Universiti Kebangsaan Malaysia
The National University of Malaysia

NARATIF BAHARU UKM
UNIVERSITI WATAN KITA

FAKULTI SAINS & TEKNOLOGI

DATA EXPLORATION WITH R

NORATIQAH MOHD ARIFF

IMPORT AND EXPORT DATA USING R

TXT, CSV AND XLSX

IMPORT DATA INTO R

- Reading data from text file.

```
read.table(file.choose(),header=T)
read.table(file.choose(),header=T,sep="\t")
```

- Reading data from excel file.
- Save the file as a csv file.

```
read.csv(file.choose(),header=T)
```

IMPORT DATA INTO R

- Or use the “`readxl`” package

```
library(readxl)
read_excel(file.choose(), sheet="Sheet1")
```

- To find out what sheets available: `excel_sheets(file.choose())`

- Read data as a clipboard.
- Highlight wanted data. Copy the data.

```
read.table(file= "clipboard",header=T,sep= "\t")
```

EXPORT DATA FROM R

- Save data to text file.

```
write.table(variable,file="Rclass.txt", col.names=T, row.names=F)
```

- Save data to excel file.
- Save the file as a csv file.

```
write.csv(variable,file="Rclass.csv", col.names=T,  row.names=F)
```

EXPORT DATA FROM R

- Save data as a clipboard.

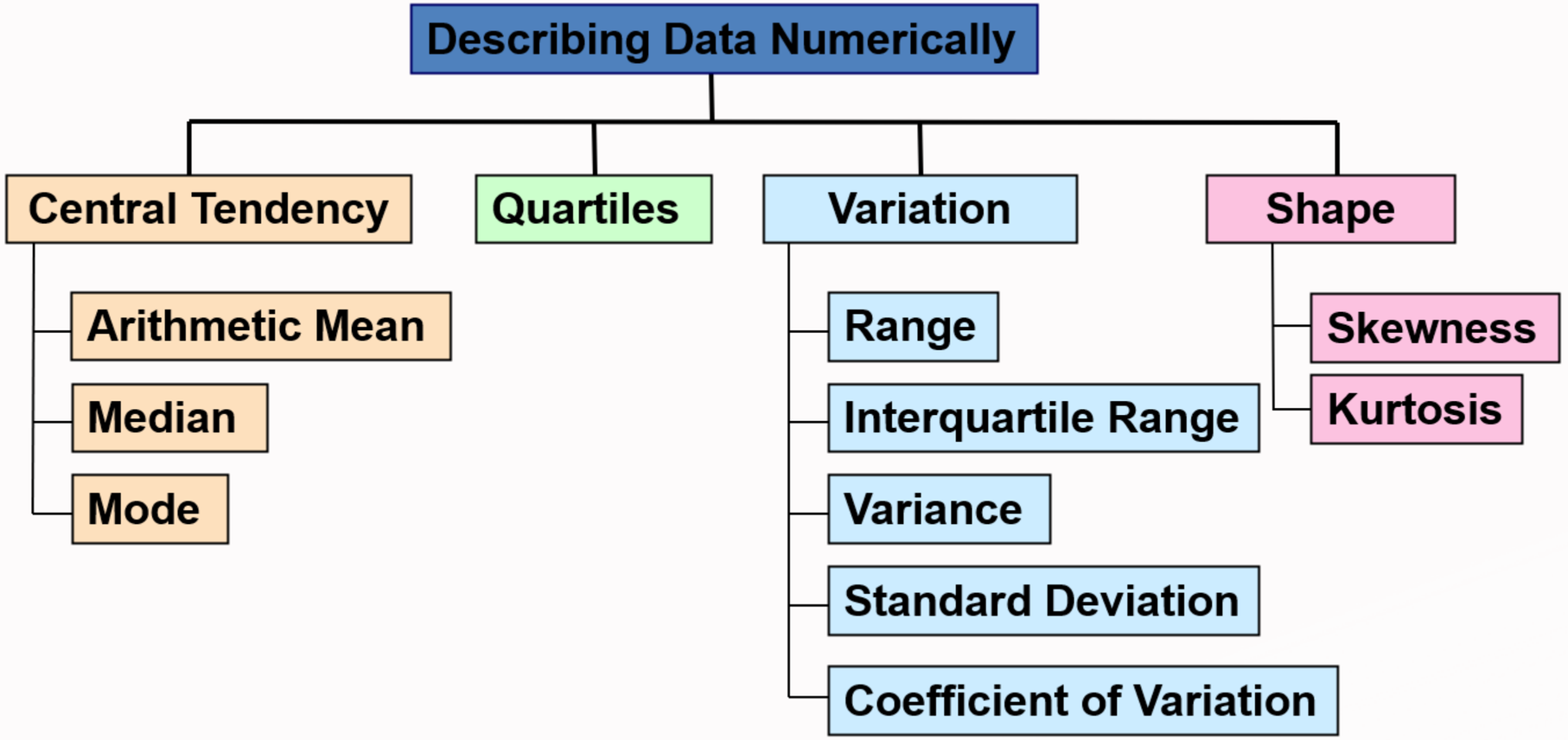
```
write.table(variable,col.names=T,row.names=F,
file= "clipboard",sep= "\t")
```

