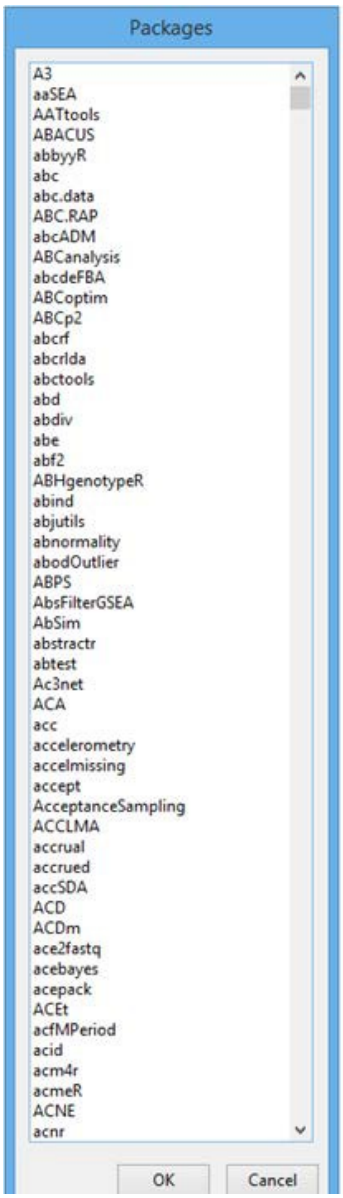
STAT S251F

STATISTICAL DATA ANALYSIS in R

Tony Chan, Ph.D.

Chapter 1 Introduction to R

1.1 Installing and Loading Add-on Packages



There are *base* packages (which come with R automatically), and *contributed* packages (which must be downloaded for installation). For example, on the version of R being used the default base packages loaded at startup are

```
> getOption("defaultPackages")
```

[1] "datasets" "utils" "grDevices" "graphics" "stats" "methods"

"CRAN task views" aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. A part of the CRAN task views is shown below:

Topics

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data

The general command install.packages() will (on most operating systems) open a window containing *a huge list of available packages* (see the table on P.1):

```
> install.packages()
```

- The base packages are maintained by a select group of volunteers, called "R Core". In addition to the base packages, there are literally thousands of additional contributed packages written by individuals all over the world.

- These are stored worldwide on mirrors of the *Comprehensive R Archive Network*, or **CRAN** for short.

- Given an active internet connection, anybody is free to download and install these packages and even inspect the source code. To install a package named foo, open up R and type install.packages("foo"). To install foo and additionally install all of the other packages on which foo depends, instead we type:

  ```
  > install.packages("foo", depends = TRUE).
  ```

- Simply choose one or more to install. No matter how many packages are installed onto the system, *each one must first be loaded for use with the library function*. For instance, the `foreign` package contains all sorts of functions needed to import data sets into R from other software such as SPSS, SAS, etc. But none of those functions will be available until the command library(`foreign`) is issued.

- Type library() at the command prompt (described below) to see a list of all available packages in your library.

For complete, precise information regarding installation of R and add-on packages, see the R Installation and Administration manual, http://cran.r-project.org/manuals.html.

1.2 Communicating with R

One line at a time

This is the most basic method and is the first one that beginners will use.

RGui (R Graphics User Interface), Microsoft Windows

By the word “RGUI”, it means that an interface in which the user communicates with R by way of points-and-clicks in a menu of some sort.

Multiple lines at a time

- For longer programs (called *scripts*) there are too many codes to write all at once at the command prompt. Furthermore, for longer scripts it is convenient to be able to only modify a certain piece of the script and run it again in R.

- Programs called *script editors* are specially designed to aid the communication and code writing process. They have all sorts of helpful features including R syntax highlighting, automatic code completion, delimiter matching, and dynamic help on the R functions as they are being written.

R Editor (Windows)

In Microsoft Windows, RGui has its own built-in script editor, called R Editor. From the console window, select:

File ► New Script

A script window opens, and the lines of code can be written in the window.

When the programmer satisfies with the codes, the user highlights all of the commands and presses:

Ctrl + R

The commands are automatically run at once in R and the output is shown.

To save the script for later, click File . Save as... in R Editor. The script can be reopened later with:

File ► Open Script...

in RGui.

1.3 Terminologies in R

A *library* is a directory that contains a set of packages.

Use library() and give it the name of the package you want to install.

1.4 Loading Data from an Excel File

Problem
You want to load data from an Excel file.

