Chapter 4

Data Analytics 2: Exploratory Data Analysis (EDA)

Prepared by Dr Yuslina Zakaria

Exploratory Data Analysis (EDA) is a step to understand the underlying structure of data, to identify and explore patterns in the data, and hence guide us to answer our assumptions. Most of the steps in performing EDA are focusing on describing and visualising data distribution.

R Packages:

For this lesson, the following packages are used:

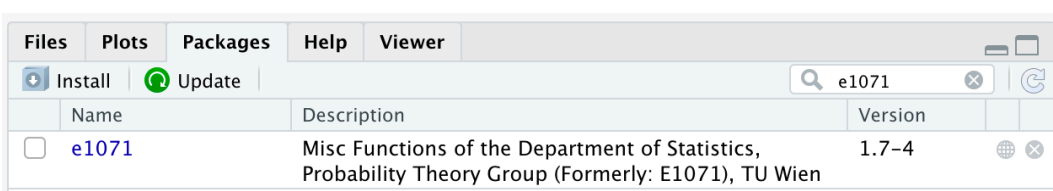
- `DescTools`
- `dplyr`
- `e1071`

Package Installation:

- install `DescTools` by typing `install.packages("DescTools")` in the R Console

Installation not required:

- `dplyr` package is a part of `tidyverse` core packages.
- `e1071` package is readily installed in the latest version of R. To verify the installation, search the package in the Packages tab of Files pane such as in the following figure.



If it is not found in the installed packages list, install the package by typing:

```
`install.packages ("e1071", dependencies = TRUE)`
```

Instructions:

- explains the steps for activity you need to follow.
- section contains practice questions for you to work on and submit as your lab report (Lab Report 2).

Data set:

The chronic kidney disease (CKD) data set is taken over 2-month period in a hospital in Tamil Nadu, India in 2015. The original data set contains 400 rows, corresponding to 400 patients with 25 features e.g. blood pressure, sugar, albumin and etc with the aim to identify if a patient has chronic kidney disease.

`ckd_clean.csv` contains a subset of 10 variables from the original data set which cases are complete (i.e. missing values removed).

The description of `ckd_clean.csv` is as follows:

- id : patient's identification no.
- age : age in years
- bp : blood pressure in mm/Hg (diastolic)
- rbc : red blood cells (in urine)
- bgr : blood glucose random in mgs/dl
- ane : anemia ("yes" and "no")
- htn : hypertension ("yes" and "no")
- dm : diabetes mellitus ("yes" and "no")
- cad : coronary artery disease ("yes" and "no")
- ckd : classification of patients ("yes" if has CKD and "no" without CKD)

Download the CKD data set from http://tiny.cc/phc410_da2 and place the files in the `input` folder, in your R project workspace.

Before we start, let us load all the required libraries.

Load all of the required libraries.

```
library(DescTools)
library(dplyr)
library(e1071)
```

Import the clean version of CKD dataset `ckd_clean.csv` from `input` folder as `ckd_clean` using `read.csv()`.

```
ckd_clean <- read.csv("input/ckd_clean.csv")
```

Transform `rbc`, `ane`, `htn`, `dm`, `cad`, `ckd` as factors. Then, observe the structure using `str()`.

```
#since only these columns contain characters
ckd_clean <- ckd_clean %>% mutate_if(is.character,as.factor)
str(ckd_clean)
## 'data.frame':    215 obs. of  10 variables:
##  $ id : int  2 3 4 8 9 11 14 20 22 27 ...
##  $ age: int  62 48 51 52 53 63 68 61 48 69 ...
##  $ bp : int  80 70 80 100 90 70 80 80 80 70 ...
##  $ rbc: Factor w/ 2 levels "abnormal","normal": 2 2 2 2 1 1 2 1 2 2 ...
##  $ bgr: int  423 117 106 138 70 380 157 173 95 264 ...
##  $ ane: Factor w/ 2 levels "no","yes": 2 2 1 2 2 1 1 2 2 1 ...
##  $ htn: Factor w/ 2 levels "no","yes": 1 2 1 2 2 2 2 2 2 2 ...
##  $ dm : Factor w/ 2 levels "no","yes": 2 1 1 2 2 2 2 2 1 2 ...
##  $ cad: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 2 2 1 2 ...
##  $ ckd: Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
```

4.1 Describing Distributions (Numerical Variables)

4.1.1 Measures of Central Tendency (MCT)

Measures of central tendency describes the central location of data distribution. Two commonly reported measures i.e. the median, the mean can be assessed using R built-in functions, whereas the third measure i.e. mode can be retrieved using several packages including `DescTools`.

4.1.1.1 The Median and the Mean

`median()` and `mean()` functions calculate both median and mean by assuming that there are no missing values. However, these functions return `NA` if the column contains missing values (i.e. `NA`). Thus, it is necessary to add the `na.rm=TRUE` option to remove missing values (i.e `NA`) from the calculations.

Calculate the median and the mean for `age` of `ckd_clean` data set.

```
#calculate median of patients' age in ckd_clean
median(ckd_clean$age,na.rm=TRUE)
## [1] 51

#calculate mean of patients' age in ckd_clean
mean(ckd_clean$age,na.rm=TRUE)
## [1] 49.63721
```

4.1.1.2 The Mode

To identify mode from the data distribution, external package is needed as no specific built-in function available for that purpose. In this practical, `Mode()` function from `DescTools` package is used to get the mode. Please note that the first letter of the function is a **capital letter** - to avoid confusion with R built-in function `mode()`.

Identify the mode from patients' age using `Mode()` function.

```
#get the mode
Mode(ckd_clean$age)
## [1] 55
```

```
## attr(,"freq")
## [1] 9
```

The results returned by `Mode()` indicate that the most frequently observed age of patients in the data set is 55 years old. It appears 9 times in a given data set.

`Mode()` also returns a subset of values if there are multiple values repeatedly occurring in the data set.

Getting the mode value if there are multiple values frequently appear in the data set.

```
#assign a numerical vector
score <- c(4,5,6,3,4,5,7,4,5,10,45,7,7,11,12)

Mode(score)
## [1] 4 5 7
## attr(,"freq")
## [1] 3
```

Three values of {4, 5, 7} appear 3 times in the data set.

Alternatively, `sort()` and `table()` built-in functions can also be used to identify the values of multiple modes.