- Paste to wanted file.

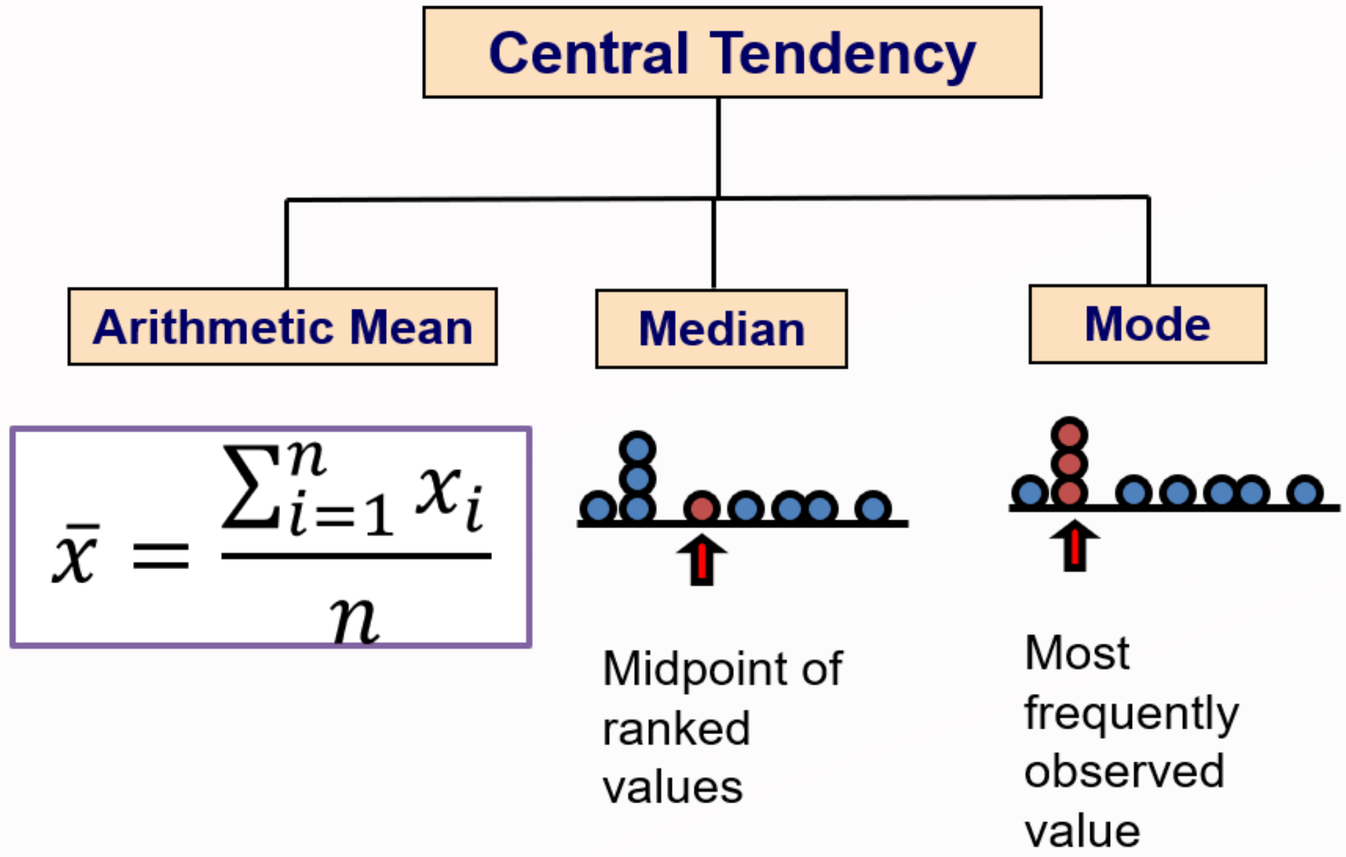
DESCRIPTIVE STATISTICS USING R

FUNCTIONS FOR SUMMARY STATISTICS

DESCRIPTIVE STATISTICS USING R: Summary Measures

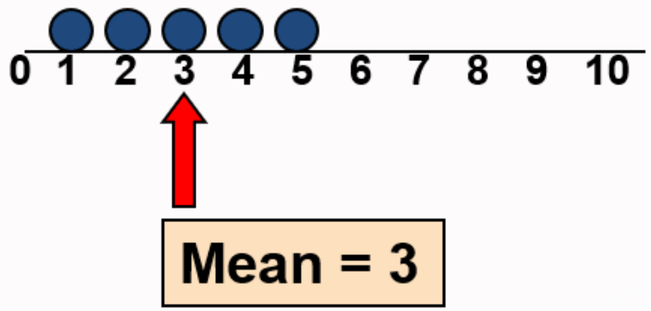


DESCRIPTIVE STATISTICS USING R: Measures of Central Tendency

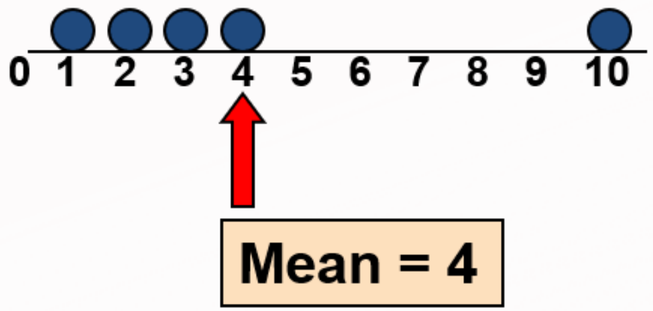


DESCRIPTIVE STATISTICS USING R: Arithmetic Mean

- The most common measure of central tendency