Solution
The xlsx package has the function read.xlsx() for reading Excel files. This will read the first sheet of an Excel spreadsheet:

```
# Only need to install once
> install.packages("xlsx")
> library(xlsx)
> data <- read.xlsx("datafile.xlsx", 1)
```

For reading older Excel files in the *.xls* format, the gdata package has the function read.xls():

```
# Only need to install once
> install.packages("gdata")
> library(gdata)
# Read first sheet
> data <- read.xls("datafile.xls")
```

```
> data <- read.xlsx("C:\\Users\\Tony\\Desktop\\HWData.xlsx", 1)
```

OR

```
> data <- read.xlsx("C:/Users/Tony/Desktop/HWData.xlsx", 1)
```

Discussion
With read.xlsx(), you can load from other sheets by specifying a number for sheetIndex or a name for sheetName:

```
> data <- read.xlsx("datafile.xls", sheetIndex=2)
> data <- read.xlsx("datafile.xls", sheetName="Revenues")
```

With read.xls(), you can load from other sheets by specifying a number for sheet:

```
> data <- read.xls("datafile.xls", sheet=2)
```

Remarks

(1) "read.xlsx( )" vs "read.xlsx2( )"

Both functions work exactly the same except that:

- read.xlsx( ) is slow for large data sets (worksheet with more than 100,000 cells).
- read.xlsx2( ) is faster on big files.

(2) If the worksheet is named, we can use `sheetIndex= "Sheet1".`

(3) `startRow =2` can ignore the first row containing the variable names.

1.5 Loading Data from an SPSS File

Problem

You want to load data from an SPSS file.

Solution

The foreign package has the function read.spss() for reading SPSS files. To load data from the first sheet of an SPSS file:

```
# Only need to install the first time
> install.packages("foreign")
```

library(foreign)

```
> data <- read.spss("datafile.sav")
```

Chapter 2

2.1 Basic Calculations

To get started, we'll use R like a simple calculator.

2.1.1 Addition, Subtraction, Multiplication and Division

Math	R	Result
$3+2$	`3 + 2`	5
$3-2$	`3 - 2`	1
$3 \cdot 2$	`3 * 2`	6
$3/2$	`3 / 2`	1.5

2.1.2 Exponents

Math	R	Result
3^2	`3 ^ 2`	9
$2^{(-3)}$	`2 ^ (-3)`	0.125
$100^{1/2}$	`100 ^ (1 / 2)`	10
$\sqrt{100}$	`sqrt(100)`	10

2.1.3 Mathematical Constants

Math	R	Result
π	`pi`	3.1415927
e	`exp(1)`	2.7182818

2.1.4 Logarithms

Note that we will use ln and log interchangeably to mean the natural logarithm. There is no ln() in R, instead it uses log() to mean the natural logarithm.

Math	R	Result
$\log(e)$	`log(exp(1))`	1
$\log_{10}(1000)$	`log10(1000)`	3
$\log_2(8)$	`log2(8)`	3
$\log_4(16)$	`log(16, base = 4)`	2

2.1.5 Trigonometry

Math	R	Result
$\sin(\pi/2)$	`sin(pi / 2)`	1
$\cos(0)$	`cos(0)`	1

2.2 Getting Help

In using R as a calculator, we have seen a number of functions: sqrt(), exp(), log() and sin(). To get documentation about a function in R, simply put a question mark in front of the function name and R will display the documentation, for example:

```
> ?log

> ?sin

> ?paste

> ?lm
```

- Frequently one of the most difficult things to do when learning R is asking for help.
- First, you need to decide to ask for help, then you need to know *how* to ask for help. Your very first line of defense should be to Google your error message or a short description of your issue.
- The ability to solve problems using this method is quickly becoming an extremely valuable skill.) If that fails, and it eventually will, you should ask for help.

2.3 Data

2.3.1 Data Types

R has a number of basic data *types*.

- Numeric
  - Also known as Double. The default type when dealing with numbers.
  - Examples: `1`, `1.0`, `42.5`
- Integer
  - Examples: `1L`, `2L`, `42L`
- Complex
  - Example: `4 + 2i`
- Logical
  - Two possible values: `TRUE` and `FALSE`
  - You can also use `T` and `F`, but this is *not* recommended.
  - `NA` is also considered logical.
- Character
  - Examples: `"a"`, `"Statistics"`, `"1 plus 2."`

2.3.2 Data Structures

R also has a number of basic data *structures*. A data structure is either homogeneous (all elements are of the same data type) or heterogeneous (elements can be of more than one data type).

Dimension	**Homogeneous**	**Heterogeneous**
1	Vector	List
2	Matrix	Data Frame
3+	Array	

(a) Vectors

Many operations in R make heavy use of **vectors**. Vectors in R are indexed starting at 1. That is what the [1] in the output is indicating, that the first element of the row being displayed is the first element of the vector. Larger vectors will start additional rows with [*] where * is the index of the first element of the row.