Identify the mode from `score` vector using `sort()` and `table()` functions.

```
#get the mode using sort() and table()
sort(table(score), decreasing = TRUE)
## score
##  4  5  7  3  6 10 11 12 45
##  3  3  3  1  1  1  1  1  1
```

The first three values have the same frequencies (i.e. 3) in the data set. Thus, the mode of this data set is a subset of {4, 5, 7}.

4.1.2 Measures of Dispersion

For measuring the variation of data distribution, the same precautions of usage concerning missing frequencies apply in R. It will be necessary to add the `na.rm=TRUE` or `na.rm=T` option to notify R in the presence of missing data.

4.1.2.1 Range

Range is defined as the difference between the highest and the lowest values in a data set. A built-in function of `range()` returns the minimum and the maximum values from specific variables.

Use `range()` to get the minimum and the maximum values for `age`. Then, use `diff()` to get the value for range.

```
#get minimum and maximum values
range(ckd_clean$age,na.rm = TRUE)
## [1]  6 83

#get the difference (i.e. maximum - minimum) for range
```

```
diff(range(ckd_clean$age,na.rm = TRUE))
## [1] 77
```

You can also use `min()` and `max()` functions to get the minimum and maximum values, respectively, and hence, use both values to get the range.

Using `min()` and `max()` functions to get range. *(formula: range = max - min).*

```
max(ckd_clean$age) - min(ckd_clean$age)
```

```
## [1] 77
```

Alternatively, you can also use `Range()` function from `DescTools` package to get the value of range together with the lower and upper bounds in a one-liner code.

Using `Range()` from `DescTools` to get the range of `age` from `ckd_clean`.

```
#get the range using Range() from DescTools
Range(ckd_clean$age,na.rm = TRUE)
## [1] 77
## attr(,"bounds")
## [1]  6 83
```

4.1.2.2 Quantiles and Interquartile Range

For calculating quartiles, `quantile()` function from R `stats` built-in package is used. The followings are the differences between quartile, quantile and percentile.

- 0 quartile = 0 quantile = 0 percentile
- 1st quartile = 0.25 quantile = 25 percentile
- 2nd quartile = 0.5 quantile = 50 percentile
- 3rd quartile = 0.75 quantile = 75 percentile
- 4th quartile = 1 quantile = 100 percentile

Note that the median is corresponding to the 2nd quartile (50 percentile).

Use `quantile()` with additional arguments to get all quartiles for `age`.

```
quantile(ckd_clean$age,na.rm = T)
##   0%  25%  50%  75% 100%
##    6   39   51   61   83
```

An additional argument of `probs` is used if we want to get the specific quartiles from the data values.

To get the 1st and the 3rd quartiles, the equivalent values of quantiles (i.e. 0.25 and 0.75) are used in the argument.

```
quantile(ckd_clean$age, probs=c(0.25,0.75),na.rm = T)
## 25% 75%
##  39  61
```

`IQR()` function can be used to directly calculate the interquartile range of the variable.

Using `IQR()` to get the interquartile range of age.

```
IQR(ckd_clean$age,na.rm = T)
## [1] 22
```

4.1.2.3 Variance

`var()` function is performed to calculate the variance from the data set.

To get the variance of age.

```
var(ckd_clean$age,na.rm = T)
## [1] 252.307
```

4.1.2.4 Standard Deviation

The standard deviation of a data set is defined as the square root of the variance. Calculation of standard deviation in R is performed with the `sd()` function.

To get the standard deviation of age using the `var()` and the equivalent function by using `sd()`.

```
# using var
sqrt(var(ckd_clean$age,na.rm = T))
## [1] 15.88418

#using sd
sd(ckd_clean$age,na.rm = T)
## [1] 15.88418
```

4.1.3 Properties of Distribution

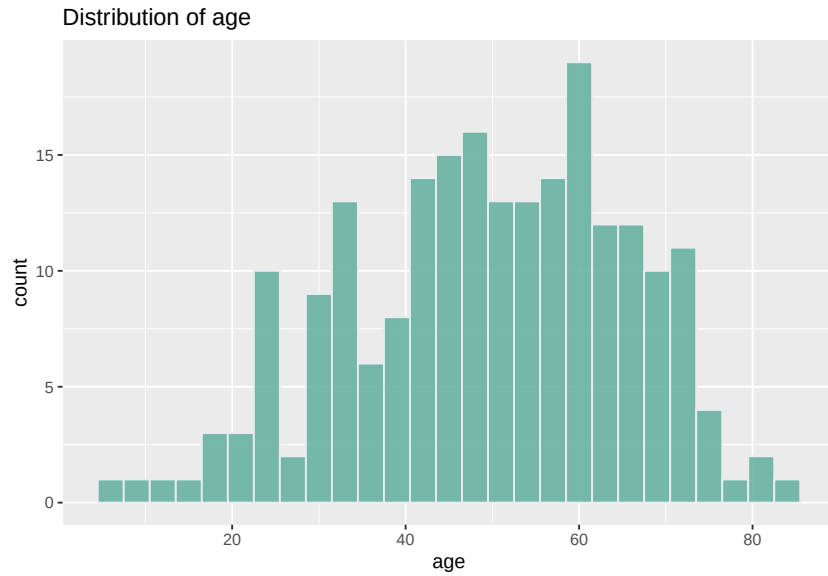
There are no base commands available in R to estimate the asymmetry or kurtosis of the data distribution. However, the `e1071` package provides `skewness()` and `kurtosis()` to measure the properies of distribution.

To assess the skewness and kurtosis of `age` using functions of `e1071` package.

```
#skewness of age
skewness(ckd_clean$age)
## [1] -0.3144032

#kurtosis of age
kurtosis(ckd_clean$age)
## [1] -0.4889174
```

Since, the skewness here is negative (i.e. -0.314), it implies that the distribution of age is negatively skewed or skewed to the left. The negative kurtosis (i.e.-0.489) indicates that the distribution of age has a platykurtic (or thin-tailed) distribution, which can also be seen in the following figure.



4.1.4 Five-values Summary

The `summary()` function summarises some of the measures previously described, all at once. For numerical variables, it calculates the minimum and maximum values, the median and the mean, and the first (or lower) and the third quartiles. Whereas, for categorical variables, it returns the count for each category.

