Batch: E2 Roll No.: 16010123325

Experiment No. 4

Title: Apply descriptive statistics techniques in R to summarize and interpret data

Aim : To apply various descriptive statistics techniques, such as measures of central tendency, variability, and distribution, to analyze and summarize the key features of a dataset.

Course Outcomes:

CO1, CO2, CO3

Books/ Journals/ Websites referred:

1. The Comprehensive R Archive Network
2. Posit

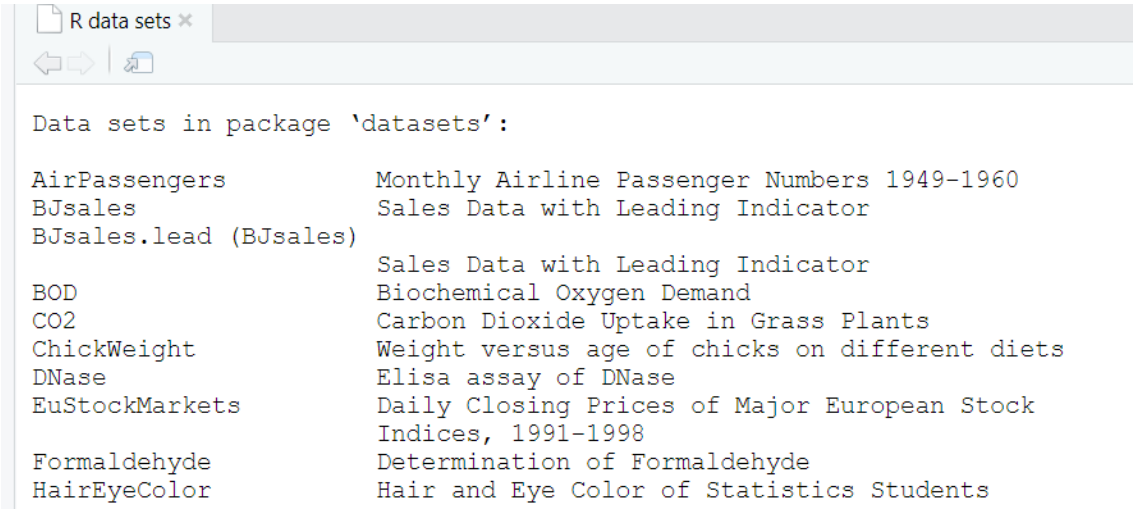
Resources used:

(Students should write)

Select a built-in R dataset

You can see a list of all the built-in datasets using the data() function.

```
> data()
```



Here, we'll use the built-in R data set named *iris*. Every student in the batch has to choose a unique dataset.

```
> # Store the data in the variable my_data
> my_data <- iris
```

Check your data

You can inspect your data using the functions **head**() and **tails**(), which will display the first and the last part of the data, respectively.

```
> # Print the first 6 rows
> head(my_data, 6)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

R functions for computing descriptive statistics

Description	R function

Mean	**mean**()
Standard deviation	**sd**()
Variance	**var**()
Minimum	**min**()
Maximum	**maximum**()
Median	**median**()
Range of values (minimum and maximum)	**range**()
Sample quantiles	**quantile**()
Generic function	**summary**()
Interquartile range	**IQR**()

Descriptive statistics for a single group

Measure of central tendency: mean, median, mode

```
> # Compute the mean value
> mean(my_data$Sepal.Length)
[1] 5.843333
>
> # Compute the median value
> median(my_data$Sepal.Length)
[1] 5.8
```

The function mfv() [in the modeest R package] can be used to compute the mode of a variable.

```
> # Compute the mode
> install.packages("modeest")
```

```
> require(modeest)
Loading required package: modeest
> mfv(my_data$Sepal.Length)
[1] 5
```

Measure of variability

Range: minimum & maximum

```
> # Compute the minimum value
> min(my_data$Sepal.Length)
[1] 4.3
> # Compute the maximum value
> max(my_data$Sepal.Length)
[1] 7.9
> # Range
> range(my_data$Sepal.Length)
[1] 4.3 7.9
```

Quantiles

```
> quantile(my_data$Sepal.Length)
  0%  25%  50%  75% 100%
 4.3  5.1  5.8  6.4  7.9
```

By default, the function returns the minimum, the maximum and three **quartiles** (the 0.25, 0.50 and 0.75 quantiles).

```
> quantile(my_data$Sepal.Length, seq(0, 1, 0.25))
  0%  25%  50%  75% 100%
 4.3  5.1  5.8  6.4  7.9
```

To compute deciles (0.1, 0.2, 0.3, …., 0.9), use this:

```
> quantile(my_data$Sepal.Length, seq(0, 1, 0.1))
  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
4.30 4.80 5.00 5.27 5.60 5.80 6.10 6.30 6.52 6.90 7.90
```

Interquartile range

```
> IQR(my_data$Sepal.Length)
[1] 1.3
```

Variance and standard deviation

```
> # Compute the variance
> var(my_data$Sepal.Length)
[1] 0.6856935
> # Compute the standard deviation =
> # square root of th variance
> sd(my_data$Sepal.Length)
[1] 0.8280661
```

Median absolute deviation

```
> # Compute the median absolute deviation
> mad(my_data$Sepal.Length)
[1] 1.03782
```

Computing an overall summary of a variable and an entire data frame

summary() function

Summary of a single variable. Five values are returned: the mean, median, 25th and 75th quartiles, min and max in one single line call:

```
> summary(my_data$Sepal.Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900
```

Summary of a data frame. In this case, the function **summary**() is automatically applied to each column. The format of the result depends on the type of the data contained in the column. For example:

- If the column is a numeric variable, mean, median, min, max and quartiles are returned.
- If the column is a factor variable, the number of observations in each group is returned.

