DATA EXPLORATION

Working with datasets and producing descriptive statistics

A few basic things first…

Working directory

□ The working directory is the directory in which you are working in.

□ To check your current working directory, use the function `getwd()`.

□ To change your working directory, use the function `setwd(directory_path)`.

□ Tips: running "`setwd(choose.dir())`" allows you to select the directory interactively.

Using packages

- Packages are a collection of codes and functions written by a third party and is not included in the base R.

- To install and download a package, use the function `install.packages("package_name")`.

- Then to load the package, use the function `library(package_name)`.

Using functions

- For built-in functions, using `help(function_name)` will give the documentation of the function which includes:
  - Description of the function
  - Arguments and its default value
  - Values the function returns

- The argument of a function takes the default value if it is not specified.

- When running a function, you can specify the argument by writing `fn(argument_name=value)`, or just setting `f(value)`.

- If you do not specify the argument name, then R will specify the argument by position.

Using functions

- Suppose we have:

```
fn_name(arg1 = value1, arg2 = value2)
```

  - The name of the function is `fn_name`.
  - Two arguments: `arg1` and `arg2`.
  - Default value for `arg1` is `value1`, default value for `arg2` is `value2`.
  - Running `fn_name()` is equivalent to running `fn_name(arg1=value1, arg2=value2)`.
  - Running `fn_name(a)` is equivalent to `fn_name(arg1=a, arg2=value2)`.
  - Running `fn_name(a, b)` is equivalent to `fn_name(arg1=a, arg2=b)`.
  - Running `fn_name(arg2=a, arg1=b)` is equivalent to `fn_name(arg1=b, arg2=a)` or `fn_name(b, a)`.

Datasets in R

Data frames

- A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

- What this basically mean:
  - Data frames are like a combination of matrix and list (which we will explore later), each column corresponds to a variable and each row corresponds to a sample/observation.
  - One sample or observations has multiple variables.
  - E.g.:
    - Row corresponds to each student.
    - Column 1 is students' assignment marks.
    - Column 2 is students' test marks.
  - Like matrix: has columns and rows.
    - But unlike matrix, each columns can be of different class.
  - Like list: has variable names and can call them.

Creating data frame

- Use `data.frame()` function.
- Example:

```
> x <- 1:10
> y <- x^2
> dat <- data.frame(col1=x, col2=y)
> print(dat)
   col1 col2
1     1    1
2     2    4
3     3    9
4     4   16
...
```

- You can also convert a matrix to a data frame using `as.data.frame()`.

Pulling out contents of a data frame

- We use dollar sign `$` to extract the variable nested inside of a data frame.
- Example:

```
> x <- 1:10
> y <- x^2
> dat <- data.frame(col1=x, col2=y)
> print(dat$col1)
 [1]  1  2  3  4  5  6  7  8  9 10
> print(dat$col2)
 [1]   1   4   9  16  25  36  49  64  81 100
```

Selecting a subset of observations with some criteria

- If you want to select/print observations with some conditions/criteria, you can use the `[]` notation like we did for matrix.

- Alternatively, you can use the `subset()` function.

```
subset(data_frame_name, condition)
```

Selecting a subset of observations with some criteria

- Example:

```
> dat[1:5, ]
 col1 col2
1    1    1
2    2    4
3    3    9
4    4   16
5    5   25
> dat[,1]
[1]  1  2  3  4  5  6  7  8  9 10
```

Selecting a subset of observations with some criteria

- Example:

```
> dat[dat$col2<=25, ]
  col1 col2
1    1    1
2    2    4
3    3    9
4    4   16
5    5   25
> subset(dat,col2<=20)
  col1 col2
1    1    1
2    2    4
3    3    9
4    4   16
```

Some functions for data frames

str() function

- The function `str()` can be used to quickly look into the structure of an object, including a data frame.

- It will give the number of observation as well as variable's names and classes

- Example:

```
> str(dat)
'data.frame':      10 obs. of  2 variables:
 $ col1: int  1 2 3 4 5 6 7 8 9 10
 $ col2: num  1 4 9 16 25 36 49 64 81 100
```

names() function

- The function `names()` can be used to list down all variable names (or column names) in a data frame.

- It can also be used to modify the column names

- Example:

```
> names(dat)
[1] "col1" "col2"
> names(dat) <- c("x","y") #change the var names
> names(dat)
[1] "x" "y"
```

Row and column names

- You can also use `colnames()` to get the column names, and `rownames()` to get the row names of a data frame/matrix.