Summarise all variables using `summary()`. Use the same function to provide a summary of a specific variable (i.e. `age`) in `ckd_clean`.

```
#summary of all variables in ckd_clean
summary(ckd_clean)
##        id             age              bp                rbc
##  Min.   :  2.0   Min.   : 6.00   Min.   : 50.00   abnormal: 35
##  1st Qu.:157.5   1st Qu.:39.00   1st Qu.: 70.00   normal  :180
##  Median :273.0   Median :51.00   Median : 80.00
##  Mean   :246.9   Mean   :49.64   Mean   : 75.16
##  3rd Qu.:341.5   3rd Qu.:61.00   3rd Qu.: 80.00
##  Max.   :399.0   Max.   :83.00   Max.   :110.00
##       bgr          ane       htn         dm       cad       ckd
##  Min.   : 22.0   no :189   no :160   no :163   no :202   no :130
##  1st Qu.: 99.0   yes: 26   yes: 55   yes: 52   yes: 13   yes: 85
##  Median :119.0
##  Mean   :139.8
##  3rd Qu.:140.0
##  Max.   :490.0

#summary of variable age only
summary(ckd_clean$age)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   39.00   51.00   49.64   61.00   83.00
```

4.1.5 Comparisons by groups

For comparing data distribution based on specific groups, we need to use `group_by()` and `summarise()` functions from `dplyr` package.

Calculate the median, mean and mode of diastolic blood pressure for both patients with and without CKD:

```
ckd_clean %>% group_by(ckd) %>% summarise("Median BP"=median(bp,na.rm=T),
                                          "Average BP"=mean(bp,na.rm=T),
                                          "Mode BP" = Mode(bp))
## `summarise()` ungrouping output (override with `.groups` argument)
## # A tibble: 2 x 4
##   ckd   `Median BP` `Average BP` `Mode BP`
##   <fct>       <dbl>        <dbl>     <int>
## 1 no             70         71.8        80
## 2 yes            80         80.2        70
```

From the results, we can see that the median and the mean of diastolic blood pressure for CKD patients are slightly higher than the patients without CKD. For CKD patients, blood pressure of 70 is the most frequent value occurred in the data set whereas most of the non-CKD patients had blood pressure of 80.

Identify the range, interquartile-range, variance and standard deviations of `age` of for patients with and without CKD.

```
ckd_clean %>% group_by(ckd) %>% summarise("Range"=Range(bp),
                                          "IQR"=IQR(bp,na.rm = T),
                                          "Variance"=var(bp,na.rm=T),
                                          "SD"=sd(bp,na.rm = T))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 5
##   ckd   Range   IQR Variance    SD
##   <fct> <int> <dbl>    <dbl> <dbl>
## 1 no       20    20     75.6  8.70
## 2 yes      60    20    183.  13.5
```

4.2 Describing Distributions (Categorical Variables)

4.2.1 Frequency Table

Regarding the categorical variables, either nominal or ordinal, a frequency table can be provided using the `table()` function. This function will not display missing counts as a separate category. Thus, it is important to add `useNA="always"` option in the presence of missing values. `useNA="ifany"` can also be used if we are unsure whether there are missing counts in the data set.

Display the counts of red blood cells type (i.e. `rbc`) in `ckd_clean`.

```
table(ckd_clean$rbc,useNA="ifany")
##
## abnormal   normal
##       35      180
```

4.2.2 Table of Proportions

In the following example, `prop.table()` function is used together with `table()` to create a relative frequencies table rather than a table of counts:

Create the proportion table of `rbc` in relative frequencies and percentage.

```
#create a proportion table in relative frequencies
tab <- table(ckd_clean$rbc)
prop.table(tab)
##
##  abnormal    normal
## 0.1627907 0.8372093

#create a proportion table in percentage
prop.table(tab) * 100
##
## abnormal   normal
## 16.27907 83.72093
```

We can only use mode to get the central location of categorical data. For this purpose, we can use `which.max()` function from R base and the `Mode()` function from `DescTools` package.

Get the mode of `rbc` using `which.max()` and `Mode()` functions.

```
#using the predefined tab from previous code (frequency table)
which.max(tab)
## normal
##      2

#using the Mode() from DescTools
Mode(ckd_clean$rbc)
## [1] normal
## attr(,"freq")
## [1] 180
## Levels: abnormal normal
```

The result returned by `which.max()` indicates that the most frequent type occur in `rbc` is `normal`. The `Mode()` provides more informative result by adding the frequency of mode (i.e. 180 counts).

4.3 Visualising Distributions

In this section, you will learn about basic plots in R to represent your data distribution graphically.

4.3.1 Stem and Leaf Plot

The `stem()` function creates a stem and leaf plot to represent the range of numeric categories (on the left) and the associated frequency (on the right).

Create a basic stem and leaf plot using the `stem()` of the age variable:

```
stem(ckd_clean$age)
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   0 | 68
##   1 | 2
##   1 | 5779
##   2 | 0123333444
##   2 | 55588999
##   3 | 000000223333344444
##   3 | 55567788999
##   4 | 00011112222233333444
##   4 | 555555666666777777888888899
##   5 | 00001111222223344
##   5 | 55555555566666677778888999999999
##   6 | 00000000111222233334444
##   6 | 55555566677788999
##   7 | 00000111123333334
##   7 | 5669
##   8 | 023
```

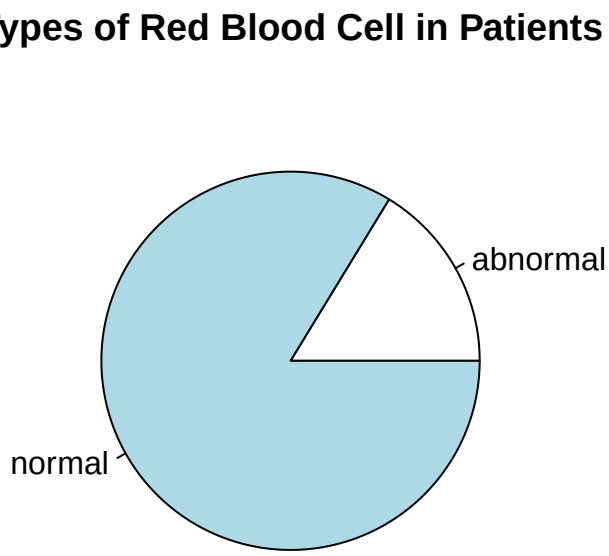
The stem and leaf plot is useful because the original values are shown and it can be used to quickly assess the data distribution. However, the `stem()` function cannot properly represent a large sample, and thus in such cases, a histogram is more useful.