Possibly the most common way to create a vector in R is using the c() function, which is short for "combine". As the name suggests, it combines a list of elements separated by commas.

```
> c(1, 3, 5, 7, 8, 9)

## [1] 1 3 5 7 8 9
```

Here R simply outputs this vector. If we would like to store this vector in a **variable** we can do so with the **assignment** operator =. In this case the variable x now holds the vector we just created, and we can access the vector by typing x.

```
> x = c(1, 3, 5, 7, 8, 9)

> x

## [1] 1 3 5 7 8 9
```

Frequently you may wish to create a vector based on a sequence of numbers. The quickest and easiest way to do this is with the ":" operator, which creates a sequence of integers between two specified integers.

```
> (y = 1:100)
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
## [19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
## [55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
## [73] 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
## [91] 91 92 93 94 95 96 97 98 99 100
```

Here we see R labeling the rows after the first since this is a large vector. Also, we see that by putting parentheses around the assignment, R both stores the vector in a variable called y and automatically outputs y to the console. Note that scalars do not exists in R. They are simply vectors of length 1.

```
> 2

## [1] 2
```

If we want to create a sequence that isn't limited to integers and increasing by 1 at a time, we can use the seq() function.

```
> seq(from = 1.5, to = 4.2, by = 0.1)

## [1] 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9
3.0 3.1 3.2 3.3

## [20] 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2
```

We will discuss functions in detail later, but note here that the input labels from, to, and by are optional.

```
> seq(1.5, 4.2, 0.1)

## [1] 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9
3.0 3.1 3.2 3.3

## [20] 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2
```

Remarks

We have now seen four different ways to create vectors:

- `c()`
- `:`
- `seq()`
- `rep()`

So far we have mostly used them in isolation, but they are often used together.

```
> c(x, rep(seq(1, 9, 2), 3), c(1, 2, 3), 42, 2:4)

## [1] 1 3 5 7 8 9 1 3 5 7 9 1 3 5 7 9 1 3 5 7 9 1 2 3 42

## [26] 2 3 4
```

The length of a vector can be obtained with the length() function.

```
> length(x)

## [1] 6

> length(y)

## [1] 100
```

(b) Subsetting

To subset a vector, we use square brackets, [].

```
> x
## [1] 1 3 5 7 8 9


> x[1]
## [1] 1


> x[3]
## [1] 5
```

We see that x[1] returns the first element, and x[3] returns the third element.

```
> x[-2]
## [1] 1 5 7 8 9
```

We can also exclude certain indexes, in this case the second element.

```
> x[1:3]
## [1] 1 3 5

> x[c(1,3,4)]
## [1] 1 5 7
```

(c) Logical Operators

Operator	Summary	Example	Result
x < y	x less than y	3 < 42	TRUE
x > y	x greater than y	3 > 42	FALSE
x <= y	x less than or equal to y	3 <= 42	TRUE

Operator	Summary	Example	Result
x >= y	x greater than or equal to y	3 >= 42	FALSE
x == y	xequal to y	3 == 42	FALSE
x != y	x not equal to y	3 != 42	TRUE
!x	not x	!(3 > 42)	TRUE
x | y	x or y	(3 > 42) | TRUE	TRUE
x & y	x and y	(3 < 4) & (42 > 13)	TRUE

In R, logical operators are vectorized.

```
x = c(1, 3, 5, 7, 8, 9)
x > 3
## [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE
x < 3
## [1]  TRUE FALSE FALSE FALSE FALSE FALSE
x == 3
## [1] FALSE  TRUE FALSE FALSE FALSE FALSE
x != 3
## [1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
x == 3 & x != 3
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
x == 3 | x != 3
## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

(d) Matrices

R can also be used for **matrix** calculations. Matrices have rows and columns containing a single data type. In a matrix, the order of rows and columns is important. (This is not true of *data frames*, which we will see later.) Matrices can be created using the matrix function.

```
x = 1:9
x
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

```
X = matrix(x, nrow = 3, ncol = 3)
X
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Note here that we are using two different variables: lower case x, which stores a vector and capital X, which stores a matrix. This follows the usual mathematical convention. We can do this because ***R is case sensitive.***

<u>Remark</u>
By default the matrix function reorders a vector into columns, but we can also tell R to use rows instead.

```
Y = matrix(x, nrow = 3, ncol = 3, byrow = TRUE)
Y
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

R can then be used to perform matrix calculations.

```r
x = 1:9
y = 9:1
X = matrix(x, 3, 3)
Y = matrix(y, 3, 3)
X
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
Y
```

```
##      [,1] [,2] [,3]
## [1,]    9    6    3
## [2,]    8    5    2
## [3,]    7    4    1
```

```r
X + Y
```

```
##      [,1] [,2] [,3]
## [1,]   10   10   10
## [2,]   10   10   10
## [3,]   10   10   10
```

```r
X - Y
```

```
##      [,1] [,2] [,3]
## [1,]   -8   -2    4
## [2,]   -6    0    6
## [3,]   -4    2    8
```

```r
X * Y
```

```
##      [,1] [,2] [,3]
## [1,]    9   24   21
## [2,]   16   25   16
## [3,]   21   24    9
```

```r
X / Y
```

```
##           [,1]      [,2]     [,3]
## [1,] 0.1111111 0.6666667 2.333333
## [2,] 0.2500000 1.0000000 4.000000
## [3,] 0.4285714 1.5000000 9.000000
```