- Like the `names()` function, you can also use these to modify the column or row names.

Row and column names

- Example:

```
> colnames(dat)
[1] "x" "y"
> rownames(dat)
[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
> rownames(dat)[1] <- "No.1"
> dat
        x   y
No.1    1   1
2       2   4
3       3   9
4       4  16
5       5  25
6       6  36
7       7  49
8       8  64
9       9  81
10     10 100
> rownames(dat)[1] <- "1"
```

View() and fix() functions

- You can use `View()` function (capital V) to view the whole data frame on a separate window. But you can't edit it.
- If you want to edit it interactively, you can use `fix()` function.
- Example:

```
> View(dat)
> fix(dat)
```

head() and tail() functions

- Sometimes you don't want to view all the dataset, but only a few just to see what it looks like.
- You can use the `head()` or `tail()` functions to print out the first or the last n rows in the dataset.
- The default value for n is 6, but you can modify it by specifying `n=value` in the argument of the functions.

head() and tail() functions

- Example:

```
> head(dat)
   x   y
1  1   1
2  2   4
3  3   9
4  4  16
5  5  25
6  6  36
> tail(dat)
       x    y
5      5   25
6      6   36
7      7   49
8      8   64
9      9   81
10  10  100
```

More functions for data frames

- `data()`:
  - R has built-in datasets. To load these datasets, use `data(dataset_name)`
  - E.g.: `data(mtcars)`

- `na.omit()`:
  - Remove rows with missing values (coded as NA in the dataset).
  - Useful for analysis if you want to remove them.

More functions for data frames

- `merge()`:
  - Merge two data frames by common columns or row names.
  - In some ways similar to `rbind()` and `cbind()` but better as it merge based on column or row names.

- `lapply()`/`sapply`:
  - Apply function to each variables in the data frame.
  - `lapply()` returns a list. `sapply()` returns a vector.
  - E.g.:
    - `lapply(dat, FUN=mean)`
    - `sapply(dat, FUN=mean)`

More functions for data frames

- which():
  - Used to determine which rows satisfy a condition.
  - E.g.:

```
> which(dat$y > 60)
[1]  8  9 10
> which(dat$x < 5)
[1] 1 2 3 4
```

- attach():
  - Attach a data frame to R search path.
  - When a data frame is attached, there is no need to use $ to call the variable.
  - The function detach() must be used to detach the data frame.

Importing and exporting data

Importing data into R

- We can type them down manually and use the `data.frame()`.
- For large amount of data, this is inefficient.
- It might be better to have the data in a separate txt, csv or excel file.