```
> summary(my_data, digits = 2)
  Sepal.Length  Sepal.Width   Petal.Length  Petal.Width          Species
 Min.   :4.3   Min.   :2.0   Min.   :1.0   Min.   :0.1   setosa     :50
 1st Qu.:5.1   1st Qu.:2.8   1st Qu.:1.6   1st Qu.:0.3   versicolor:50
 Median :5.8   Median :3.0   Median :4.3   Median :1.3   virginica :50
 Mean   :5.8   Mean   :3.1   Mean   :3.8   Mean   :1.2
 3rd Qu.:6.4   3rd Qu.:3.3   3rd Qu.:5.1   3rd Qu.:1.8
 Max.   :7.9   Max.   :4.4   Max.   :6.9   Max.   :2.5
```

sapply() function

It's also possible to use the function **sapply**() to apply a particular function over a list or vector. For instance, we can use it to compute for each column in a data frame, the mean, sd, var, min, quantile, …

```
> # Compute the mean of each column
> sapply(my_data[, -5], mean)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
```

```
> # Compute quartiles
> sapply(my_data[, -5], quantile)
     Sepal.Length Sepal.Width Petal.Length Petal.Width
0%            4.3         2.0         1.00         0.1
25%           5.1         2.8         1.60         0.3
50%           5.8         3.0         4.35         1.3
75%           6.4         3.3         5.10         1.8
100%          7.9         4.4         6.90         2.5
```

Graphical display of distributions

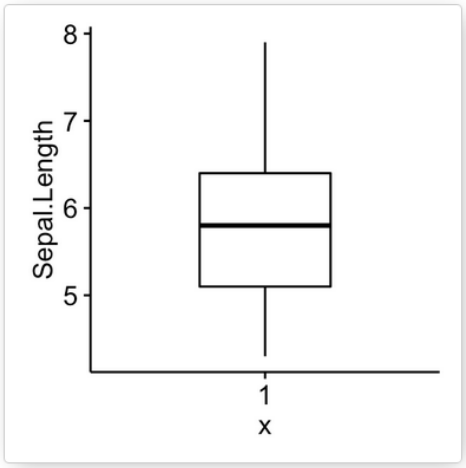
The R package **ggpubr** will be used to create graphs.

```
install.packages("ggpubr")
```

```
library(ggpubr)
```

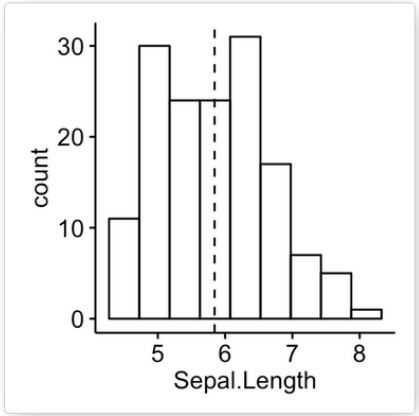
Box Plot

```
ggboxplot(my_data, y = "Sepal.Length", width = 0.5)
```



Histogram with mean line

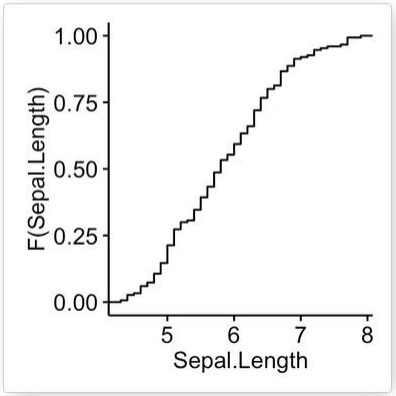
```
gghistogram(my_data, x = "Sepal.Length", bins = 9,
            add = "mean")
```



Empirical cumulative distribution function (ECDF)

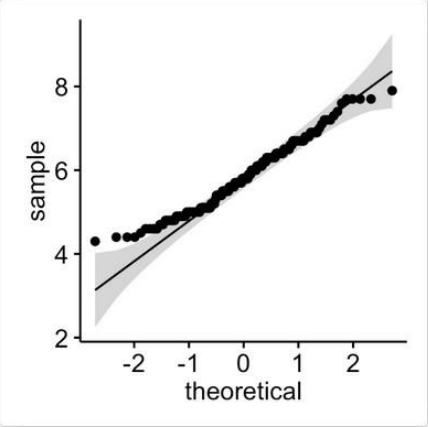
ECDF is the fraction of data smaller than or equal to x.

```
ggecdf(my_data, x = "Sepal.Length")
```



QQ plots are used to check whether the data is normally distributed.