4.3.2 Pie Charts

We can display the proportions of categorical data via pie charts by using the `pie()` function.

Create a pie chart using the `rbc` variable.

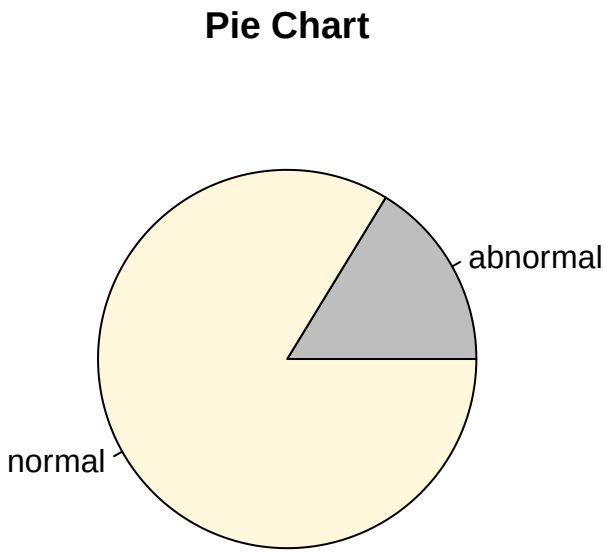
```
pie(prop.table(table(ckd_clean$rbc)),main="Types of Red Blood Cell in Patients")
```



Similarly, we can also create a pie chart using `group_by()` function from `dplyr` package as follows.

Create a pie chart using the `rbc` variable using `group_by()`.

```
#create a new object subs_rbc to contain frequency of rbc types
subs_rbc <- ckd_clean %>% group_by(rbc) %>% count()
pie(subs_rbc$n,subs_rbc$rbc,col= c("gray", "cornsilk"), main = "Pie Chart")
```



4.3.3 Histograms

The`hist()` function is used to create a histogram of a single variable. It has a variety of additional options to control the bin width, labels, colors and etc including the followings:

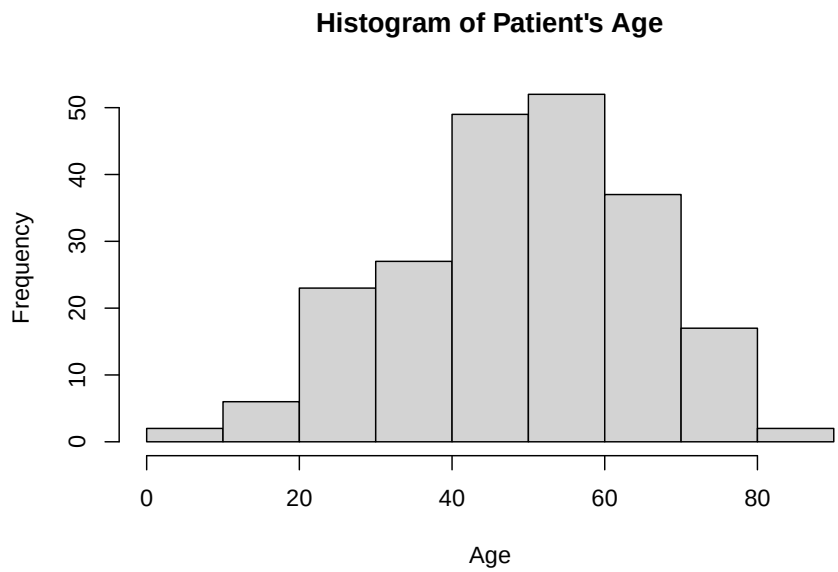
- main : set the main title for the histogram
- xlab : set the title for the x-axis.
- ylab : set the title for the y-axis.
- xlim : set the range for the x-axis i.e. `c(start, end)`
- ylim : set the range for the y-axis i.e. `c(start, end)`
- col : set the colour of the bars. use `colors()` or `colours()` to see available colours.

(*Type* ***`help(hist)`*** *in the console to know more about the options.*)

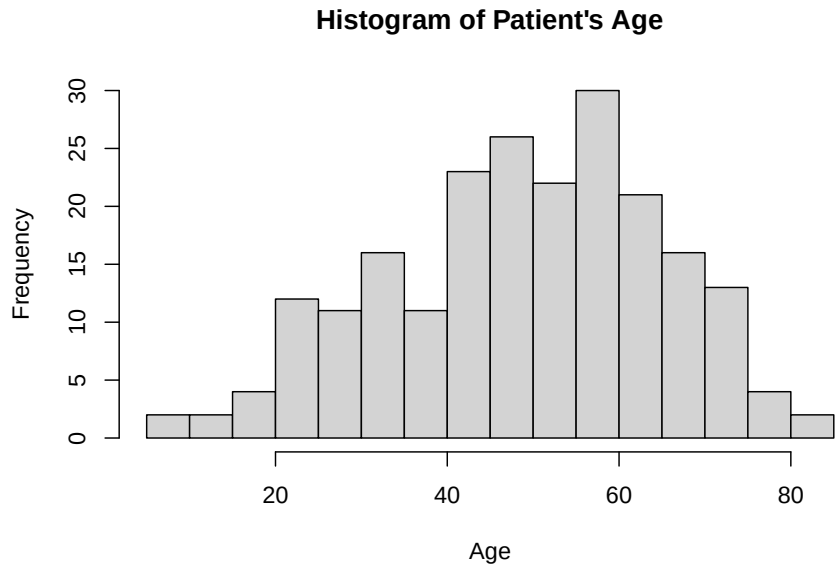
Another instruction worth to mention here is `nclass`.If we want to create a histogram where our observations are grouped, e.g. in 5 classes, then this can be made through the command `nclass=5`.