Remark

Note that X * Y is not matrix multiplication. It is element by element multiplication. (Same for X / Y). Instead, matrix multiplication uses %*%. Other matrix functions include t() which gives the transpose of a matrix and solve() which returns the inverse of a square matrix if it is invertible.

```
X %*% Y
```

```
##      [,1] [,2] [,3]
## [1,]   90   54   18
## [2,]  114   69   24
## [3,]  138   84   30
```

```
t(X)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
Z = matrix(c(9, 2, -3, 2, 4, -2, -3, -2, 16), 3, byrow = TRUE)
Z
```

```
##      [,1] [,2] [,3]
## [1,]    9    2   -3
## [2,]    2    4   -2
## [3,]   -3   -2   16
```

```
solve(Z)
```

```
##             [,1]        [,2]       [,3]
## [1,]  0.12931034 -0.05603448 0.01724138
## [2,] -0.05603448  0.29094828 0.02586207
## [3,]  0.01724138  0.02586207 0.06896552
```

(d) Data Frames

We have previously seen vectors and matrices for storing data. We will now introduce a ***data frame*** which will be the most common way that we ***store and interact with data***.

```
example_data = data.frame(x = c(1, 3, 5, 7, 9, 1, 3, 5, 7, 9),
                          y = c(rep("Hello", 9), "Goodbye"),
                          z = rep(c(TRUE, FALSE), 5))
```

Unlike a matrix, which can be thought of as a vector rearranged into rows and columns, a data frame is not required to have the same data type for each element. A data frame is a list of vectors. So, each vector must contain the same data type, but the different vectors can store different data types.

```
example_data
```

```
##    x       y     z
## 1  1   Hello  TRUE
## 2  3   Hello FALSE
## 3  5   Hello  TRUE
## 4  7   Hello FALSE
## 5  9   Hello  TRUE
## 6  1   Hello FALSE
## 7  3   Hello  TRUE
## 8  5   Hello FALSE
## 9  7   Hello  TRUE
## 10 9 Goodbye FALSE
```

Unlike a list which has more flexibility, the elements of a data frame must all be vectors, and have the same length.

```
example_data$x
```

```
##  [1] 1 3 5 7 9 1 3 5 7 9
```

```
all.equal(length(example_data$x),
          length(example_data$y),
          length(example_data$z))
```

```
## [1] TRUE
```

```
str(example_data)
```

```
## 'data.frame':    10 obs. of  3 variables:
##  $ x: num  1 3 5 7 9 1 3 5 7 9
```

2.4 Import Data from Other File Types

The data.frame() function above is one way to create a data frame. We can also import data from various file types in into R, as well as use data stored in packages.

The example data above can also be found here as a .csv file.

- To read this data into R, we would use the read_csv() function from the readr package.
- R has a built in function read.csv() that operates very similarly.
- The readr function read_csv() has a number of advantages. For example, it is much faster reading larger data. *It also uses the tibble package to read the data as a tibble.*
- A tibble is simply a data frame that prints with sanity. Notice in the output above that we are given additional information such as dimension and variable type.

```
library(readr)
example_data_from_csv = read_csv("data/example-data.csv")
```

This particular line of code assumes that the file example_data.csv exists in a folder called data in your current working directory.

```
example_data_from_csv
```

```
## # A tibble: 10 x 3
##        x y       z
##    <dbl> <chr>   <lgl>
##  1     1 Hello   TRUE
##  2     3 Hello   FALSE
##  3     5 Hello   TRUE
##  4     7 Hello   FALSE
##  5     9 Hello   TRUE
##  6     1 Hello   FALSE
##  7     3 Hello   TRUE
##  8     5 Hello   FALSE
##  9     7 Hello   TRUE
## 10     9 Goodbye FALSE
```