- Mean = sum of values divided by the number of values
- Affected by extreme values (outliers)



$$\frac{1+2+3+4+5}{5}=\frac{15}{5}=3$$

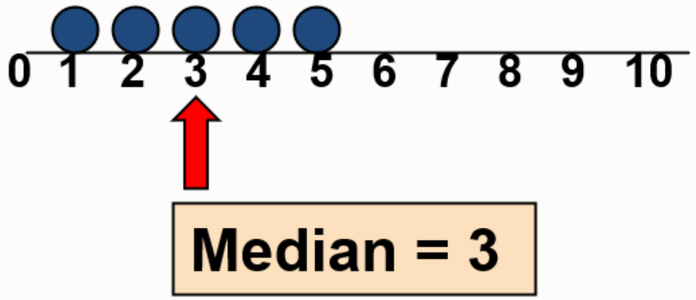


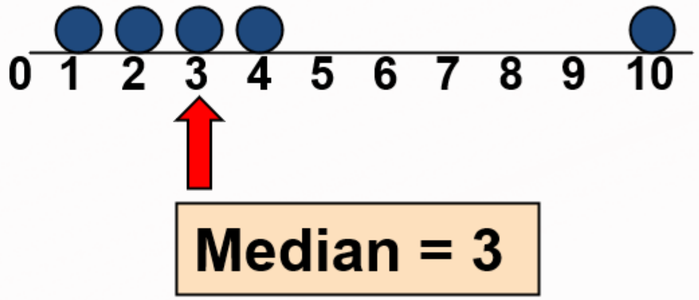
$$\frac{1+2+3+4+10}{5}=\frac{20}{5}=4$$

R code: `mean()`, `rowMeans()`, `colMeans()`

DESCRIPTIVE STATISTICS USING R: Median

- In an ordered array, the median is the “middle” number (50% above, 50% below)
- Not affected by extreme values