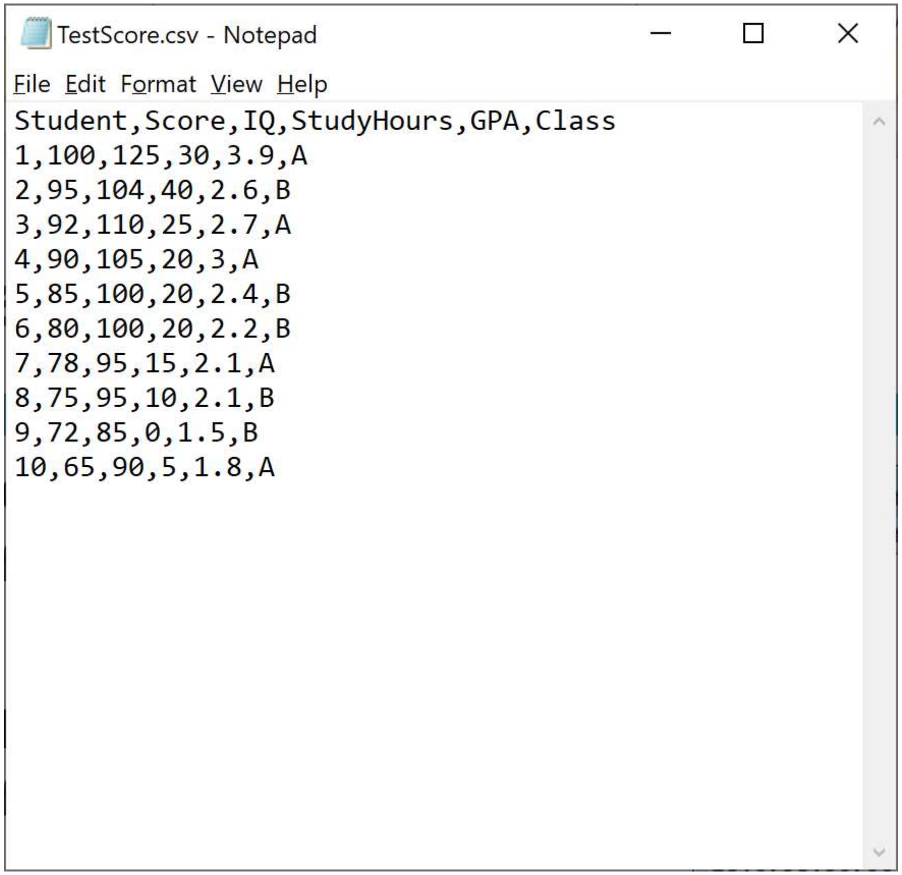
Reading from CSV or TXT file

- Use `read.table(...)` function.

- Argument: `read.table(file_location, header, sep)`
  - `file_location`: The location of the txt or csv. File location uses "/" , not "\".
  - `header`: TRUE if there is a header (column name) in the file. Otherwise, FALSE.
  - `sep`: The separator between items in the file.

Reading from CSV or TXT file

- Open the file with notepad first to see the structure.
- In this case, we have header for each column.
  - `header=TRUE`
- And columns are separated with a comma “,”.
  - `sep=","`



```
Student,Score,IQ,StudyHours,GPA,Class
1,100,125,30,3.9,A
2,95,104,40,2.6,B
3,92,110,25,2.7,A
4,90,105,20,3,A
5,85,100,20,2.4,B
6,80,100,20,2.2,B
7,78,95,15,2.1,A
8,75,95,10,2.1,B
9,72,85,0,1.5,B
10,65,90,5,1.8,A
```

Reading from CSV or TXT file

- Example:

```
> TestScore <- read.table("C:/Users/Hilmi
   Majid/OneDrive/PnP/2021 Sem
   1/STQD6214/datasets/TestScore.csv",
   header=TRUE, sep=",")
> head(TestScore)
  Student Score  IQ StudyHours GPA Class
1       1   100 125         30 3.9     A
2       2    95 104         40 2.6     B
3       3    92 110         25 2.7     A
4       4    90 105         20 3.0     A
5       5    85 100         20 2.4     B
6       6    80 100         20 2.2     B
> names(TestScore)
[1] "Student"    "Score"      "IQ"
[4] "StudyHours" "GPA"        "Class"
```

Tips and tricks

- You can choose the file interactively using `file.choose()`.
  - Eg: `TestScore <- read.table(file.choose(), header=TRUE, sep=",")`
- For CSV file, an easier function to use is `read.csv()` function.
  - Eg: `TestScore <- read.csv(file.choose())`
- If your CSV file is large, you can use `read_csv()` function in `readr` package for faster loading time.
- Lastly, you can use RStudio to import data interactively.

Reading from MS Excel file

- Reading from Excel file (xls or xlsx) requires `readxl` package:

- Example:

```
> install.packages("readxl")
> library(readxl)
> oxygen <- read_excel("C:/Users/Hilmi
Majid/OneDrive/PnP/1920 Sem 1/STQS3113/Slides/R
tutorial/oxygen.xlsx")
```

Exporting csv file from R

- To save a data frame into a CSV file, use `write.csv()` function
  - Example:

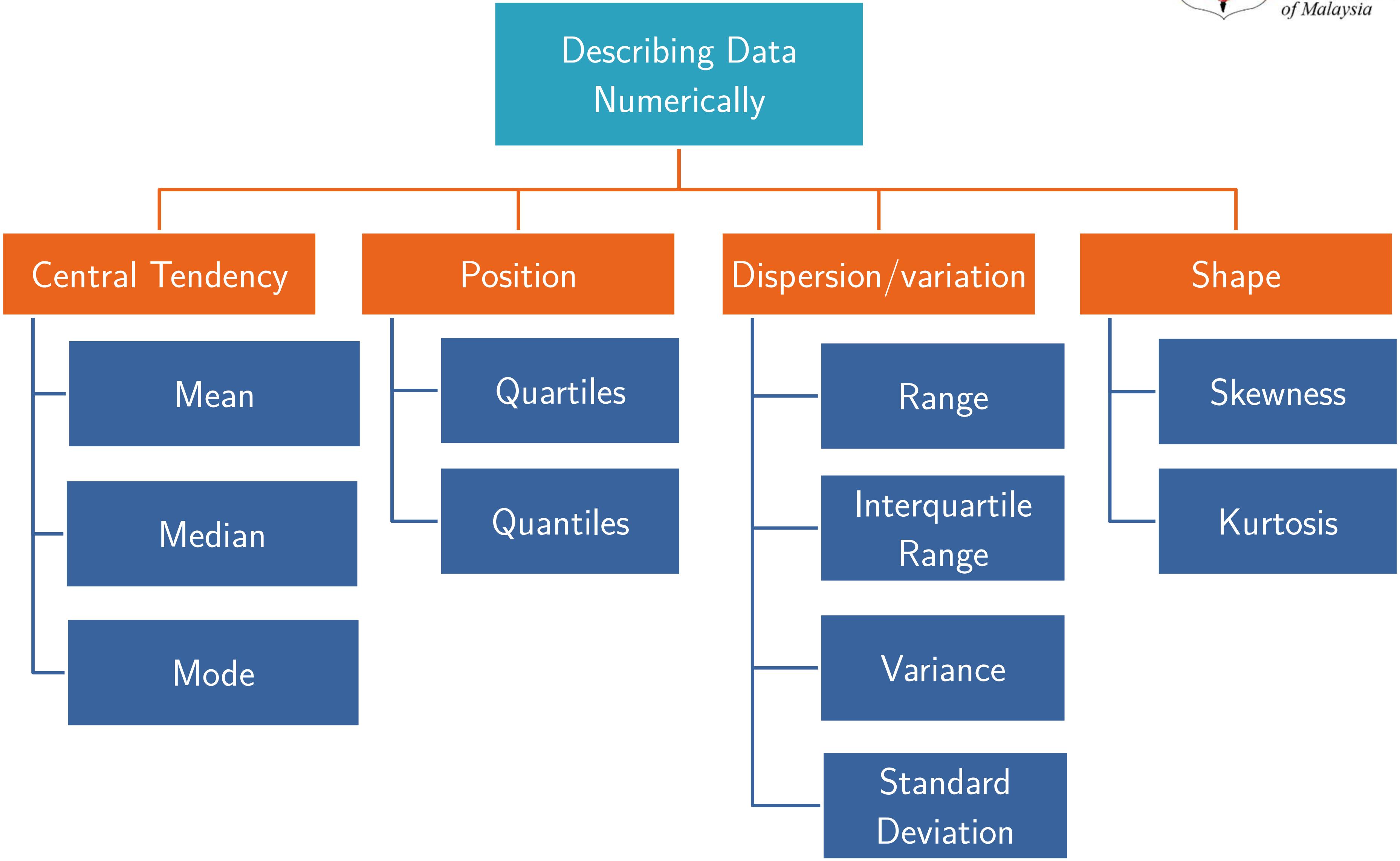
```
write.csv(TestScore, file="TestScore.csv",
                row.names=FALSE)
```

- If the data frame is too large, you can use `write_csv()` function from the `readr` package.

- (If you want to save any object in R, use the `save()` function)

Numerical descriptive statistics

Mean, median, mode, variance, standard deviation, etc



Describing Data Numerically

- Central Tendency
  - Mean
  - Median
  - Mode
- Position
  - Quartiles
  - Quantiles
- Dispersion/variation
  - Range
  - Interquartile Range
  - Variance
  - Standard Deviation
- Shape
  - Skewness
  - Kurtosis

Summary statistics

- The `summary()` function when applied to vector or data frame gives the mean, min, max, first quarter, median, third quarter of each variables.

- If we want more summary statistics, we can use `describe()` function in the `psych` package.