Chapter 3 R Graphics

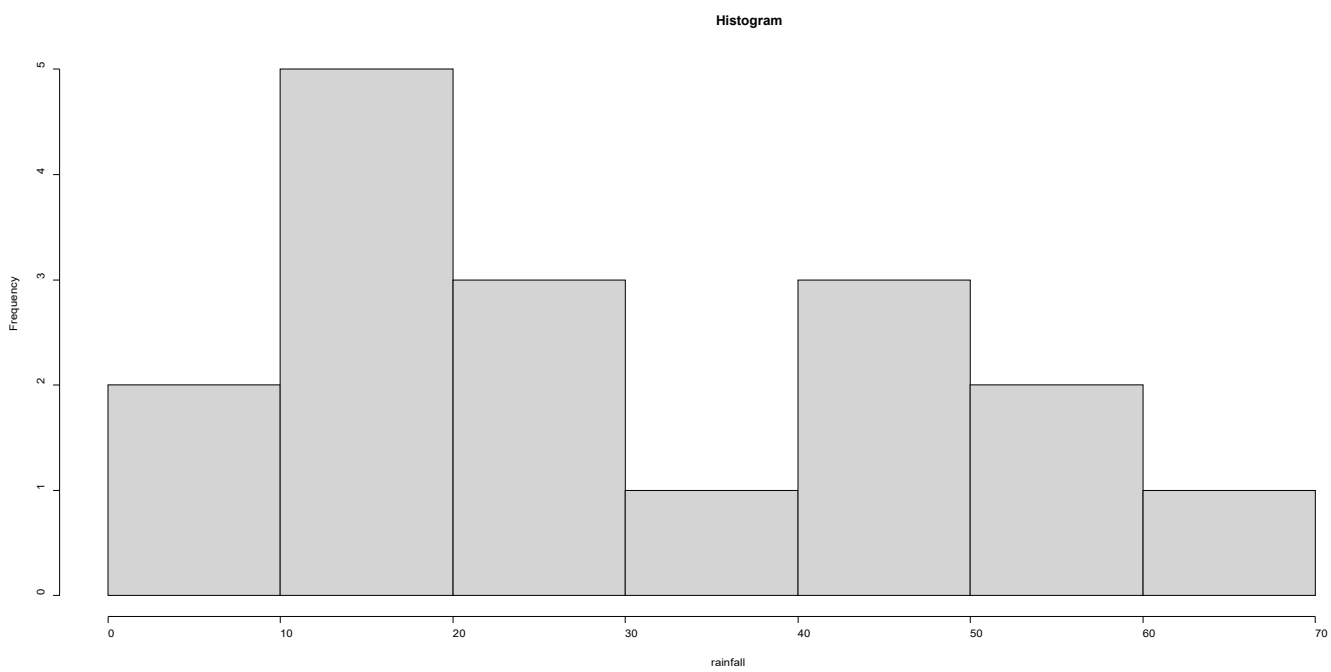
3.1 Displaying Quantitative Data

Histogram

(1) These are typically used for ***continuous data***.
(2) A histogram is constructed by first deciding on ***a set of classes/bins***, which partition the real line into a set of boxes into which the data values fall.
(3) Vertical bars are drawn over the bins with height proportional to the number of observations that fell into the bin.
(4) They are often misidentified as "bar graphs".
(5) The scale on the *y* axis can be frequency, percentage, or density (relative frequency).

<u>Example 3.1</u>
The vector, rainfall contains average amount of rainfall (in inches) for each of 10 cities in the United States.

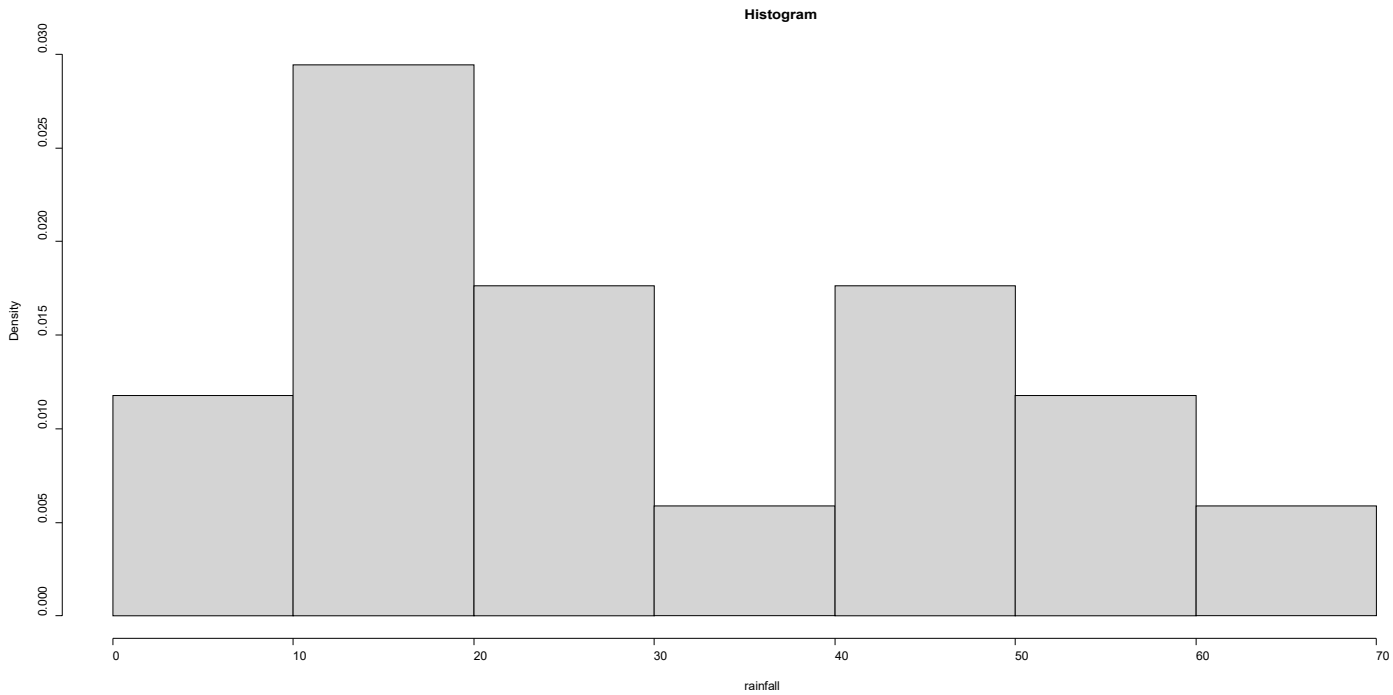
```
> rainfall <- c(67, 53.4, 54.7,7, 4, 48.5, 25.9, 14, 19.8, 17.2, 20.7,
16.3, 13, 43.4, 27.1, 40.2, 36.8)

> rainfall

[1] 67.0 54.7  7.0 48.5 14.0 17.2 20.7 13.0 43.4 40.2 36.8

> hist(rainfall, main = "Histogram")
```