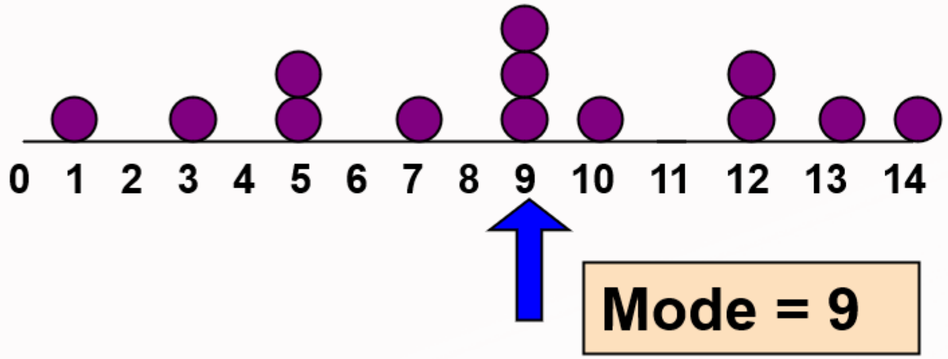
R code: `median()`

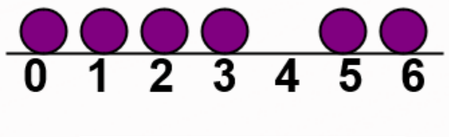
DESCRIPTIVE STATISTICS USING R: Mode

- Value that occurs most often (most popular)
- Not affected by extreme values
- Used for either numerical or categorical (nominal) data
- There may be no mode
- There may be several modes

R code:

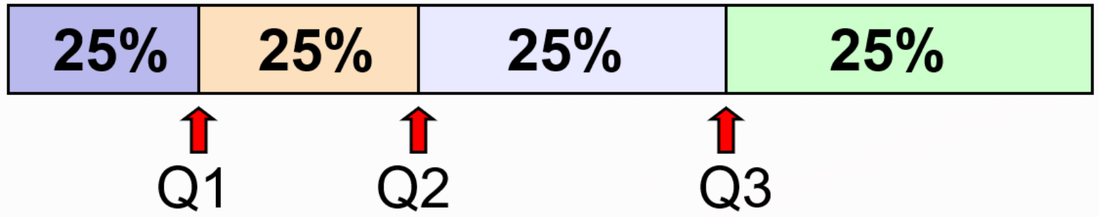
```
x <- c(1,2,2,2,3,4,5,6,7,8,9)
temp<-table(x)
names(temp)[temp == max(temp)]
```





DESCRIPTIVE STATISTICS USING R: Quartiles

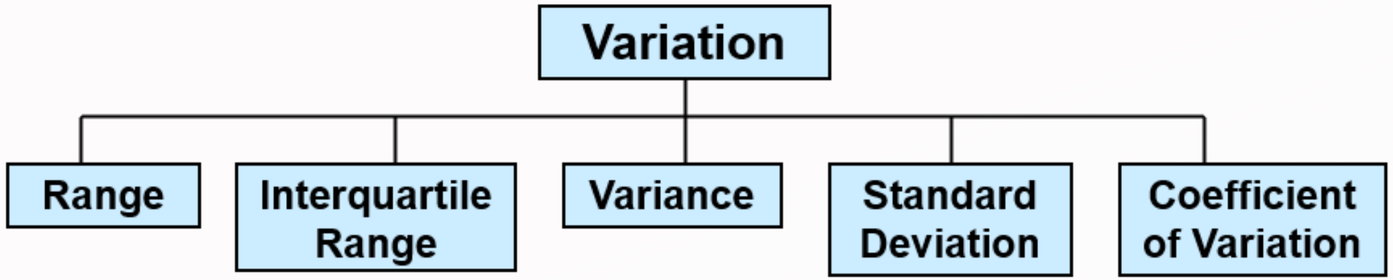
- Quartiles split the ranked data into 4 segments with an equal number of values per segment
- The first quartile, Q1, is the value for which 25% of the observations are smaller and 75% are larger
- Q2 is the same as the median (50% are smaller, 50% are larger)
- Only 25% of the observations are greater than the third quartile