Create a histogram using the `hist()` of the `age` variable. Set the main and x-axis titles. Then set the breaks by no. of groups of 20.

```
#default histogram breaks
hist(ckd_clean$age,main="Histogram of Patient's Age", xlab="Age")
```



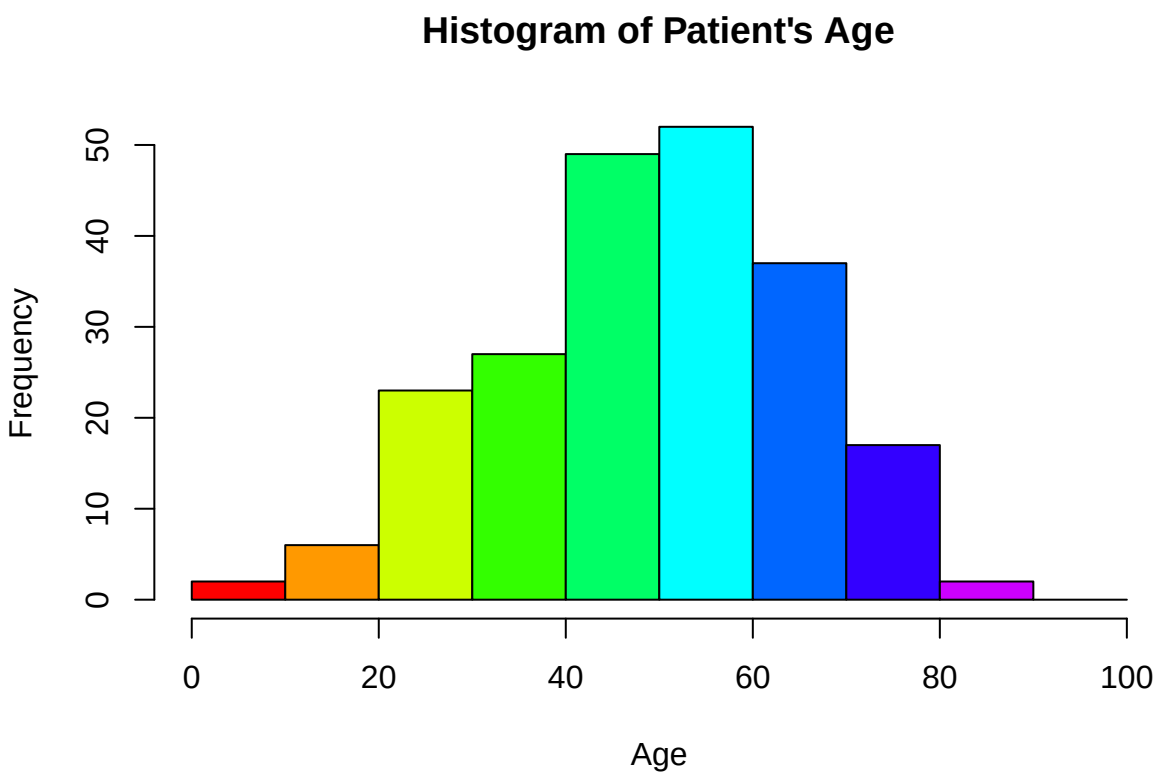

```
#histogram breaks by no of groups
hist(ckd_clean$age,main="Histogram of Patient's Age", xlab="Age",nclass=20)
```



We can also define the histogram breaks using our defined range of classes through a `seq()` function.

Create a histogram using the `hist()` of the `age` variable using defined breaks. `col` command is added to change the colour.

```
hist(ckd_clean$age,main="Histogram of Patient's Age", xlab="Age",
     breaks = seq(from=0,to=100,by=10),col=rainbow(10))
```



For more info about using colour in R and how to use them, please refer (http://www.sthda.com/english/wiki/colors-in-r).

4.3.4 Bar Charts

A histogram, which has been explained earlier in previous section, is a form of bar chart which represents the frequencies of data values. A bar chart, in contrast, is useful to represent and compare data that is classified into discrete categories (i.e. nominal or ordinal).

We can use the `barplot()` function to create bar charts, both vertically and horizontally. Two common types of bar charts are:

- Single Category (horizontally and vertically)
- Multiple Category (stacked and grouped)

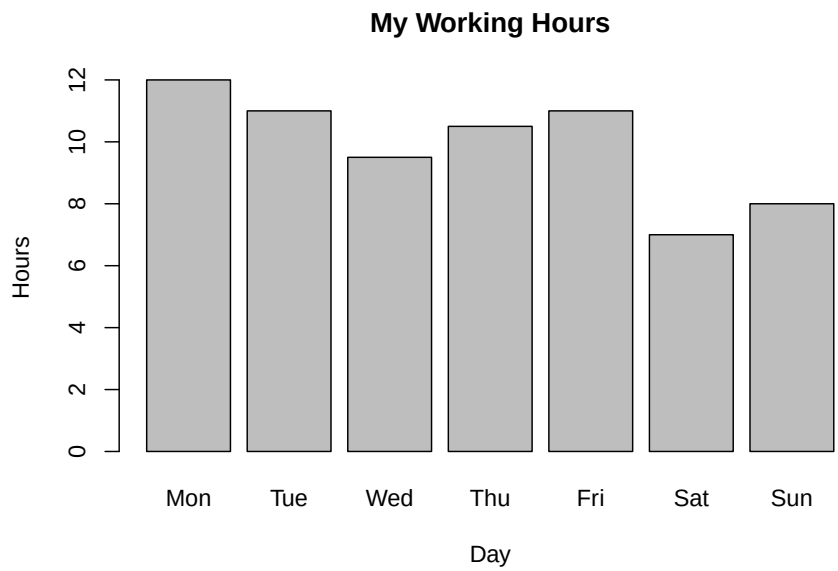
4.3.4.1 Single Category Bar Charts (Horizontal and Vertical)

The simplest plot can be made from a single numeric variable, as follows.