- The argument main = " " suppresses the main title from being displayed – it would have said "Histogram of rainfall" otherwise.

- The plot above is a frequency histogram (the default).

```
> hist(rainfall, freq = FALSE, main = "Histogram")
```



- The plot above is a relative frequency histogram (freq = FALSE).

Remarks

(1) Weakness of histograms: the graph obtained strongly depends on the bins chosen. Choose another set of bins, and a different histogram will be obtained.

(2) There are not any definitive criteria by which bins should be defined; the best choice for a given data set is the one which illuminates the data set's underlying structure (if any).

(3) Luckily for us there are algorithms to automatically choose bins that are likely to display well, and more often than not the default bins do a good job. This is not always the case, however, and a responsible statistician will investigate many bin choices to test the stability of the display.

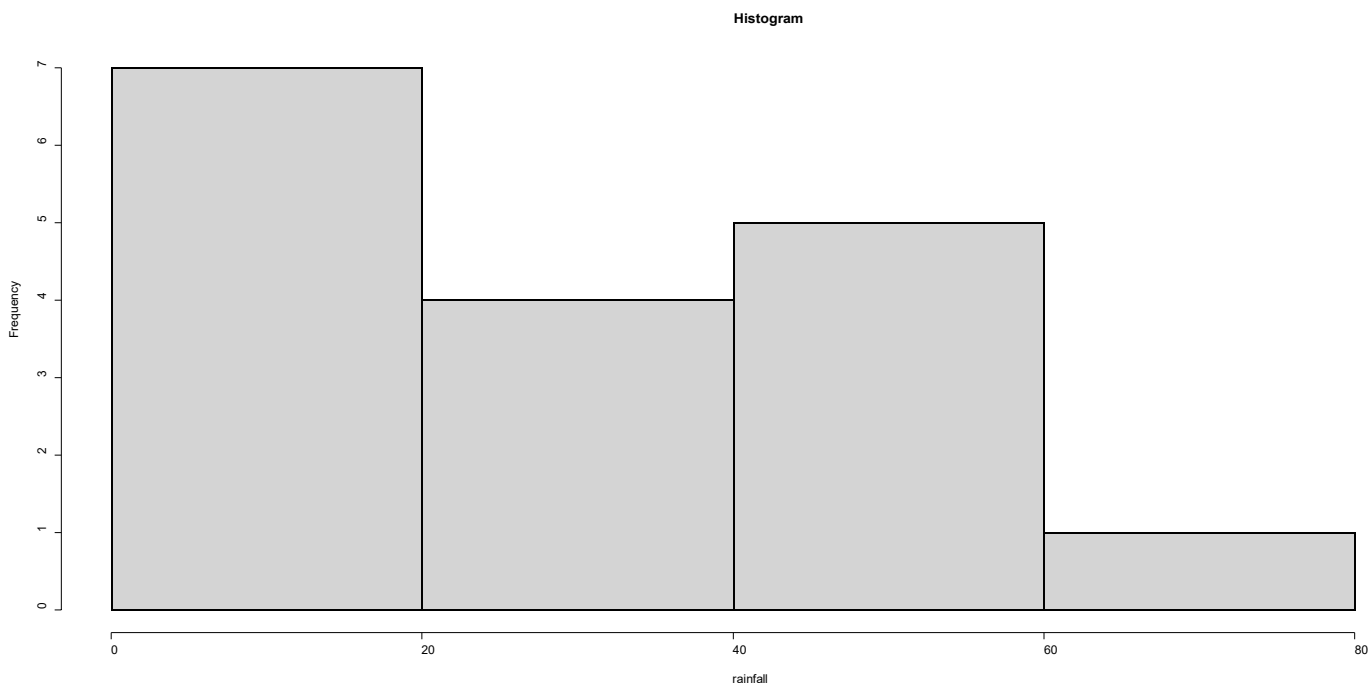
(4) Common Parameters for plot()

