Command lines

- R command lines are written with appropriate arguments following the R prompt,>, in R console
- Data is represented in R by means of vectors, matrices and data frames.
- The basic data representation in R is a column vector.
- > x <- c(1,2,3,4,5,6)↵

Command lines

- R also provides a function, named *seq*, to define evenly spaced number sequences, as in the following example:

```
> seq(-1,1,0.2)
[1] -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
```

- A matrix can be obtained in R by suitably transforming a vector. For instance,

```
> dim(x) <- c(2,3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

transforms (through the dim function) the previous vector x into a matrix of 2×3 elements. Note the display of row and column numbers.

Column binding

- We can also aggregate vectors into a matrix by using the function *cbind*

```
> u <- c(1,2,3)
> v <- c(-1,-2,-3)
> m <- cbind(u,v)
> m
     u  v
[1,] 1 -1
[2,] 2 -2
[3,] 3 -3
```

```
> m <- rbind(u,v)
> m
  [,1] [,2] [,3]
u    1    2    3
v   -1   -2   -3
```

Matrix indexing

- Matrix indexing in R uses square brackets as index qualifier. As an example,

```
>m[2,2] has the value -2.
>m[,2]
     2 -2
>m[2,]
     [1] -1 -2 -3
>m[,2:3]
    [,1] [,2]
u    2    3
v   -2   -3
```

```
> m
    [,1]  [,2]  [,3]
u    1     2     3
v   -1    -2    -3
```

Command lines

- Which objects are currently in the console environment ?
- *ls*() (“list”), such as: > ls()

  [1] "m" "u" "v" "x"
- Object removal is performed by applying the function rm (“remove”) to a list of object identifiers.

  > rm(x)

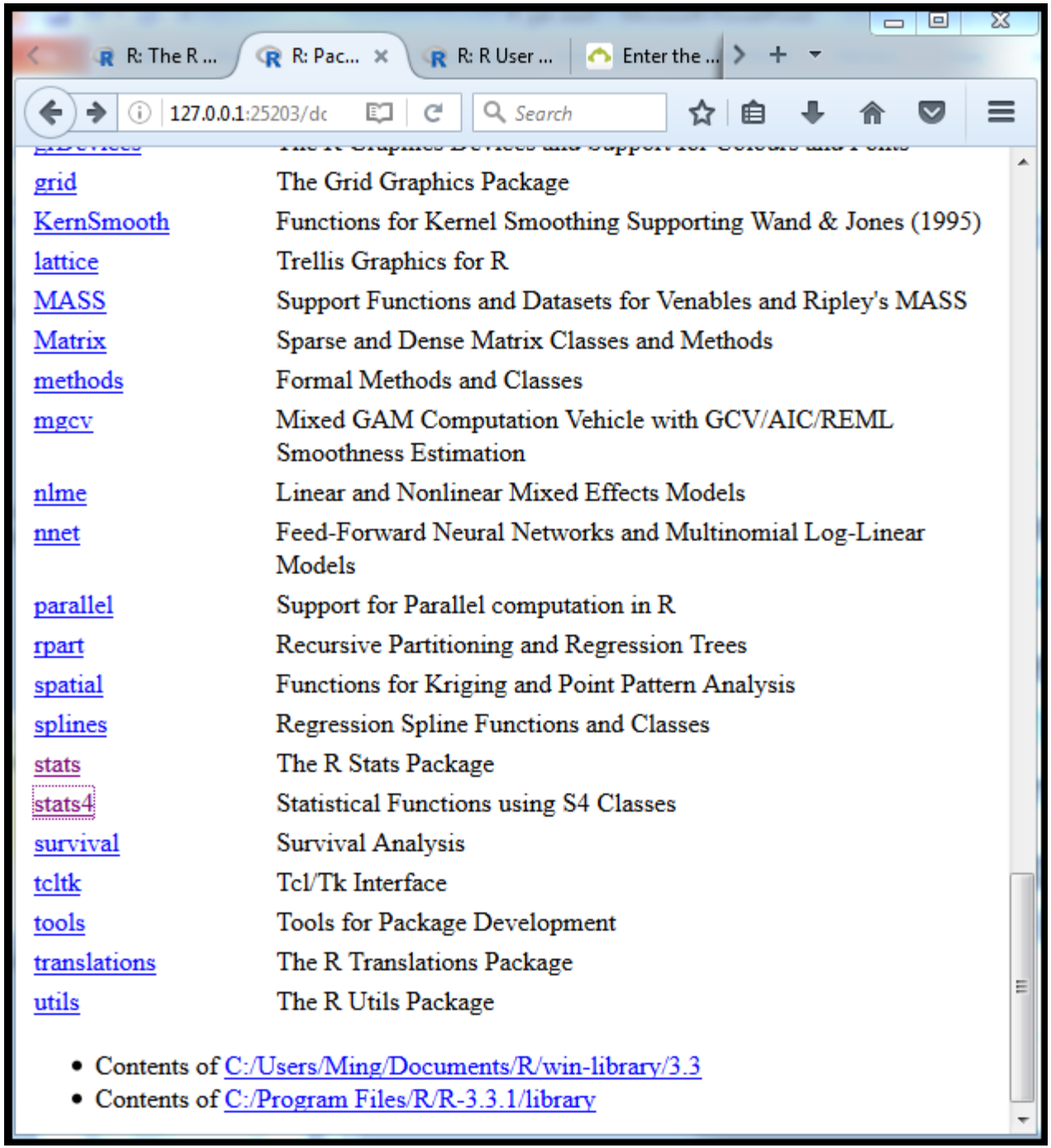
  > ls()

  [1] "m" "u" "v"

Package

- The functions available in R are collected in so-called **packages**.
- We can inspect which packages are currently loaded by issuing the *search*() command.