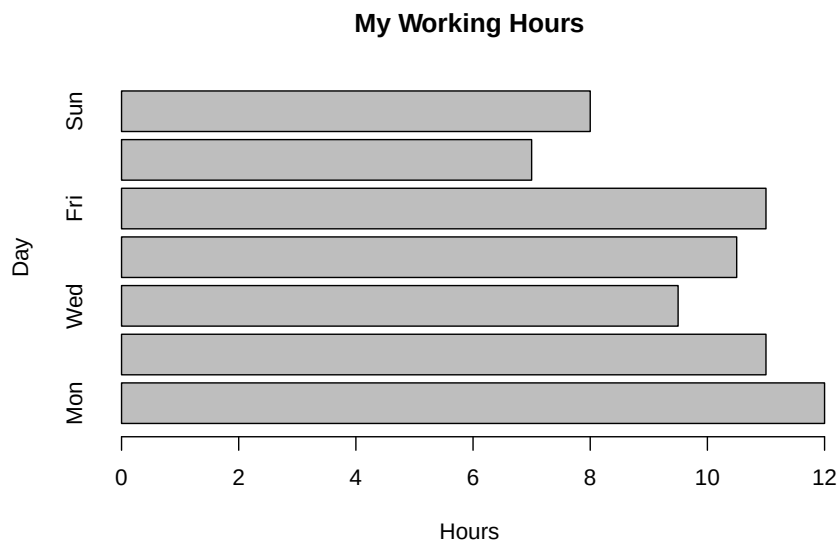
Create vertical and horizontal bar charts for `workhours`.

```
#create workhours object containing a vector of numerical values
workhours <- c(12,11,9.5,10.5,11,7,8)
names(workhours) <- c("Mon","Tue","Wed","Thu","Fri","Sat","Sun")

#create a bar chart from age of CKD patients only (vertical)
barplot(workhours,main="My Working Hours",xlab="Day",ylab="Hours")
```




```
#create a bar chart from age of CKD patients only (horizontal)
barplot(workhours,main="My Working Hours",xlab="Hours",ylab="Day",horiz = T)
```



4.3.4.2 Multiple Category Bar Charts (Stacked and Grouped)

This type of bar chart is used to represent different bars for different sub-groups of the main categories. We can use different colours and shades to differentiate specific bars from the other categories. Note that to use `barplot()` for a grouped data, the data frame needs to be firstly transformed into a contingency matrix. A contingency matrix represents two categorical variables and their frequency distributions.

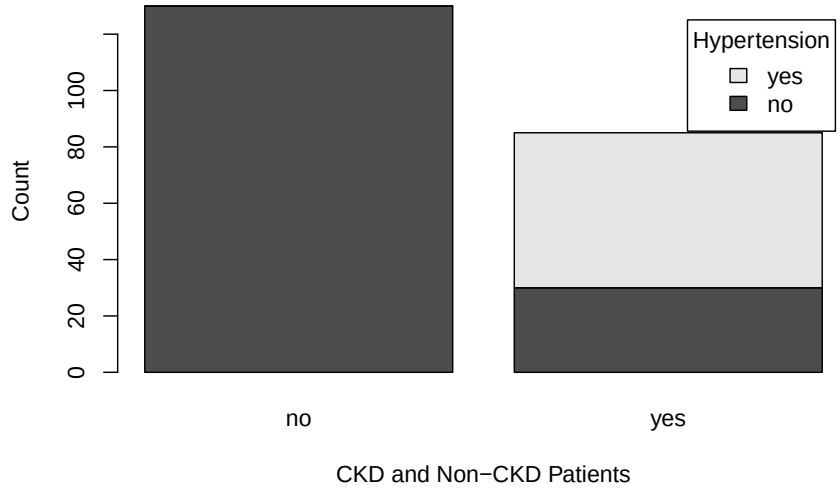
Suppose we want to display the frequency distributions of patients who have hypertension (i.e. `htn`) and diabetes mellitus (i.e. `dm`).

Create a contingency matrix of `htn` and `ckd` from `ckd_clean`. Assign the matrix to `htn_ckd`.

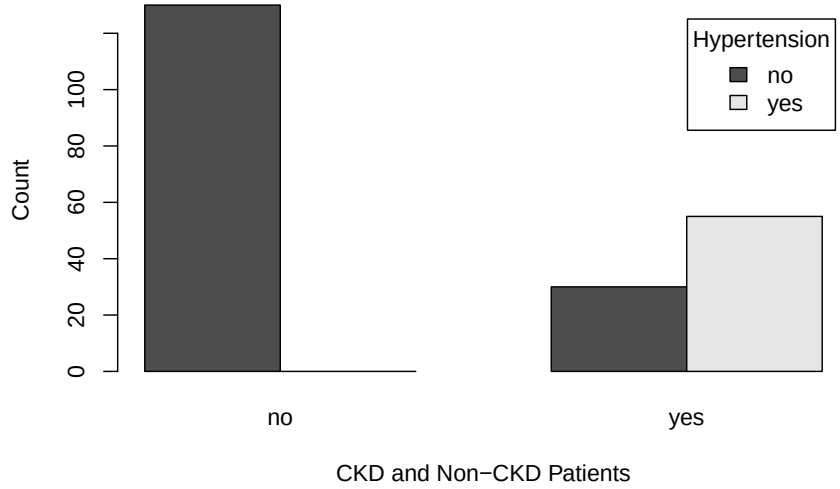
```
#a contingency matrix of htn and ckd
#htn will be set as rows, htn as columns
htn_ckd <- table(ckd_clean$htn,ckd_clean$ckd)
```

Generate stacked and grouped bar charts from `htn_ckd`. Set the space between columns as 0.2.

```
#stacked bar chart based on hypertension status
barplot(htn_ckd,xlab ="CKD and Non-CKD Patients", ylab ="Count",
        args.legend=list(title="Hypertension"), legend=T)
```




```
#grouped bar chart based on hypertension status
barplot(htn_ckd,xlab ="CKD and Non-CKD Patients", ylab ="Count",
        args.legend=list(title="Hypertension"), legend=T,beside = T)
```



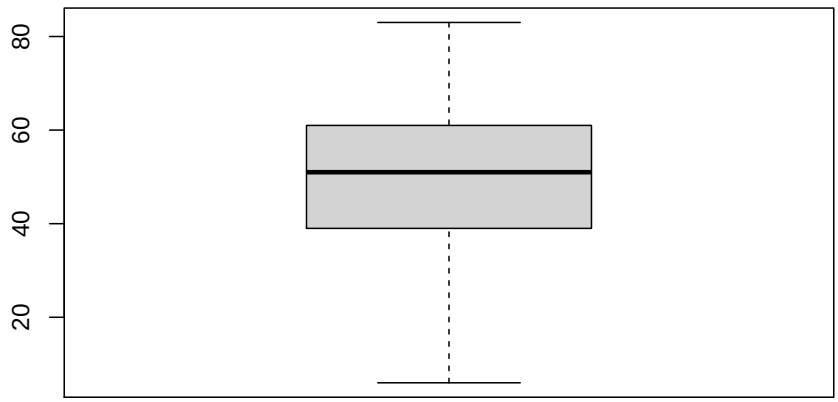
Based on the generated plots, it can be seen that the grouped bar chart provides better representation of the data based on gender. From the results, we can see that most of the CKD patients are having hypertension.