(a) Specifying labels:
main – provides a title
xlab – label for the x axis
ylab – label for the y axis

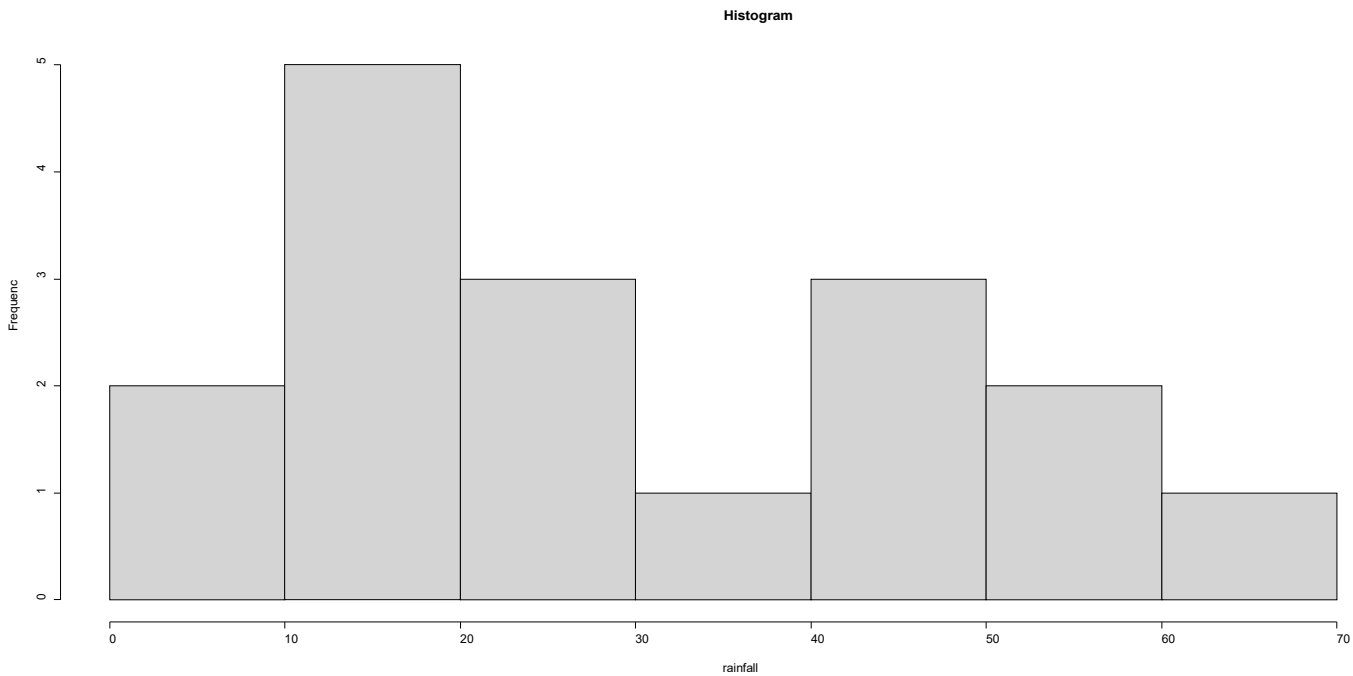
Example 3.2

We use the same data in Example 1. See what happens when we change the bins slightly (with the breaks argument to hist).

```
> hist(rainfall, breaks = 3, main = "Histogram ")
```



```
> hist(rainfall, breaks = 7, main = "Histogram")
```



3.2 Stemplots/Stem-and-leaf Plot

(1) Stemplots have two basic parts: *stems* and *leaves*.
(2) The final digit of the data values is taken to be a *leaf*, and the leading digit(s) is (are) taken to be *stems*.
(3) We draw a vertical line, and to the left of the line we list the stems. To the right of the line, we list the leaves beside their corresponding stem.
(4) There will typically be several leaves for each stem, in which case the leaves accumulate to the right. It is sometimes necessary to round the data values, especially for larger data sets.

<u>Example 3.3</u>
Consider the salary data of 33 primary school teachers.

```
> salary <- c(13,15, 26, 26, 27, 27,28, 29, 37, 37, 37, 38, 38, 39,
39, 39, 40, 41, 41, 42)

> stem(salary)
```

The decimal point is 1 digit(s) to the right of the |

```
 1 | 35
 2 | 667789
 3 | 77788999
 4 | 0112
```

3.3 Displaying Qualitative Data

Charts for a Single Categorical Variable

We often need to display a set of values each of which is associated with a single category of a factor or ordered factor. Most commonly the values are ***counts*** or ***proportions***.