```
ggqqplot(my_data, x = "Sepal.Length")
```



Descriptive statistics by groups

To compute summary statistics by groups, the functions **group_by**() and **summarise**() [in **dplyr** package] can be used.

- We want to group the data by *Species* and then:
  - compute the number of element in each group. R function: **n**()
  - compute the mean. R function **mean**()
  - and the standard deviation. R function **sd**()

Install **ddplyr** as follow:

```
install.packages("dplyr")
```

Descriptive statistics by groups:

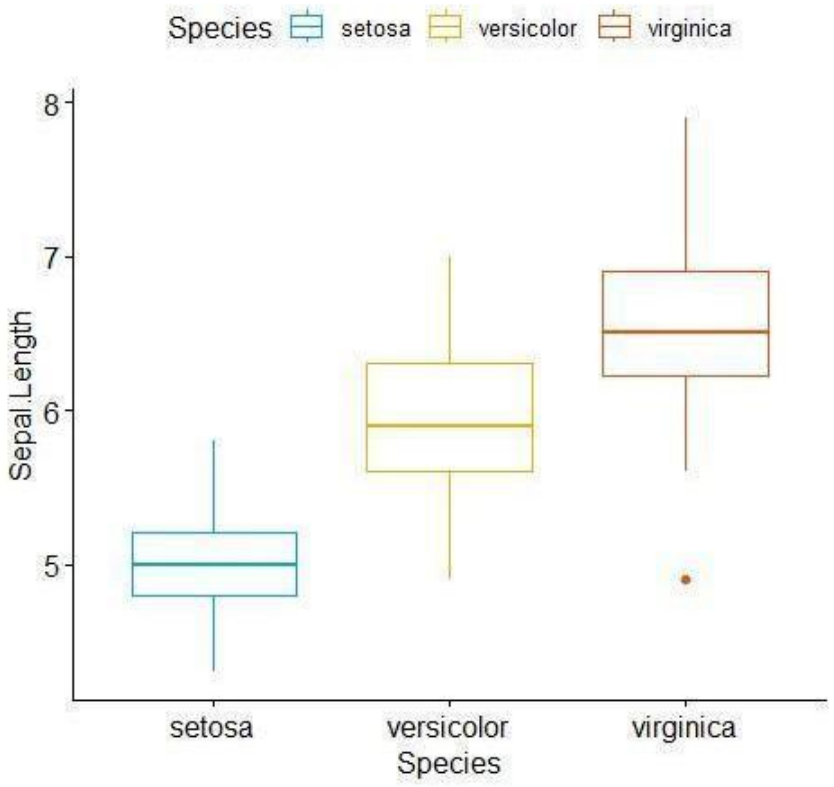
To compute summary statistics by groups, the functions **group_by**() and **summarise**() [in **dplyr** package] can be used.

- We want to group the data by *Species* and then:
  - compute the number of element in each group. R function: **n**()
  - compute the mean. R function **mean**()
  - and the standard deviation. R function **sd**()

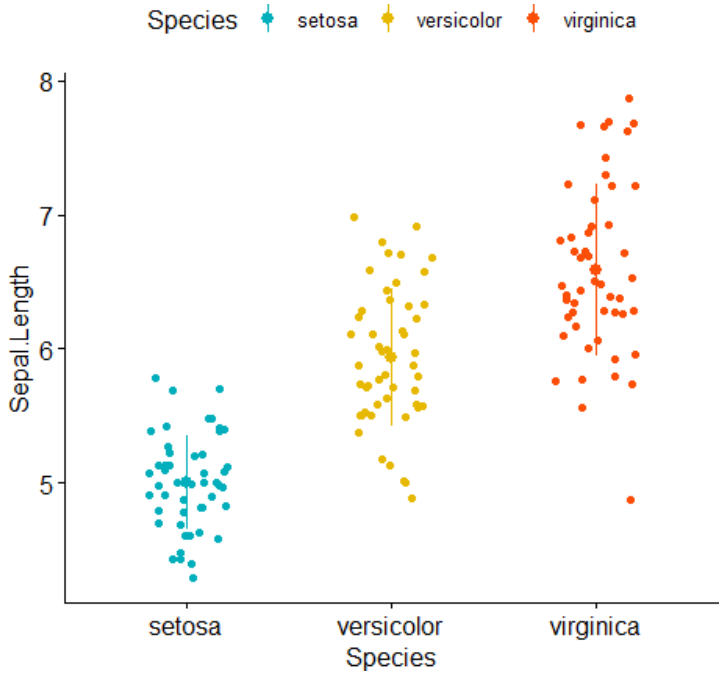
%>% is used to chain the operations.

Graphics for grouped data:

```
> # Box plot colored by groups: Species
> ggboxplot(my_data, x = "Species", y = "Sepal.Length",
+           color = "Species",
+           palette = c("#00AFBB", "#E7B800", "#FC4E07"))
```




```
> # Stripchart colored by groups: Species
> ggstripchart(my_data, x = "Species", y = "Sepal.Length",
+              color = "Species",
+              palette = c("#00AFBB", "#E7B800", "#FC4E07"),
+              add = "mean_sd")
```



Frequency tables

A frequency table (or contingency table) is used to describe categorical variables. It contains the counts at each combination of factor levels.

R function to generate tables: **table**()

For this section we will use the built-in R dataset that contains the distribution of hair and eye color by sex of 592 students:

```
> # Hair/eye color data
> df <- as.data.frame(HairEyeColor)
> hair_eye_col <- df[rep(row.names(df), df$Freq), 1:3]
> rownames(hair_eye_col) <- 1:nrow(hair_eye_col)
> head(hair_eye_col)
   Hair   Eye  Sex
1 Black Brown Male
2 Black Brown Male
3 Black Brown Male
4 Black Brown Male
5 Black Brown Male
6 Black Brown Male
```

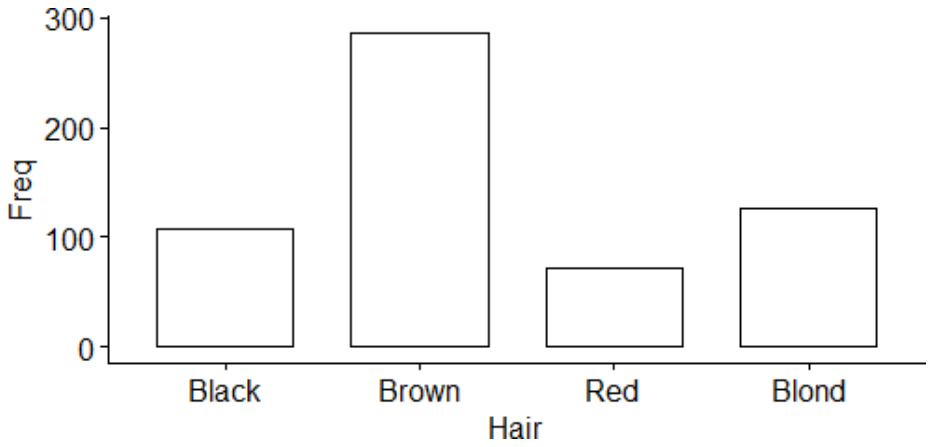
Simple frequency distribution: one categorical variable

Table of counts

```
> # hair/eye variables
> Hair <- hair_eye_col$Hair
> Eye <- hair_eye_col$Eye
> # Frequency distribution of hair color
> table(Hair)
Hair
Black Brown   Red Blond
  108   286    71   127
> # Frequency distribution of eye color
> table(Eye)
Eye
Brown  Blue Hazel Green
  220   215    93    64
```

Visualization:

```
> # Visualize using bar plot
> library(ggpubr)
> ggbarplot(df, x = "Hair", y = "Freq")
```



Two-way contingency table: Two categorical variables

```
> hair_eye <- table(Hair , Eye)
> hair_eye
       Eye
Hair    Brown Blue Hazel Green
  Black    68   20    15     5
  Brown   119   84    54    29
  Red      26   17    14    14
  Blond     7   94    10    16
```

Multiway tables: More than two categorical variables

- Hair and Eye color distributions by sex using **xtabs**():

```
> xtabs(~Hair + Eye + Sex, data = hair_eye_col)
, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

You can also use the function **ftable**() [for flat contingency tables]. It returns a cleaner looking output compared to xtabs() when you have more than two variables:

```
> ftable(Sex + Hair ~ Eye, data = hair_eye_col)
      Sex   Male                  Female
      Hair Black Brown Red Blond  Black Brown Red Blond
Eye
Brown          32    53  10     3     36    66  16     4
Blue           11    50  10    30      9    34   7    64
Hazel          10    25   7     5      5    29   7     5
Green           3    15   7     8      2    14   7     8
```

Compute table margins and relative frequency

Table margins correspond to the sums of counts along rows or columns of the table.

```
> # Margin of rows
> margin.table(hair_eye, 1)
Hair
Black Brown   Red Blond
  108   286    71   127

> # Margin of columns
> margin.table(hair_eye, 2)
Eye
Brown  Blue Hazel Green
  220   215    93    64
```

Relative frequencies express table entries as proportions of table margins (i.e., row or column totals).

```
> # Frequencies relative to row total
> prop.table(hair_eye, 1)
       Eye
Hair         Brown       Blue      Hazel      Green
  Black 0.62962963 0.18518519 0.13888889 0.04629630
  Brown 0.41608392 0.29370629 0.18881119 0.10139860
  Red   0.36619718 0.23943662 0.19718310 0.19718310
  Blond 0.05511811 0.74015748 0.07874016 0.12598425
> # Table of percentages
> round(prop.table(hair_eye, 1), 2)*100
       Eye
Hair    Brown Blue Hazel Green
  Black    63   19    14     5
  Brown    42   29    19    10
  Red      37   24    20    20
  Blond     6   74     8    13
> # Table of percentages
> round(prop.table(hair_eye, 1), 4)*100
       Eye
Hair    Brown  Blue Hazel Green
  Black 62.96 18.52 13.89  4.63
  Brown 41.61 29.37 18.88 10.14
  Red   36.62 23.94 19.72 19.72
  Blond  5.51 74.02  7.87 12.60
```

Students have to perform **ALL** the operations shown above on a different dataset of their choice (unique within the lab-batch) and add their screenshots here. Students can use datasets inbuilt within R, or students can download and use freely available datasets from Kaggle, UCI repository, etc.