Summary statistics

- Example:

```
> summary(TestScore)
    Student           Score              IQ
 Min.   : 1.00   Min.   : 65.00   Min.   : 85.0
 1st Qu.: 3.25   1st Qu.: 75.75   1st Qu.: 95.0
 Median : 5.50   Median : 82.50   Median :100.0
 Mean   : 5.50   Mean   : 83.20   Mean   :100.9
 3rd Qu.: 7.75   3rd Qu.: 91.50   3rd Qu.:104.8
 Max.   :10.00   Max.   :100.00   Max.   :125.0
   StudyHours         GPA          Class
 Min.   : 0.00   Min.   :1.500   A:5
 1st Qu.:11.25   1st Qu.:2.100   B:5
 Median :20.00   Median :2.300
 Mean   :18.50   Mean   :2.430
 3rd Qu.:23.75   3rd Qu.:2.675
 Max.   :40.00   Max.   :3.900
```

Summary statistics

- Example:

```
> install.packages("psych")
> library(psych)
> describe(TestScore)
           vars  n   mean    sd median trimmed   mad  min
Student       1 10   5.50  3.03    5.5    5.50  3.71  1.0
Score         2 10  83.20 11.10   82.5   83.38 12.60 65.0
IQ            3 10 100.90 11.22  100.0   99.88  7.41 85.0
StudyHours    4 10  18.50 11.80   20.0   18.12 11.12  0.0
GPA           5 10   2.43  0.68    2.3    2.36  0.52  1.5
Class*        6 10   1.50  0.53    1.5    1.50  0.74  1.0
             max range  skew kurtosis   se
Student     10.0   9.0  0.00    -1.56 0.96
Score      100.0  35.0 -0.06    -1.43 3.51
IQ         125.0  40.0  0.65    -0.32 3.55
StudyHours  40.0  40.0  0.14    -0.99 3.73
GPA          3.9   2.4  0.73    -0.27 0.21
Class*       2.0   1.0  0.00    -2.19 0.17
```

Table of frequency

- If the vector or variable is discrete, use `table()` function to create a table for the frequency of each value.

- Example:

```
> x <- c(3,2,2,4,2,1,2,2,3,4)
> table(x)
x
1 2 3 4
1 5 2 2
```

Universiti Kebangsaan Malaysia
The National University of Malaysia

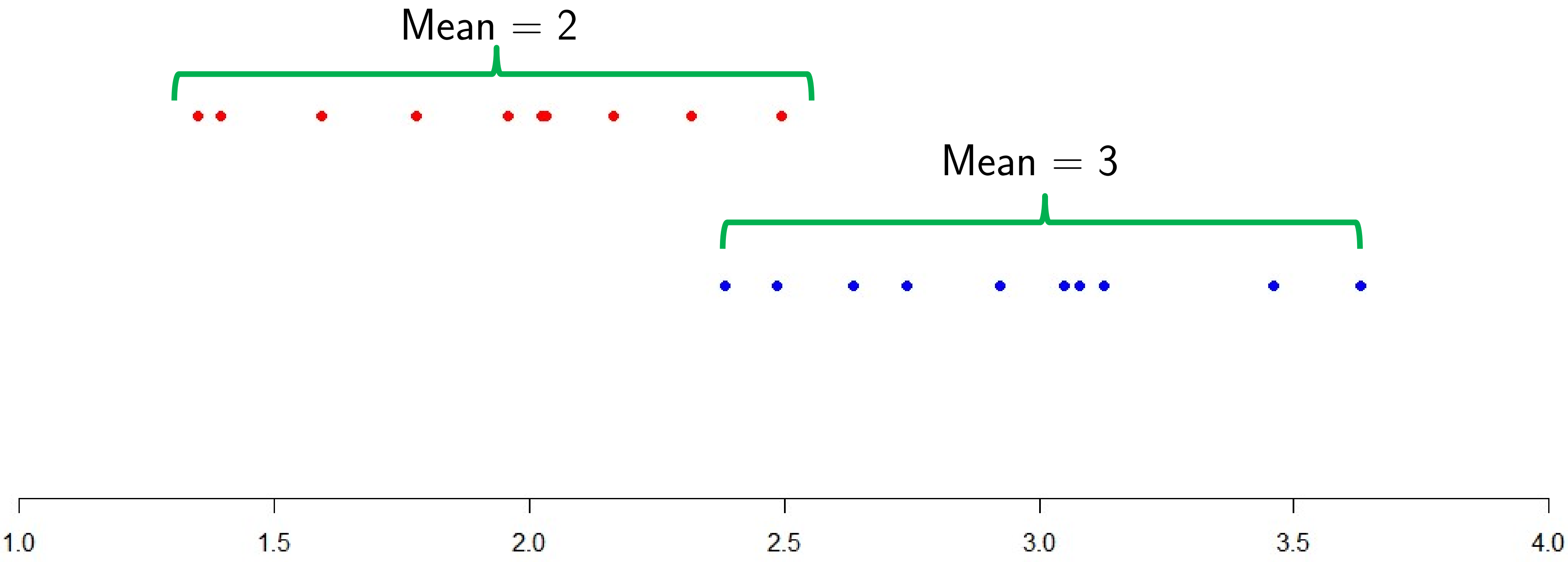
Measures of central tendency

Mean, median, mode

Mean

- What it is:
  - The sum of all the data entries divided by the number of entries.

