, keep_char=FALSE){

  # Extarct column data for two features we are plotting box plots side by side
  column_list = names(df)
  column_index = match(column_name, column_list)
  column_data = df[, column_index]

  # Typecast columns to double if our columns of our interests are of type character
  if(typeof(column_data) == "character" && keep_char==FALSE)
  {
    column_data = as.double(column_data)
  }

  # Plot box plots
  boxplot(column_data, main=main_title, xlab=x_lab, ylab=y_lab)
}
```

Plot boxplot for fatal accidents ocuring per county in South Carolina

```
box_plots(df=motorcycle_df, column_name="Fatal.Motorcycle.Accidents",
          main_title="Box plot of Fatal Accidents in each counties of South Carolina",
          x_lab="Fatal Accidents", y_lab="Fatal Accidents Quantile Plot")
```

Getting the outlier counties

```
# computing cutoffs for identifying the outliers
min=quantile(accidents,prob=0.25)-(1.5)*IQR(accidents)
max=quantile(accidents,prob=0.75)+(1.5)*IQR(accidents)

# extract counties column information
county_data = extract_column_func(df=motorcycle_df, column_name="County",
                                  keep_char=TRUE)

# compute the outliers by identifying counties which lies outside the cutoffs
outlier=county_data[which((accidents<min)|(accidents>max))]
outlier
```

[1] "GREENVILLE" "HORRY"

The outlier are points which doesn't belong to range[-1.5*IQR-Q1,Q3+1.5*IQR]
So the outliers are "GREENVILLE", "HORRY".

The reason for more fatalities in these areas might be less awareness of traffic rules,
casual attitude towards traffic rules, improper road infrastructure,
high use of alcohol and drugs, etc..