Using dataset->Trees

```
> data()
> my_data<-trees
```

```
> head(my_data,6)
  Girth Height Volume
1   8.3     70   10.3
2   8.6     65   10.3
3   8.8     63   10.2
4  10.5     72   16.4
5  10.7     81   18.8
6  10.8     83   19.7
```

```
> mean(my_data$Girth)
[1] 13.24839
> median(my_data$Girth)
[1] 12.9
```

Mode:

```
> mfv(my_data$Girth)
[1] 11.0 11.4 12.9 18.0
```

Range: minimum & maximum:

```
> min(my_data$Girth)
[1] 8.3
> max(my_data$Girth)
[1] 20.6
> range(my_data$Girth)
[1]  8.3 20.6
```

Quantiles:

```
> quantile(my_data$Girth)
   0%   25%   50%   75%  100%
 8.30 11.05 12.90 15.25 20.60
> quantile(my_data$Girth, seq(0,1,0.25))
   0%   25%   50%   75%  100%
 8.30 11.05 12.90 15.25 20.60
> quantile(my_data$Girth, seq(0,1,0.1))
  0%  10%  20%  30%  40%  50%  60%  70%
 8.3 10.5 11.0 11.2 11.4 12.9 13.7 14.2
 80%  90% 100%
16.3 17.9 20.6
```

Interquartile range:

```
> IQR(my_data$Girth)
[1] 4.2
```

Variance and standard deviation:

```
> #Compute the variance
> var(my_data$Girth)
[1] 9.847914
> sd(my_data$Girth)
[1] 3.138139
```

Median absolute deviation:

```
> mad(my_data$Girth)
[1] 2.81694
```

summary() function:

```
> summary(my_data$Girth)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.30   11.05   12.90   13.25   15.25   20.60
```

```
> summary(my_data,digits=2)
     Girth           Height       Volume
 Min.   : 8.3   Min.   :63   Min.   :10
 1st Qu.:11.1   1st Qu.:72   1st Qu.:19
 Median :12.9   Median :76   Median :24
 Mean   :13.2   Mean   :76   Mean   :30
 3rd Qu.:15.2   3rd Qu.:80   3rd Qu.:37
 Max.   :20.6   Max.   :87   Max.   :77
```

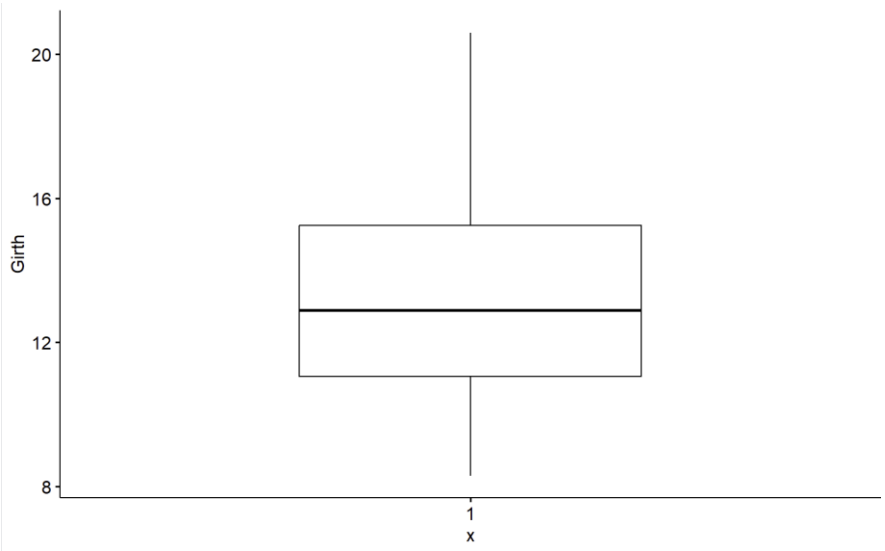
sapply() function

```
> sapply(my_data[,-5], mean)
   Girth   Height   Volume
13.24839 76.00000 30.17097
> sapply(my_data[,-5], quantile)
     Girth Height Volume
0%    8.30     63   10.2
25%  11.05     72   19.4
50%  12.90     76   24.2
75%  15.25     80   37.3
100% 20.60     87   77.0
```

Graphical display of distributions:

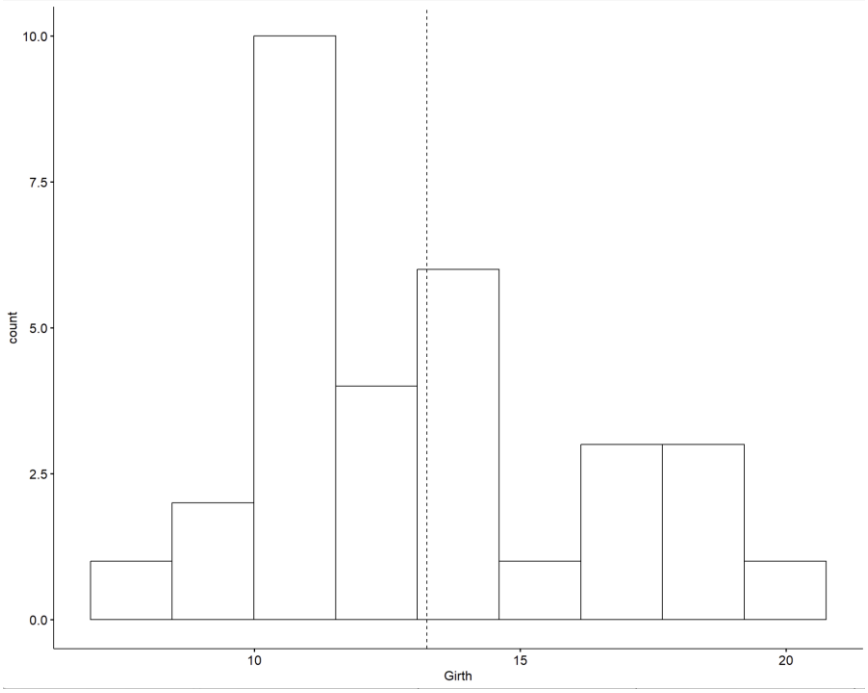
Box Plot:

```
> ggboxplot(my_data,y="Girth",width=0.5)
```