```
> search()
[1] ".GlobalEnv"      "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils“ "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```



grid	The Grid Graphics Package
KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)
lattice	Trellis Graphics for R
MASS	Support Functions and Datasets for Venables and Ripley's MASS
Matrix	Sparse and Dense Matrix Classes and Methods
methods	Formal Methods and Classes
mgcv	Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation
nlme	Linear and Nonlinear Mixed Effects Models
nnet	Feed-Forward Neural Networks and Multinomial Log-Linear Models
parallel	Support for Parallel computation in R
rpart	Recursive Partitioning and Regression Trees
spatial	Functions for Kriging and Point Pattern Analysis
splines	Regression Spline Functions and Classes
stats	The R Stats Package
stats4	Statistical Functions using S4 Classes
survival	Survival Analysis
tcltk	Tcl/Tk Interface
tools	Tools for Package Development
translations	The R Translations Package
utils	The R Utils Package

- Contents of C:/Users/Ming/Documents/R/win-library/3.3
- Contents of C:/Program Files/R/R-3.3.1/library

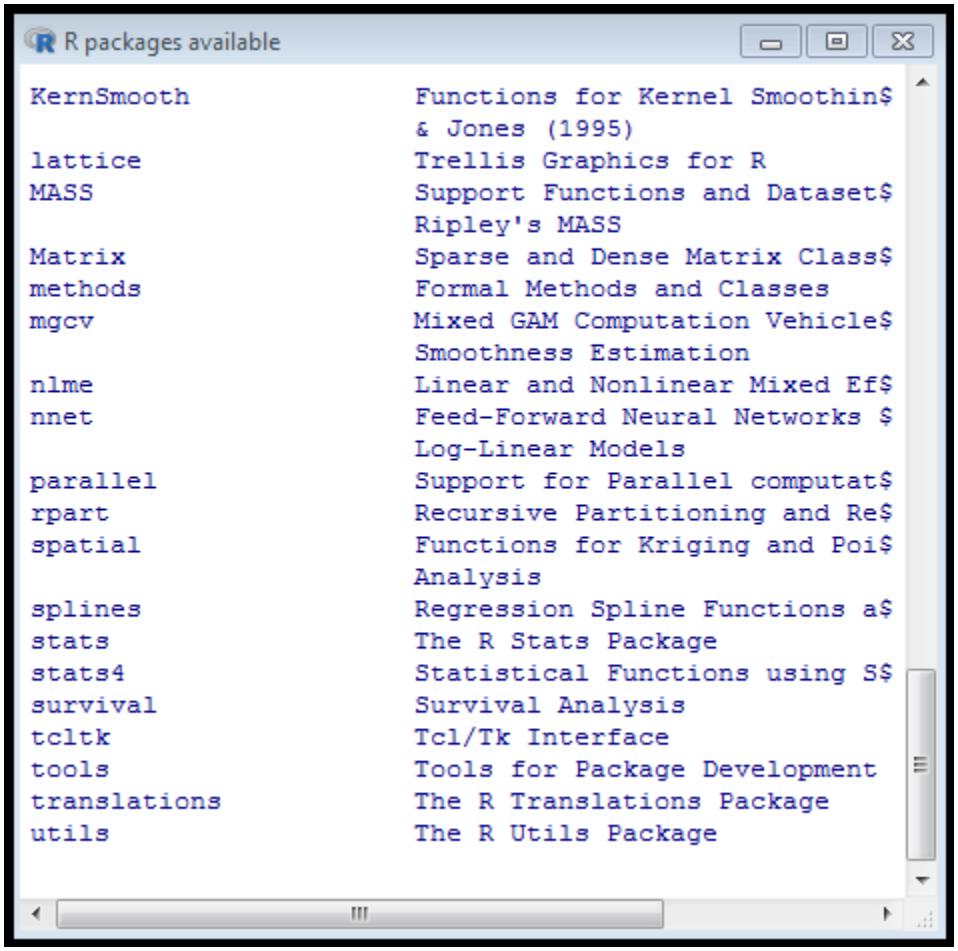