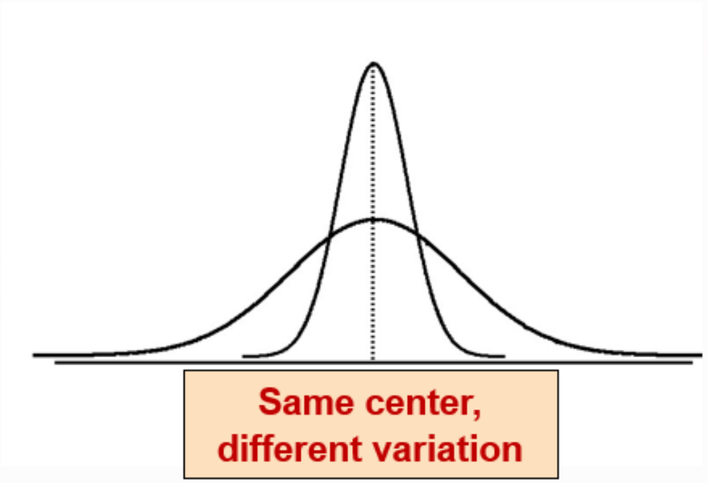


R code: `quantile()`, `summary()`

DESCRIPTIVE STATISTICS USING R: Measures of Variation

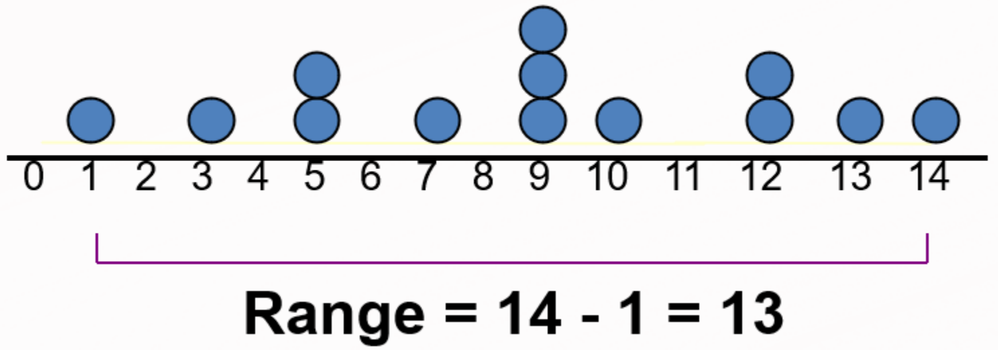
- Measures of variation give information on the spread or variability of the data values.





DESCRIPTIVE STATISTICS USING R: Range

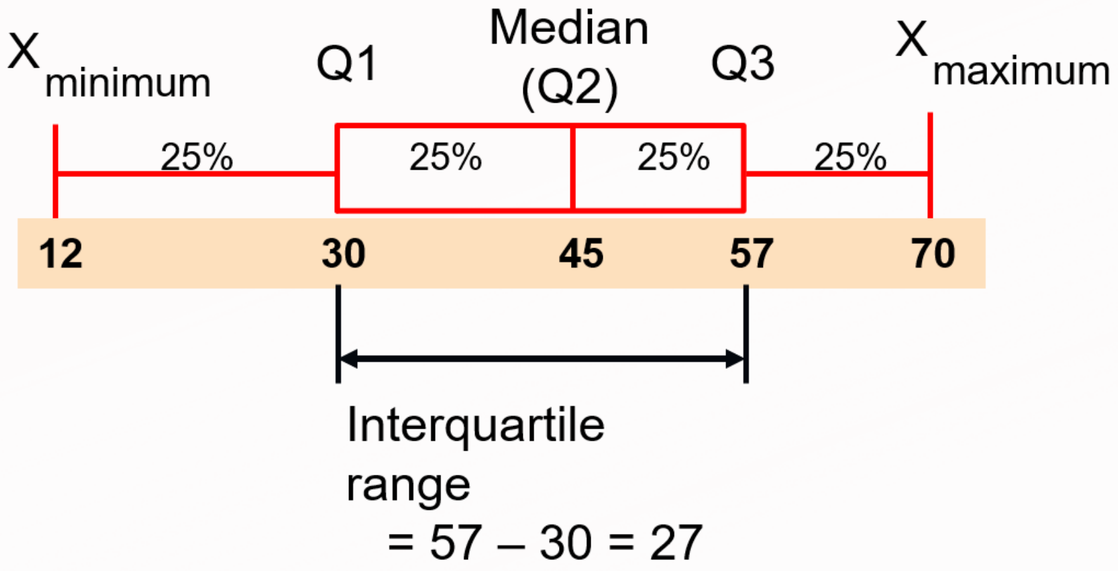
- Simplest measure of variation
- Difference between the largest and the smallest values in a set of data.