4.3.5 Boxplots

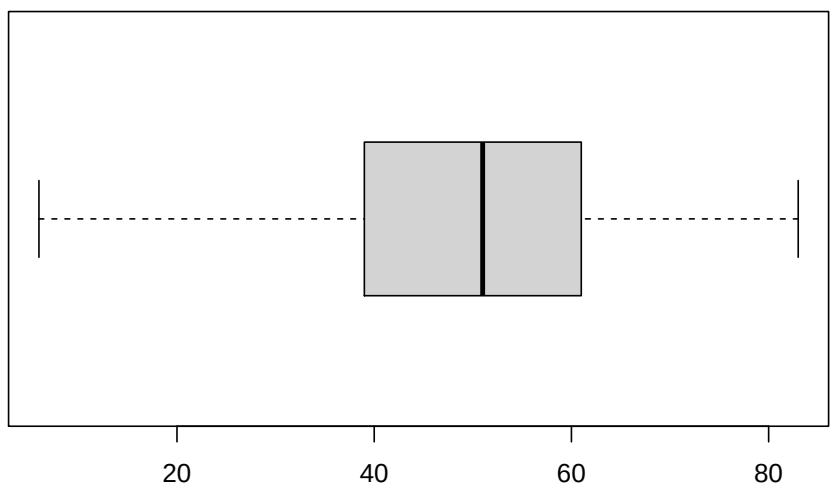
The boxplot (or known as box-and-whiskers plot) is a standard plot of displaying distribution based on the five values of the first quartile (Q1), second quartile (Q2) or the median and third quartile (Q3). The whiskers, on the other hand, denote the range from the smallest value (minimum) to the highest value (maximum) found in the observations. The boxplot also helps you to detect the outliers, and get the information about the symmetry and the skewness of your data.

Create a typical boxplot for `age` from `ckd_clean`.

```
#vertical boxplot
boxplot(ckd_clean$age)
```




```
#horizontal boxplot
boxplot(ckd_clean$age,horizontal = T)
```



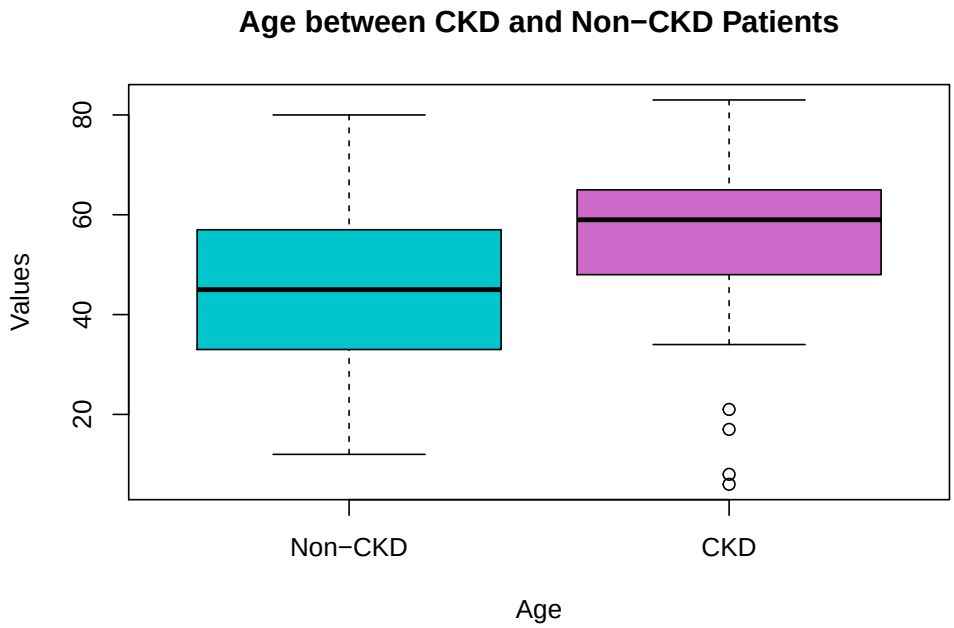
To customize the appearance of the boxplot, you may use the additional instructions in the function arguments (use `help(boxplot)` or `?boxplot` to know more).

We usually use a boxplot when we have a continuous variable which is associated with at least a categorical variable. The distributions of the continuous variable can be compared in the perspective of the categorical variable.

In the following example, we will compare the characteristics of age between CKD and non-CKD patients using the side-by-side boxplot.

Compare `age` between CKD and non-CKD patients.

```
boxplot(age~ckd, ckd_clean,xlab = 'Age', names=c("Non-CKD","CKD"),
        ylab = 'Values',col=c("turquoise3","orchid3"),
        main="Age between CKD and Non-CKD Patients")
```



From the resulting boxplot, we can see that CKD patients have a smaller range of `age` compared to the non-CKD patients. The age distribution of CKD patients is skewed to the left (skewness of -1.087) and it is approximately symmetric for non-CKD patients (skewness of 0.09). The boxplot of CKD patients' age appears higher than the non-CKD which indicates that most CKD patients are older patients.

Notice that the age values for CKD patients contain outliers (not found in Non-CKD). From the boxplot, we can see that the values of outliers are smaller than 30.

Using the conditional statement of age for CKD patients, we can get the values of outliers in the following examples.

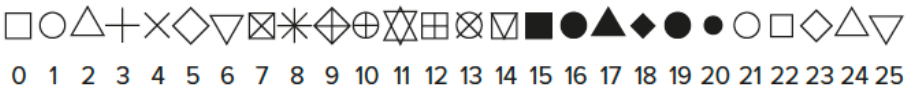
Get outliers in `age` from CKD patients.

```
ckd_clean %>% filter(ckd=="yes" & age < 30) %>% select(age)
##   age
## 1   8
## 2  17
## 3  21
## 4   6
```

4.3.6 Scatter Plots

In R, basic `plot()` command can be used to produce `xy` scatter plots from two numeric data values i.e. `x` and `y`. The scatter plot is usually used to represent the relationship between two variables, especially when we want to identify the correlation between variables.