Help

- To get the information of which functions are available in the stats package one may issue the *help.start*() command.
- An Internet window pops up from where one clicks on "Packages" and obtains the "Package Index" window partially shown in the Figure.
- By clicking on stats of the "Package Index", a complete list of the available stats functions.

Command *library*()

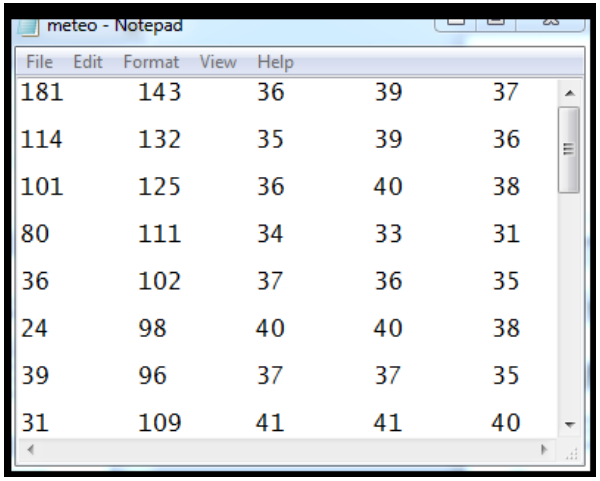
> library()

- List of the packages installed will be shown.



Descriptive statistics
R Data Entry

- The tabular form of data in R is called *data frame.*
- We can create an R data frame from a text file, Ex. meteo.txt



> meteo <- read.table(file("meteo.txt"))

R Data Entry

```
> meteo <- read.table(file("meteo.txt"))
```

- file is the path to the file we want to read in.
- read.table creates data frame and return to meteo

➢ meteo

```
      V1   V2  V3  V4  V5
1     181 143  36  39  37
2     114 132  35  39  36
3     101 125  36  40  38
:       :                :
25     14  70  35  37  39
```

R Data Entry

- Save this data frame in meteo by

  > save(meteo,file="meteo")

- Later, we can immediately load in the data frame with load("meteo").