$$\text{Mean}, \bar{x} = \frac{\sum x}{n}$$



Mean

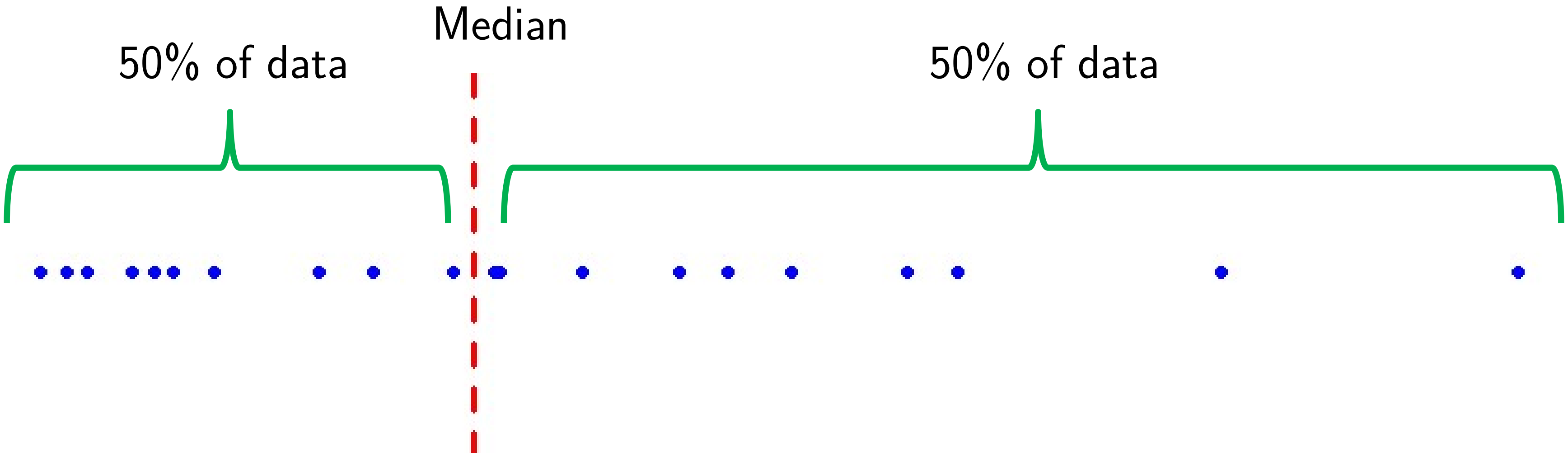
- Use `mean()` function to calculate the mean of a vector.

- Example:

```
> mean(TestScore$Score)
[1] 83.2
> mean(TestScore$IQ)
[1] 100.9
> mean(TestScore$StudyHours)
[1] 18.5
```

Median

- What it is:
  - The value that lies in the middle of the data when the data set is ordered.



Median

- Use `median()` function to calculate the median of a vector.

- Example:

```
> median(TestScore$Score)
[1] 82.5
> median(TestScore$IQ)
[1] 100
> median(TestScore$StudyHours)
[1] 20
```

Mode

- What it is:
  - The data entry that occurs with the highest number of frequency.

- If no entry is repeated the data set has no mode.

- If two entries occur with the same greatest frequency, each entry is a mode (bimodal).