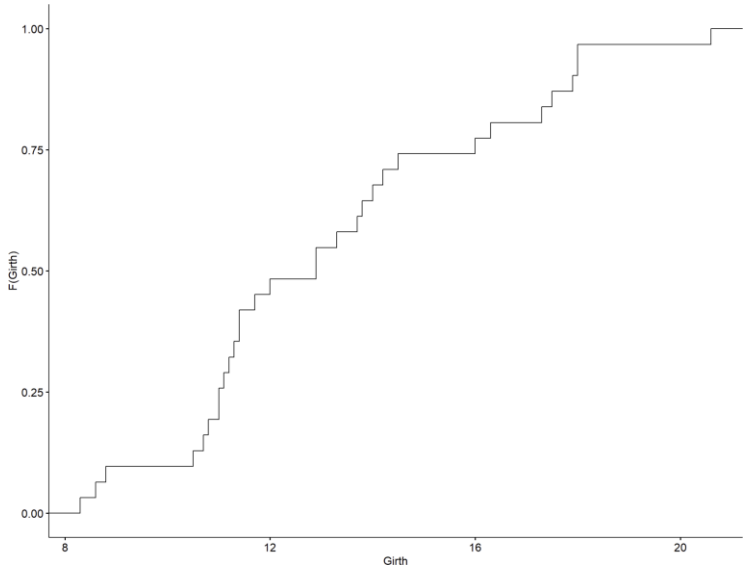
Histogram with mean line:

```
> gghistogram(my_data,x="Girth",bins=9,add="mean")
```



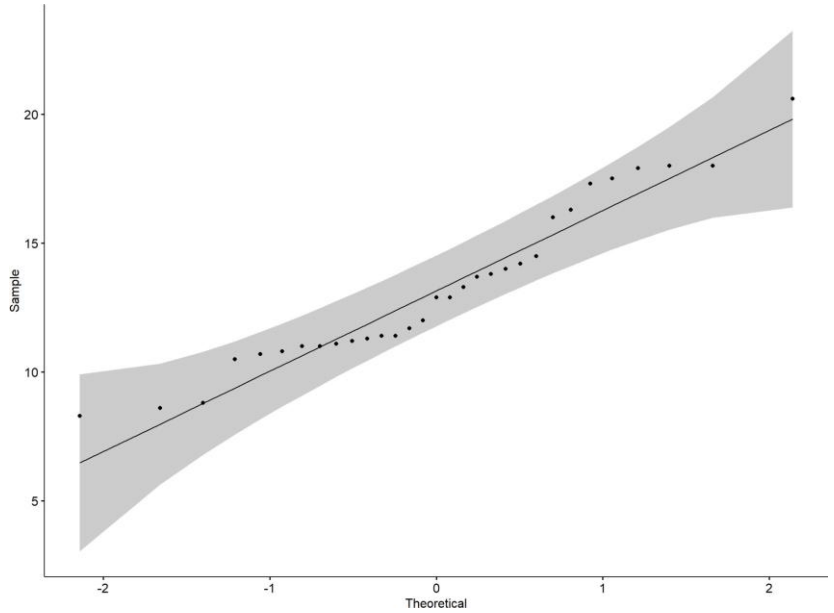
Empirical cumulative distribution function (ECDF):

```
> ggecdf(my_data, x="Girth")
```



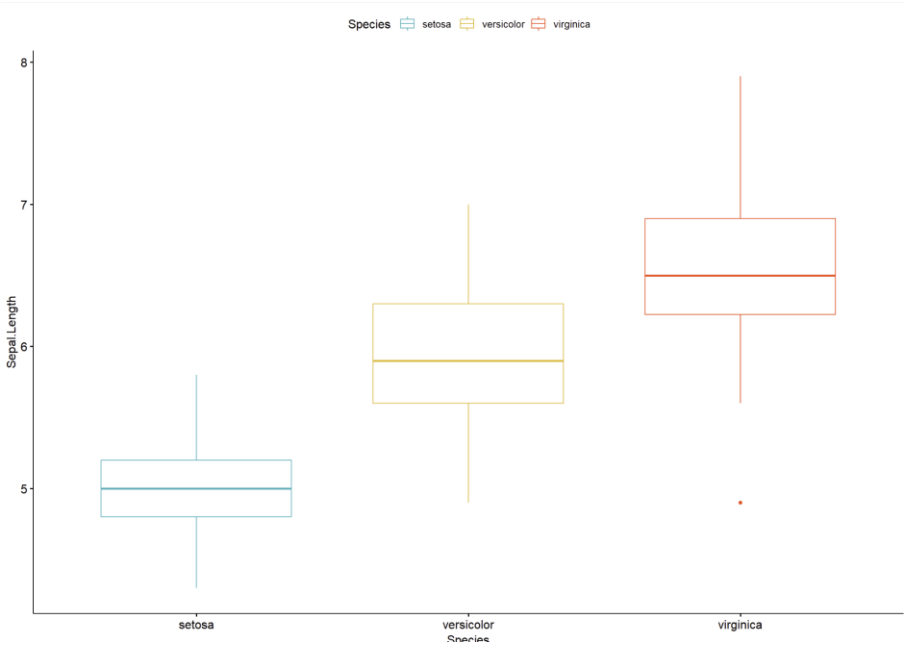
QQ plots are used to check whether the data is normally distributed.

```
> ggqqplot(my_data,x="Girth")
```