  > load("meteo")

R Data Entry

- To have appropriate column names for the data, instead of the default V1, V2, etc.
- Create a string vector and pass it to the read.table function as:

```
> l <- c("PMax", "RainDays", "T80", "T81", "T82")
> meteo<-read.table(file("meteo.txt"),col.names=l)
> meteo
   PMax RainDays T80 T81 T82
1   181      143  36  39  37
2   114      132  35  39  36
3   101      125  36  40  38
4    80      111  34  33  31
:
```

R Data Entry

- Column names and row names can also be set or retrieved with the functions *colnames* and *rownames*, respectively.
- Commands assigns row names to meteo corresponding to the names of the places:

```
> r <- c("V. Castelo", "Braga", "S. Tirso",
+ "Montalegre", "Bragança", "Mirandela", "M. Douro",
+ "Régua", "Viseu", "Guarda", "Coimbra", "C. Branco",
+ "Pombal", "Santarém", "Dois Portos", "Setúbal",
+ "Portalegre", "Elvas", "Évora", "A. Sal", "Beja",
+ "Amareleja", "Alportel", "Monchique", "Tavira");
> rownames(meteo) <- r;
> meteo
```

	PMax	RainDays	T80	T81	T82
V. Castelo	181	143	36	39	37
Braga	114	132	35	39	36
S. Tirso	101	125	36	40	38

…

Data operations

- Every data column can be extracted from this data frame, such as:

```
> meteo$PMax
 [1] 181 114 101  80  36  24  39  31  49  57  72  60  36  45  36  28  41  13  14
[20]  16   8  18  24  37  14
```

- Or

```
> meteo[,1]
 [1] 181 114 101  80  36  24  39  31  49  57  72  60  36  45  36  28  41  13  14
[20]  16   8  18  24  37  14
```

Data operations

- Categorize data

```
> PClass <- 1 + (meteo$PMax>20) + (meteo$PMax>80)
> PClass
 [1] 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 1
> meteo <- cbind(meteo,PClass)
> meteo
                PMax RainDays T80 T81 T82 PClass
V. Castelo   181     143          36  39  37     3
Braga        114     132          35  39  36     3
S. Tirso     101     125          36  40  38     3
Montalegre 80     111          34  33  31     2
...
Tavira         14      70          35  37  39     1
```

Data operations

- Transpose

  > x <- *t*(meteo)

- Sorting a vector

  > o <- order(meteo[,1], decreasing=TRUE)

- The permutation list can now be used to perform the sorting of PMax or any other variable of meteo:

  > x <- meteo[,1]

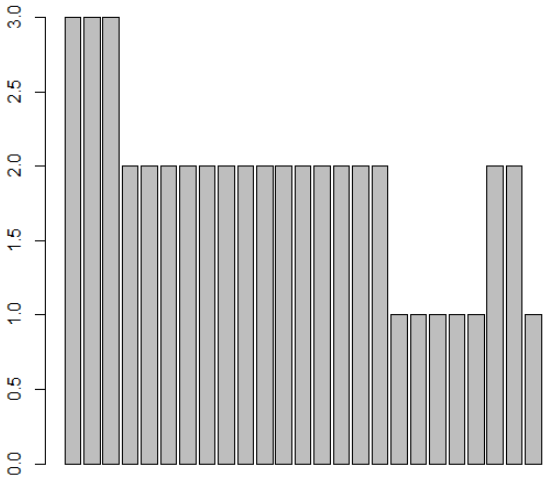
  > x[o]

  [1] 181 114 101 80 72 60 57 49 45 41 39 37 36 36 36 31 28 24 24
  [20] 18 16 14 14 13 8

Data Presentations by Counts and Bar Graphs

- Consider the Meteo dataset,
- From categories variable PClass,

```
> PClass <- 1 + (meteo[,1]>20) + (meteo[,1]>80)
> barplot(PClass)
```