Mode

- If the vector or variable is discrete, we can use the `table()` function to find the frequency of each value and find the one with the highest frequency.
- Example:

```
> x <- c(3,2,2,4,2,1,2,2,3,4)
> table(x)
x
1 2 3 4
1 5 2 2
> names(table(x))[which.max(table(x))]
[1] "2"
```

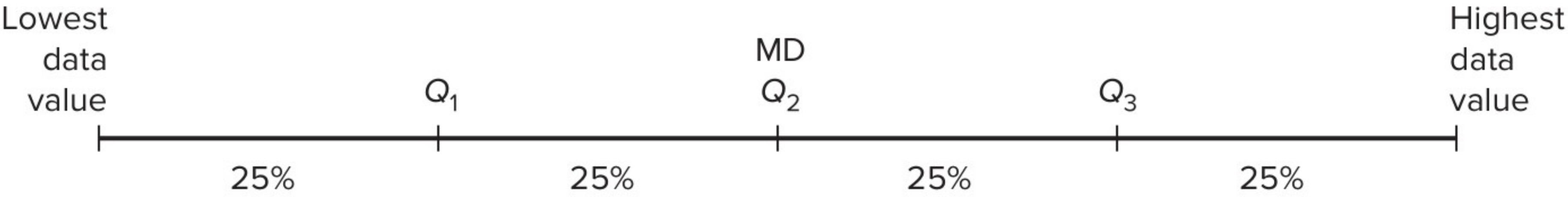
Measures of position

Quartiles, quantiles

Quartile

- What it is:
  - Quartiles divide the distribution into four equal groups.

- The boundaries for the groups are denoted by Q_1 , Q_2 , Q_3.



- Eg: Approximately 75% of values in a ranked data set are more than Q_1.

- Note that the median is equal to Q_2.

Quartile

- The `quantile()` function by default gives the min, Q_1, Q_2, Q_3, and the max of a vector.

- But the argument can be modified to find any quantiles.

- Example:

```
> quantile(TestScore$Score)
    0%    25%    50%    75%   100% 
 65.00  75.75  82.50  91.50 100.00 
> quantile(TestScore$Score, probs=c(0.1,0.9))
 10%  90% 
71.3 95.5 
```

Measures of dispersion

Variance, standard deviation, range, interquartile range

Variance & standard deviation

- What it is:
  - The values tell how closely the observations are to the mean.
  - Variance and standard deviation are the most used measures of dispersion.

- Sample variance and sample standard deviation:

$$s^2 = \frac{\sum(x-\bar{x})^2}{n-1}, \qquad s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$$

- The lower the value, the closer the observations are to the sample mean.

Variance & standard deviation

- The `var()` and `sd()` functions calculate these values.

- Example:

```
> var(TestScore$Score)
[1] 123.2889
> sd(TestScore$Score)
[1] 11.10355
```

- Note that these are sample variance and standard deviation, not population variance and standard deviation.

Range

- What it is: Range = Largest value – smallest value

- We can use `max()` and `min()` to get the largest and smallest values respectively to calculate range.

- The function `range()` also gives the largest and smallest value.

- Example:

```
> max(TestScore$Score) - min(TestScore$Score)
[1] 35
> range(TestScore$Score)
[1]  65 100
```

Interquartile range

- What it is:
  - The difference between Q_1 and Q_3.
  - $IQR = Q_3 – Q_1$

- The `IQR()` function calculates the interquartile range.

- Example:

```
> quantile(TestScore$Score, c(0.25,0.75))
  25%   75% 
75.75 91.50 
> IQR(TestScore$Score)
[1] 15.75
```

Shape of the distribution

Skewness and kurtosis

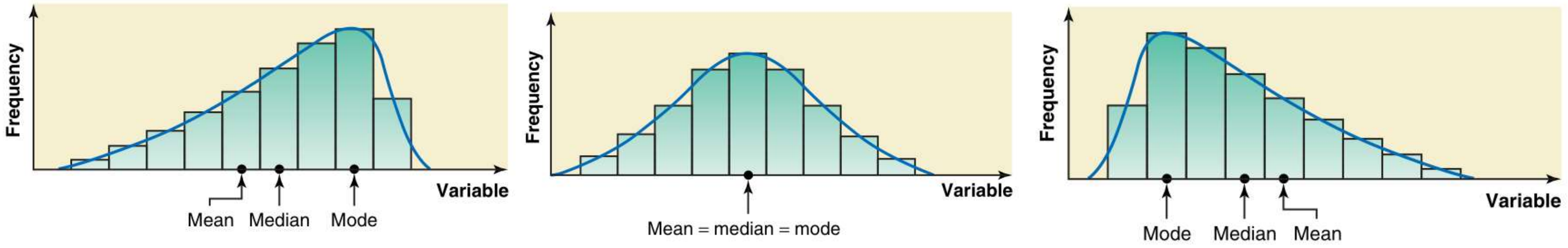
Skewness

- Skewness measures the degree of asymmetry exhibited by the data.

$$\text{skewness} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^3}{ns^3}$$

- Skewness:
  - zero: data is symmetric about the mean
  - negative: data is skewed to the left
  - positive: data is skewed to the right

Skewness



□ We can use `skewness()` function from `e1071` package to calculate this value.