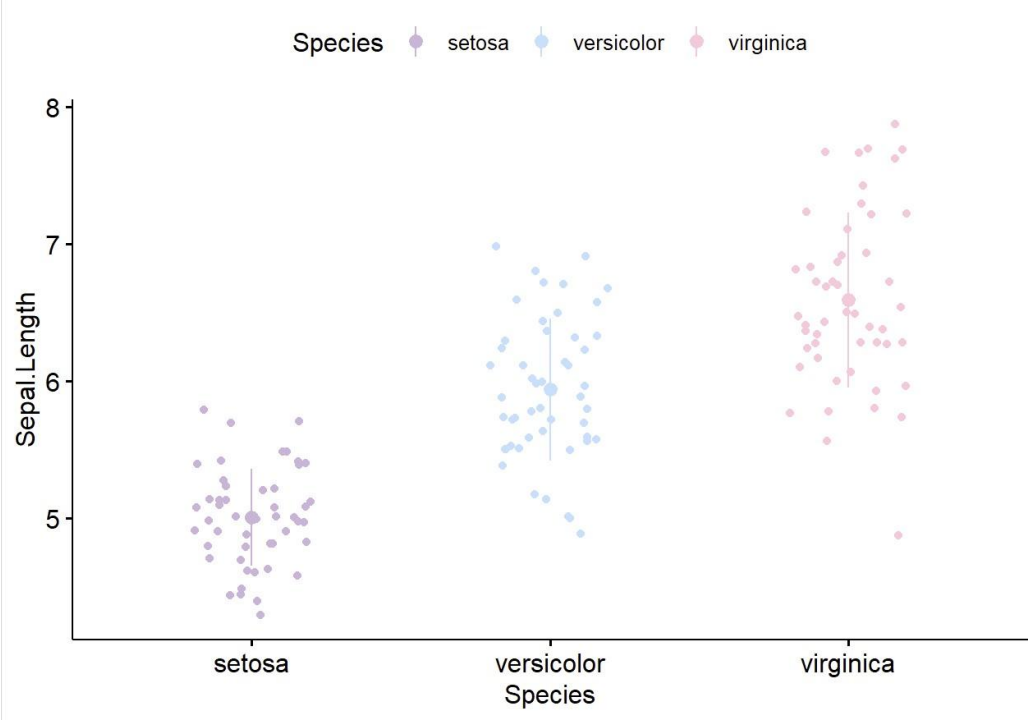


Descriptive statistics by groups

Graphics for grouped data:




```
> ggstripchart(my_data1, x="Species", y="Sepal.Length",
+              color = "Species",
+              palette = c("#cdb4db","#bde0fe","#ffc8dd"),
+              add="mean_sd")
```



Frequency tables

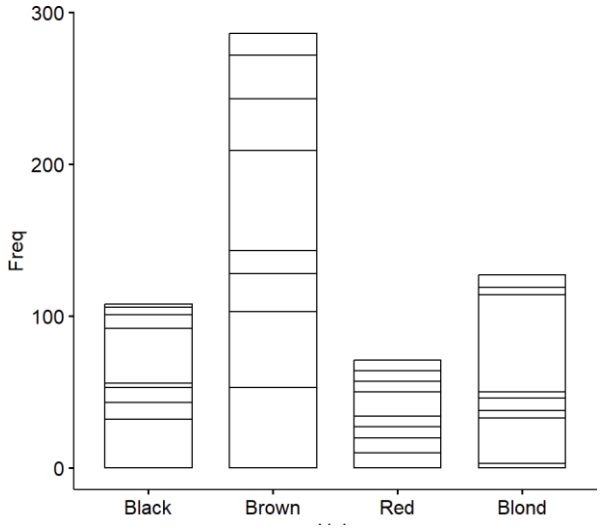
```
> df<-as.data.frame(HairEyeColor)
> hair_eye_col<-df[rep(row.names(df), df$Freq), 1:3]
> rownames(hair_eye_col)<-1:nrow(hair_eye_col)
> head(hair_eye_col)
   Hair   Eye  Sex
1 Black Brown Male
2 Black Brown Male
3 Black Brown Male
4 Black Brown Male
5 Black Brown Male
6 Black Brown Male
```

Table of counts:

```
> Hair<-hair_eye_col$Hair
> Eye<-hair_eye_col$Eye
> table(Hair)
Hair
Black Brown   Red Blond
  108   286    71   127
> table(Eye)
Eye
Brown  Blue Hazel Green
  220   215    93    64
```