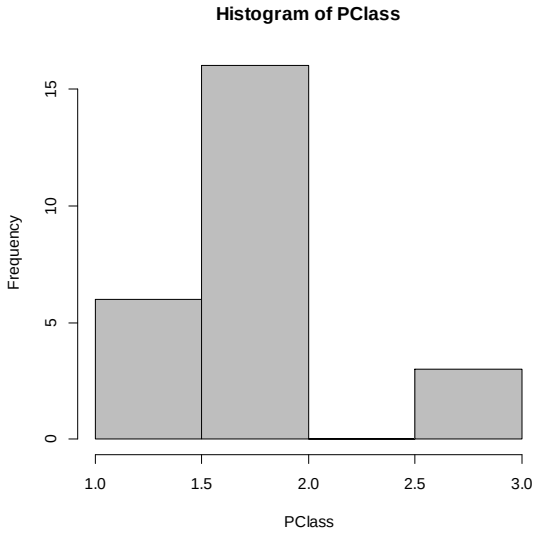
Histogram

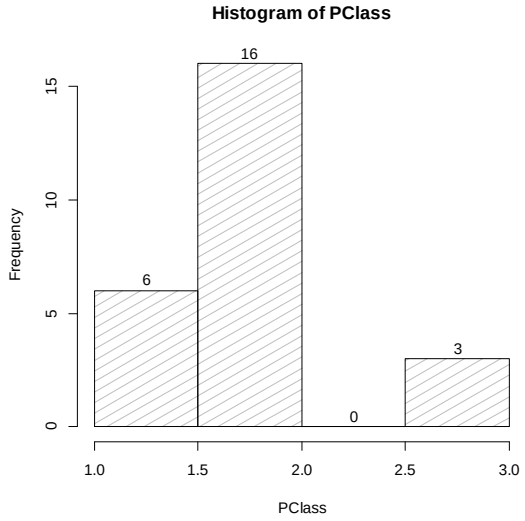
- *hist* function when applied to a discrete variable plots.

  > hist(PClass,*col*="gray");

- *col* determines the filling colour of the bars.
- Arguments for specifying shading lines, the border colour of the bars, the labels, and so on, for instance:

  > hist(PClass, density = 10, angle = 30, border = "black", col = "gray", labels = TRUE);



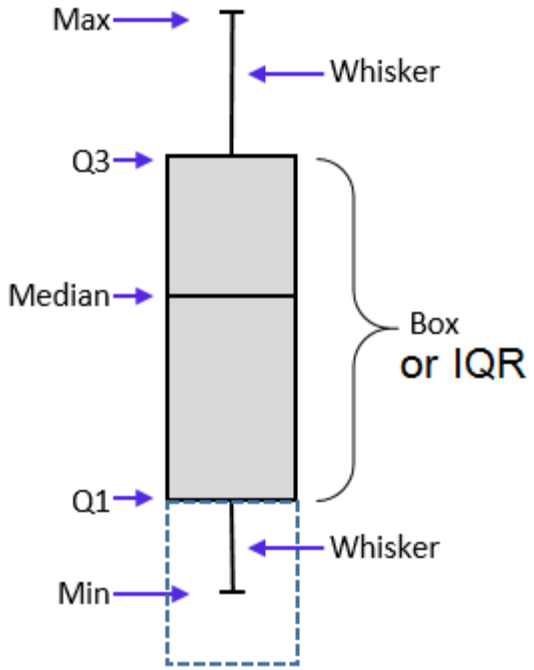


Categorized Plots

- Comparing random distributions of the same variables for different values of an extra *grouping variable*.
- For instance, in the case of the Meteorological Dataset, one might be interested in comparing Pmax for the three different groups (or *classes*) of the maximum precipitation (mm) in 1980.
- To compare the data across the groups, we have *categorized plots.*