R code: `max() - min()`
#In R, `range()` will give you two values which is the minimum and maximum values

DESCRIPTIVE STATISTICS USING R: Interquartile Range

- Difference between the largest and the first and third quartile in a set of data.



R code: `IQR()`

DESCRIPTIVE STATISTICS USING R: Variance

- Average (approximately) of squared deviations of values from the mean.
- Sample variance:

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}$$

where

$\bar{X}$ = mean

n = sample size

X_i = i^{th} value of the variable X

R code: `var()`
#This will give the sample variance

DESCRIPTIVE STATISTICS USING R: Standard Deviation

- Most commonly used measure of variation
- Shows variation about the mean
- Is the square root of the variance
- Has the same units as the original data
- Sample standard deviation:

$$S = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}}$$

where

$\bar{X}$ = mean

n = sample size

X_i = i^{th} value of the variable X

R code: `sd()`

#This will give the sample standard deviation

DESCRIPTIVE STATISTICS USING R: Coefficient of Variation

- Measures relative variation
- Always in percentage (%)
- Shows variation relative to mean
- Can be used to compare two or more sets of data measured in different units

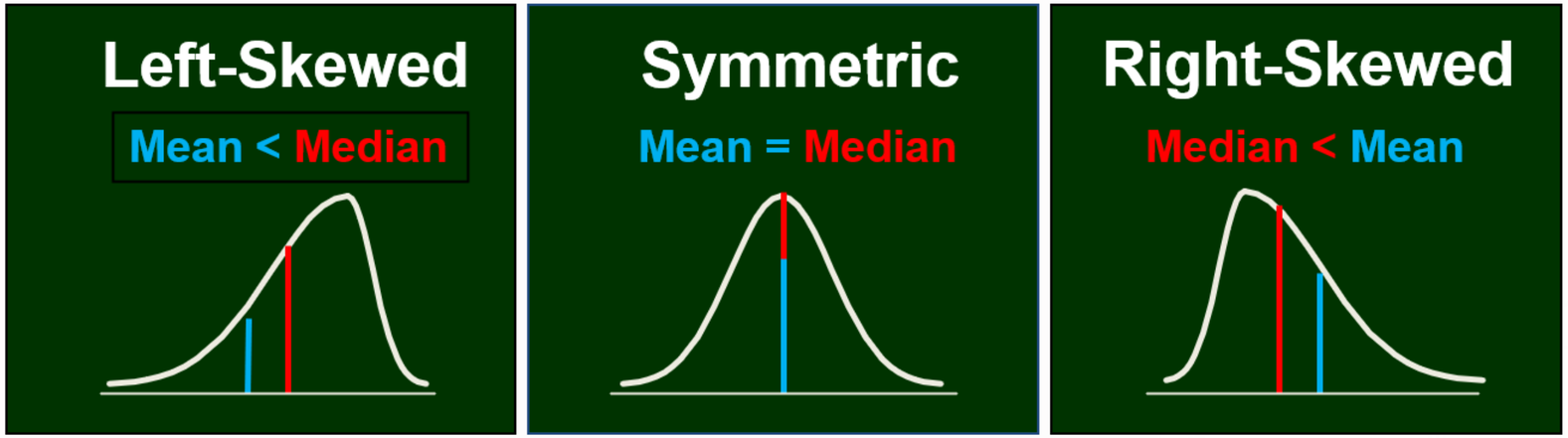
$$CV = \left(\frac{S}{\overline{X}} \right) \times 100\%$$

R code: `sd()/mean()*100`

DESCRIPTIVE STATISTICS USING R: Measures of Shape: Skewness

- Skewness measures the degree of asymmetry exhibited by the data.
- If skewness equals zero, the histogram is symmetric about the mean.