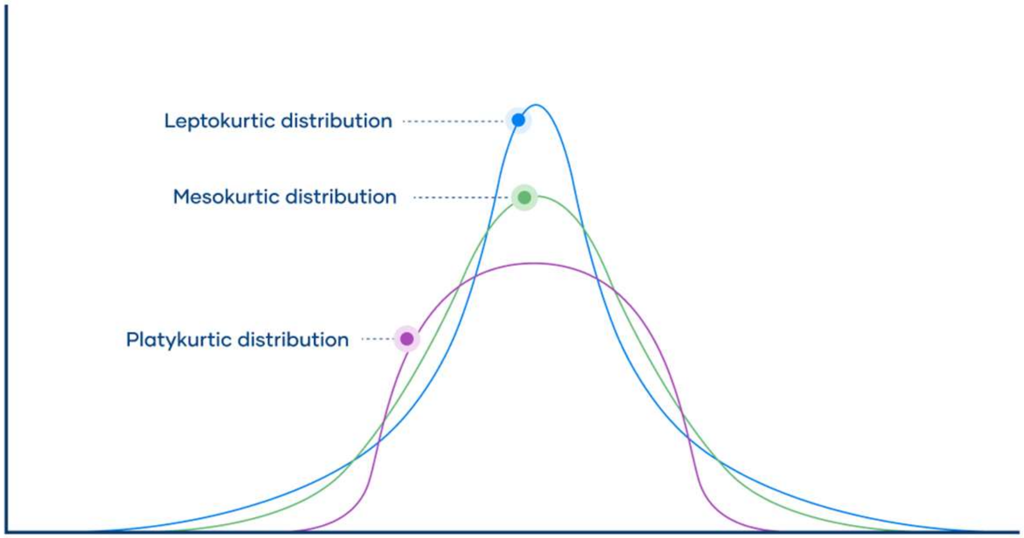
□ Example:

```
> library(e1071)
> skewness(TestScore$Score)
[1] -0.05638212
```

Kurtosis

- Kurtosis is a measure of the heaviness of the tail of a distribution.
- It is often used to represent how often outliers occur
- Excess kurtosis is the tailedness of a distribution relative to a normal distribution.
- Excess kurtosis:
  - High/positive (leptokurtic): fat tails
  - Low/negative (platykurtic): thin tails
  - Medium/zero (mesokurtic): medium tails

Kurtosis



- □ Example:

```
> library(e1071)
> kurtosis(TestScore$Score)
[1] -1.434192
```

Functions & descriptions

Function	Description
`data.frame()`	Creates a new data frame class object.
`str()`	Gives the structure of an object.
`subset()`	Creates a subset of the data frame based on the specified condition.
`names()`	Gives the available variable names in the data frame/list.
`colnames()`, `rownames()`	Gives the column and row names in the data frame/matrix.
`head()`, `tail()`	Gives the first few rows and last few rows in the data frame.
`getwd()`, `setwd()`	Gives and sets the location of current working directory.
`read.table()`	Imports CSV or TXT files as data frame.
`read.csv()`	Imports CSV files as data frame.
`write.csv()`	Exports a data frame into a CSV file.

Functions & descriptions

Function	Description
`summary()`	When the argument/input is a vector, it gives some summary statistics including min, max, mean, and the quartiles.
`describe()`	Gives more summary statistics. This function requires `psych` package to be loaded first.
`table()`	Gives a table of frequency.
`mean()`, `median()`	Calculates the mean and median of a vector.
`quantile()`	Gives the sample quantiles. By default, it gives min, Q_1, Q_2, Q_3 and max.
`IQR()`	Calculates the interquartile range of a vector.
`var()`, `sd()`	Calculates the (sample) variance and standard deviation of a vector.
`skewness()`, `kurtosis()`	Gives skewness and kurtosis measures of a vector. These functions require `e1071` package to be loaded first.