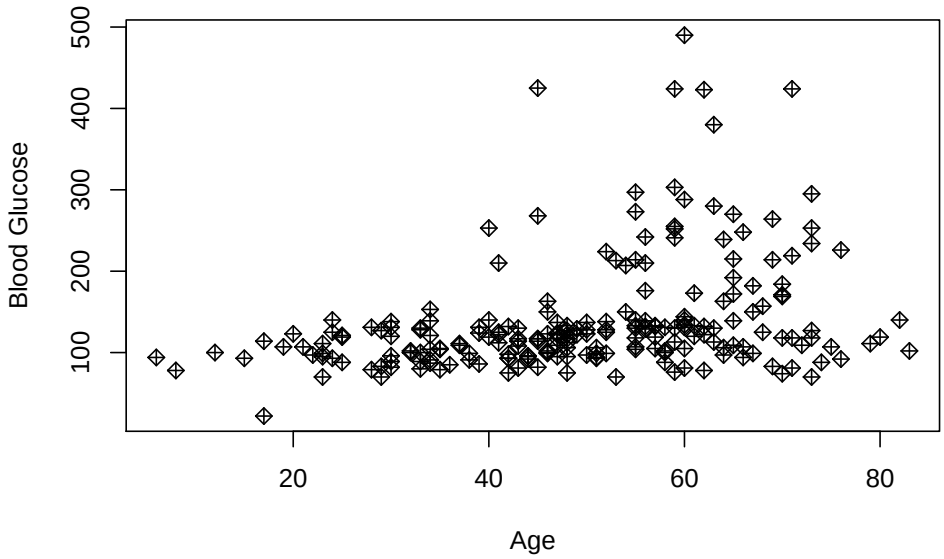
As additional instructions, you may want to change the look of your scatter plot using `pch` option of your preferred plot symbols, as follows:



In the following example, you will create a scatter plot between numeric values in `ckd_clean`.

Create a scatter plot between age and blood glucose.

```
#create a scatter plot between age and blood glucose. Set the symbol using cex=9
plot(ckd_clean$age, ckd_clean$bgr,xlab="Age",ylab="Blood Glucose",pch=9)
```

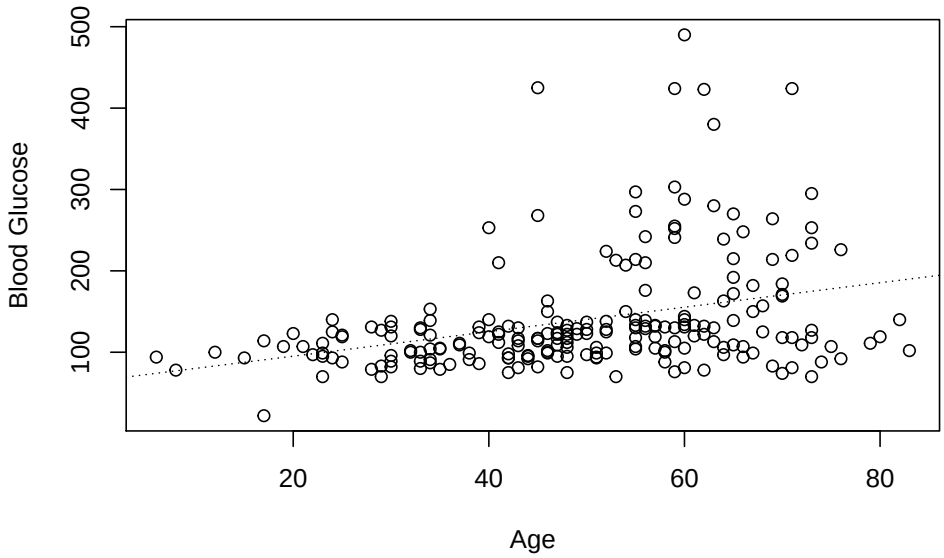


You can also draw a fitted line on your scatter plot using `abline()` and specify the line type using `lty` option based on the following:

VALUE	LABEL	RESULT
0	`blank`	Blank
1	`solid`	Solid (default)
2	`dashed`	Dashed
3	`dotted`	Dotted
4	`dotdash`	Dot-Dash
5	`longdash`	Long dash
6	`twodash`	Two dash

Create a scatter plot between age and blood glucose. Draw a fitted line using dotted line.

```
#create a scatter plot between age and blood glucose. Set the symbol using cex=9
plot(ckd_clean$age, ckd_clean$bgr,xlab="Age",ylab="Blood Glucose")
abline(lm(ckd_clean$bgr ~ ckd_clean$age),lty=3)
```



4.4 Advance Graphics using ggplot2

To create more sophisticated and beautiful outputs, you may want to use functions provided by `ggplot2` package. The package is effective at working with tidy data and `dplyr` package. Generating graph using `ggplot2` package is not covered in this lesson. However, there are immense online resources that can help you with `ggplot` if you are interested. To get familiar with ggplot, you may want to start with this link: https://www.r-graph-gallery.com/.

4.5 Practice

Data set:

The World Health Organisation (WHO) estimates that approximately one million people die from suicide each year. This practice will use WHO world suicide data from 1985 to 2016, occurred in 101 countries. The following features are present in the data:

- country : name of the countries
- year : year of the suicides recorded
- sex : gender
- age : age groups
- suicide_100_population : number of suicides per 100k population
- gdp : gdp per capita
- generation : generation for each group

Put your answers in the Rmarkdown file `da2_practice.Rmd` which can be found in http://tiny.cc/phc410_da2 and knit to PDF to generate a PDF report.

Questions:

1. Import a csv file `WHO_suicide.csv` from `input` folder to an object `subs_suicide`.
2. Convert `country`, `sex`, `age`, and `generation` as factor. *Hint: use as.factor()*
3. Create a scatter plot between `gdp` and `suicide_100_population`. Interpret the relationship between the two variables from the graph.
4. Create both stacked and grouped bar charts that represent the age and the type of generation. Display the legend and set the title of legend as `Age group`. Use `rainbow(6)` as the colours and set a smaller size of x axis labels with `cex.names=0.7`.
5. Based on your results in question 4, which generation has the highest number of age groups who had attempted suicide?