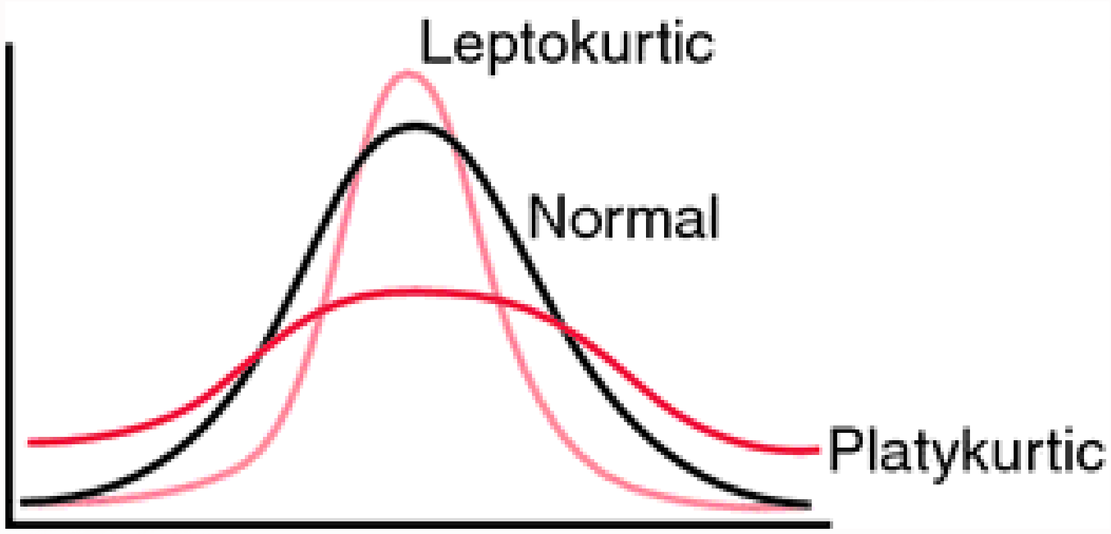
$$skewness = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^3}{ns^3}$$



R code: `library(e1071)`
`skewness()`

DESCRIPTIVE STATISTICS USING R: Measures of Shape: Kurtosis

- Kurtosis is based on the size of a distribution's tails.
- Negative kurtosis (platykurtic) – distributions with short tails.
- Positive kurtosis (leptokurtic) – distributions with relatively long tails.



R code: `library(e1071)`
`kurtosis()`

SIMPLE DATA WRANGLING WITH R

COMMON FUNCTIONS

COMMON FUNCTIONS IN R:
`head` and `tail`, `which` function

- Obtain the first several rows of a matrix or data frame using `head` and use `tail` to obtain the last several rows.
- Example:

```
head(trees,3)
tail(trees,2)
```

- `which` returns the position of the vector where `which` are TRUE
- Example:

```
which(mtcars$cyl==6)
mtcars[which(mtcars$cyl==4&mtcars$gear==5),]
```

COMMON FUNCTIONS IN R:
`subset, sort` and `order` function

- `subset` is the easiest way to select variables and observations.
- Example:

```
subset(mtcars,subset=hp<100,select=mpg)
```

- The `sort` function sorts a vector or factor into ascending or descending order.
- Example:

```
sort(mtcars$mpg,decreasing=T)
```

- The `order` function sorts a data frame.
- Example:

```
mtcars[order(mtcars$mpg, -mtcars$cyl),]
```

COMMON FUNCTIONS IN R:
`merge` function

- `merge` joins two data frames by one or more common key variables.
- Example:

Name	Age	Weight
Abu	23	78
Ali	25	75
Ahmad	24	80
Amir	22	74

Data A

Name	Height	Address
Abu	80	Urban
Ahmad	87	Rural
Amir	79	Rural
Ali	80	Urban

Data B

```
dataC<-merge(dataA,dataB,by="Name")
```

COMMON FUNCTIONS IN R:
`factor` function

- The output levels will tell us how many and what factors are in the data set.
- Example:

```
num<-c(1,2,2,3,1,2,3,3,1,2,3,3)
fnum<-factor(num)
```