Visualization:

```
> library(ggpubr)
> ggbarplot(df,x="Hair",y="Freq")
```



Two-way contingency table: Two categorical variables

```
> #two way contingency table
> hair_eye<-table(Hair,Eye)
> hair_eye
        Eye
Hair     Brown Blue Hazel Green
  Black     68   20    15     5
  Brown    119   84    54    29
  Red       26   17    14    14
  Blond      7   94    10    16
```

Multiway tables: More than two categorical variables:

```
> xtabs(~Hair + Eye + Sex, data=hair_eye_col)
, , Sex = Male

        Eye
Hair     Brown Blue Hazel Green
  Black     32   11    10     3
  Brown     53   50    25    15
  Red       10   10     7     7
  Blond      3   30     5     8

, , Sex = Female

        Eye
Hair     Brown Blue Hazel Green
  Black     36    9     5     2
  Brown     66   34    29    14
  Red       16    7     7     7
  Blond      4   64     5     8
```

```
> ftable(Sex+ Hair ~ Eye, data=hair_eye_col)
      Sex   Male                    Female
      Hair Black Brown Red Blond  Black Brown Red Blond
Eye
Brown          32    53  10     3     36    66  16     4
Blue           11    50  10    30      9    34   7    64
Hazel          10    25   7     5      5    29   7     5
Green           3    15   7     8      2    14   7     8
```

Compute table margins and relative frequency:

```
> margin.table(hair_eye,1)
Hair
Black Brown   Red Blond
  108   286    71   127
> margin.table(hair_eye,2)
Eye
Brown  Blue Hazel Green
  220   215    93    64
```

```
> prop.table(hair_eye, 1)
       Eye
Hair          Brown       Blue      Hazel      Green
  Black 0.62962963 0.18518519 0.13888889 0.04629630
  Brown 0.41608392 0.29370629 0.18881119 0.10139860
  Red   0.36619718 0.23943662 0.19718310 0.19718310
  Blond 0.05511811 0.74015748 0.07874016 0.12598425
> round(prop.table(hair_eye,1),2)*100
       Eye
Hair    Brown Blue Hazel Green
  Black    63   19    14     5
  Brown    42   29    19    10
  Red      37   24    20    20
  Blond     6   74     8    13
> round(prop.table(hair_eye,1),4)*100
       Eye
Hair    Brown  Blue Hazel Green
  Black 62.96 18.52 13.89  4.63
  Brown 41.61 29.37 18.88 10.14
  Red   36.62 23.94 19.72 19.72
  Blond  5.51 74.02  7.87 12.60
```

Post Lab questions

1. Critically assess the limitations of using only measures of central tendency in data analysis.
   Ans:
   Measures of central tendency, such as the mean, median, and mode, provide a summary of a dataset by identifying a central point. However, relying solely on these measures has several limitations:

- **Loss of Information**: Central tendency measures do not convey the distribution of data points. For example, two datasets can have the same mean but differ significantly in their spread or shape.
- **Sensitivity to Outliers**: The mean is particularly sensitive to extreme values, which can skew the results and provide a misleading representation of the data.
- **Non-Normal Distributions**: In skewed distributions, the mean may not accurately reflect the typical value, while the median may provide a better central point.
- **Lack of Context**: Central tendency measures do not provide insights into the variability or range of the data, which are crucial for understanding the data's context and implications.

2. Compare and contrast the different measures of variability, with the focus on when one measure might be more informative than the other.
   Ans:
   Measures of variability, such as range, variance, and standard deviation, provide insights into the spread of data. Here's a comparison:

- **Range**: The difference between the maximum and minimum values. It is simple to calculate but can be heavily influenced by outliers, providing a limited view of variability.
- **Variance**: The average of the squared differences from the mean. It accounts for all data points but is expressed in squared units, making it less interpretable in its raw form.
- **Standard Deviation**: The square root of variance, providing a measure of spread in the same units as the data. It is widely used and easily interpretable, making it a preferred choice in many analyses.

When to Use Each:

- Use **range** for a quick, rough estimate of variability, especially in small datasets.
- Use **variance** when you need to perform further statistical analysis, such as ANOVA, where squared differences are necessary.
- Use **standard deviation** for general descriptive statistics, as it provides a clear understanding of data spread in the same units as the original data.

3. Imagine you are presented with a dataset from a research study. Discuss how applying descriptive statistics techniques could aid in understanding the key features and trends in the data. Take any real life examples to aid your analysis.

Ans:

Descriptive statistics techniques, including measures of central tendency and variability, can significantly aid in understanding key features and trends in a dataset. For example, consider a dataset from a health study examining the blood pressure levels of a population.

- **Mean and Median**: Calculating the mean and median blood pressure can help identify the central tendency of the population's health status. If the mean is significantly higher than the median, it may indicate the presence of outliers (e.g., individuals with extremely high blood pressure).
- **Standard Deviation**: Assessing the standard deviation can reveal how much individual blood pressure readings vary from the mean. A high standard deviation would suggest a wide range of blood pressure levels, indicating a diverse population in terms of health.
- **Visualizations**: Creating histograms or box plots can visually represent the distribution of blood pressure levels, highlighting skewness, outliers, and the overall shape of the data.
- **Trends Over Time**: If the dataset includes longitudinal data, descriptive statistics can help identify trends, such as whether average blood pressure levels are increasing or decreasing over time, which could inform public health interventions.

Conclusion:

We learned how to work on datasets and data frames, apply various functions on it, summarize it and work on.