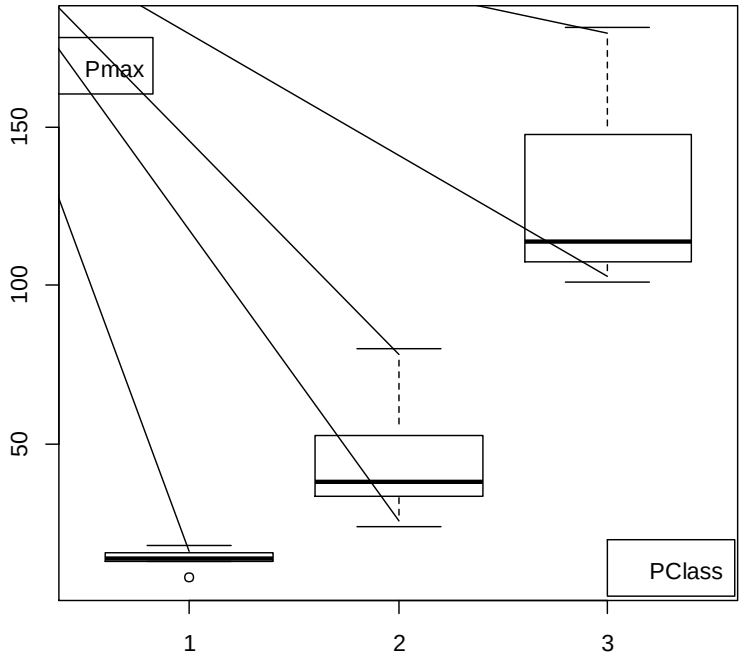
Categorized Plots

- To comparing data distributions for several groups, we can use *box plot* (or *box-and-whiskers plot*).
- Box plot uses a rectangular box for each group
- The central 50% of the cases, the so-called *inter-quartile range* (IQR).



Categorized Plots

```
> boxplot(meteo[,1]~PClass)
> legend(3,20,legend="PClass")
> legend(0.3,178,legend="Pmax")
```



Summarizing the Data

- In data analysis, usually determine some indices to give a global picture on:
  - where and how the data is concentrated and
  - what is the shape of its distribution, i.e.,

  indices that are useful for the purpose of summarizing the data.
- These indices are known as *descriptive statistics*.

Measures of Location

- *Location* determine where the data distribution is concentrated that consists of:
  - ***Arithmetic Mean***
  - ***Median***
  - ***Mode***
  - ***Quantiles***

  The quantile of order α ($0 < \alpha < 1$) of a random variable distribution $F_x(x)$ is

  $$F_X(x) = \alpha$$

  defined as the root of the equation

Quantiles

The quantile of order α ($0 < \alpha < 1$) of a random variable distribution $F_X(x)$ is defined as the root of the equation

$$F_X(x) = \alpha$$

We denote the root as: x_α.

Often used quantiles are:

- *Quartiles*, corresponding to multiples of 25% of the cases.
- *Deciles*, corresponding to multiples of 10% of the cases.
- *Percentiles*, corresponding to multiples of 1% of the cases. We will often use the percentile p = 2.5% and its complement p = 97.5%.

Measures of Location

- *mean*(), *median*(), *summary*(), and *quantile*() functions use to calculate data ***location***
- From data set meteorology, we can measure location of Pmax by

```
> mean(Pmax)
[1] 46.96
```

- *Summary()* function provides mean, median, range, and quartile

```
> summary(Pmax)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  8.00   24.00   36.00   46.96   57.00  181.00
```

Measures of Spread

- *Spread* (or *dispersion*) give an indication of how concentrated a data distribution is.
- R provides functions to measure spread consisting of:
  - IQR(x) ;
  - range(x)
  - sd(x),
  - var(x)

Measures of Shape

- ***Skewness:*** use to measure symmetrical distribution around the mean
  - *g=skewness(x)* ;
  - *g* >0 Skewed to the right,
  - *g* <0 Skewed to the left.
- ***Kurtosis:*** The degree of flatness of a probability or density function near its center
  - *k=kurtosis(x)*
  - *k* = 0 : for the normal distribution
  - Distributions flatter than the normal distribution have *k* < 0;
  - distributions more peaked than the normal distribution have *k* > 0.