- We can also relabel the factors.

```
fnum<-factor(num,labels=c("I","II","III"))
```

COMMON FUNCTIONS IN R:
`table` function

- Tabulate data and provide the frequency of each factor.
- Example:

```
mons <- c("March","April","January","November","January","September",
"October","September", "November","August","January",
"November", "November", "February","May","August","July",
"December","August","August","September","November",
"February","April")
table(mons)
```

COMMON FUNCTIONS IN R:
`table` function

- We can also reorder the table.
- Example:

```
mons <- factor(mons,levels=c("January","February","March",
"April","May","June","July", "August","September", "October",
"November", "December"), ordered=TRUE)
table(mons)
```

- Names of all factors: `names(table(mons))`

COMMON FUNCTIONS IN R:
`table` function

- Tabulate continuous data by first creating intervals using the `cut`() function.
- Convert a numeric variable into a factor.
- The argument `breaks=` describe how ranges of numbers will be converted to factor values.
- If `breaks=`(a number), the resulting factor will be created by dividing the range of the variable into that number of equal-length intervals.
- If `breaks=`(a vector of values), the values in the vector are used to determine the breakpoints. The number of levels will be one less than the number of values in the vector

COMMON FUNCTIONS IN R:
`table` function

- Example:

```
wfact <- cut(women$weight,breaks=3)

wtab <- table(wfact)
```

- We can relabel the intervals

```
wfact <- cut(women$weight,breaks=3,labels=c("Low","Medium","High"))
```

- By using a vector of values.

```
wfact <- cut(women$weight,breaks=c(100,120,140,160,180))
```

COMMON FUNCTIONS IN R:
`sapply` and `lappy` function

- To calculate the same function for all the variables in a data frame.
- The function to be executed is represented by the argument `FUN=`.
- The difference between the two functions `sapply` and `lapply` is in the presentation of output.
- Example:

```
data(rock)
lapply(rock, FUN=mean)
sapply(rock, FUN=mean)
```

COMMON FUNCTIONS IN R:
`apply` and `do.call` function

- `apply` is similar to `sapply` and `lapply` function.
- Contains the argument `MARGIN=`. If `MARGIN=1`, R will apply the function to each row. If `MARGIN=2`, R will apply the function to each column.
- Example:

```
apply(rock,MARGIN=1, FUN=mean)
```

- `do.call` allows you to call any R function, but instead of writing out the arguments one by one, you can use a list to hold the arguments of the function.
- The `do.call` function only works with list as the second argument.
- Example:

```
do.call(rbind,lapply(rock,FUN=mean))
```

COMMON FUNCTIONS IN R:
`is.na` and `na.omit` function

- Important to recognize missing values as early as possible.
- May be part of original data or may arise as part of a computation.
- Missing values are represented by `NA` without quotes.
- To test for missing values, use `is.na` function.
- If missing values occur because of computation, it may display `Inf, -Inf` or `NaN`.
- `is.nan` can also be used to check for `Inf, -Inf` or `NaN`.
- Most statistical function accept argument `na.rm=T/F` to remove missing values.
- `na.omit` function eliminates missing values from data.

COMMON FUNCTIONS IN R: `paste` function

- Paste characters or strings together using the `paste`() or `paste0`() function
- Argument "`sep=`" for character string to separate terms
- Argument collapse "`collapse=`" for character strings to separate results
- Example:

```
mine<-c("a","b","c")
paste(mine,"!")
paste(mine,1:3)
paste0(mine,"!")
paste(mine,1:3,sep=",")
paste(mine,1:3,collapse=",")
```

COMMON FUNCTIONS IN R: Tools in `tidyverse` environment

- Functions in `dplyr` package:
  - `select`() extracts columns and returns a tibble
  - `arrange`() changes the ordering of the rows
  - `filter`() picks cases based on their values
  - `mutate`() adds new variables that are functions of existing variables
  - `rename`() easily changes the name of a column(s)
  - `pull`() extracts a single column as a vector

THANK YOU