Example 3.4

Consider the following data:

New Zealand Meat Consumption (1997)

Lamb	Mutton	Pigmeat	Poultry	Beef
8%	10%	16%	25%	41%

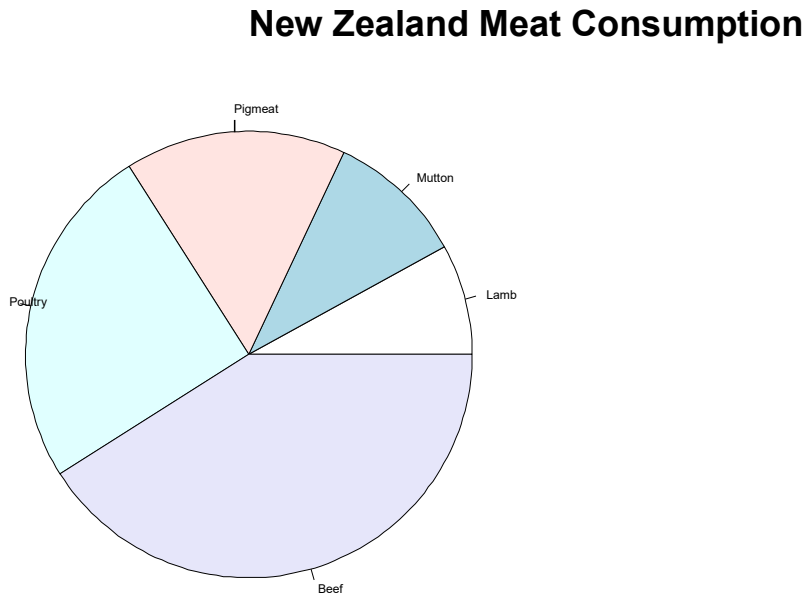
Remark: Proportions are often presented in a pie chart.

A basic pie chart is produced from a vector of named values. Such a vector can be created as follows:

```
> meat = c(8, 10, 16, 25, 41)
> names(meat) = c("Lamb", "Mutton", "Pigmeat","Poultry", "Beef")
```

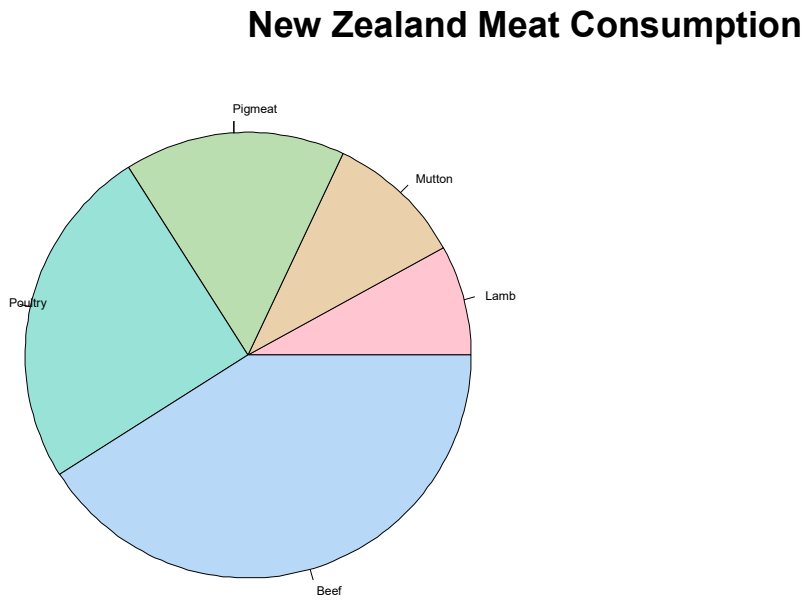
Once the data vector is created, the plot is easy to create.

```
> pie(meat, main = "New Zealand Meat Consumption", cex.main=3,
cex.lab=4)

# cex.main (font size for the title)
# cex.lab: Size of axis labels (the text describing the axis)
```



Different colors can be specified for the pie slices with a col= argument. For example:

```
> pie(meat,main = "New Zealand Meat Consumption",col = hcl(seq(0, 240,
by = 60)), cex.main=3, cex.lab=4)
```



3.4 Simple Bar Chart

(1) A bar chart is the analogue of a histogram for categorical data.
(2) A bar is displayed for each level of a factor, with the heights of the bars proportional to the frequencies of observations falling in the respective categories.
(3) A disadvantage of bar charts is that the levels are ordered alphabetically (by default), which may sometimes obscure patterns in the display.

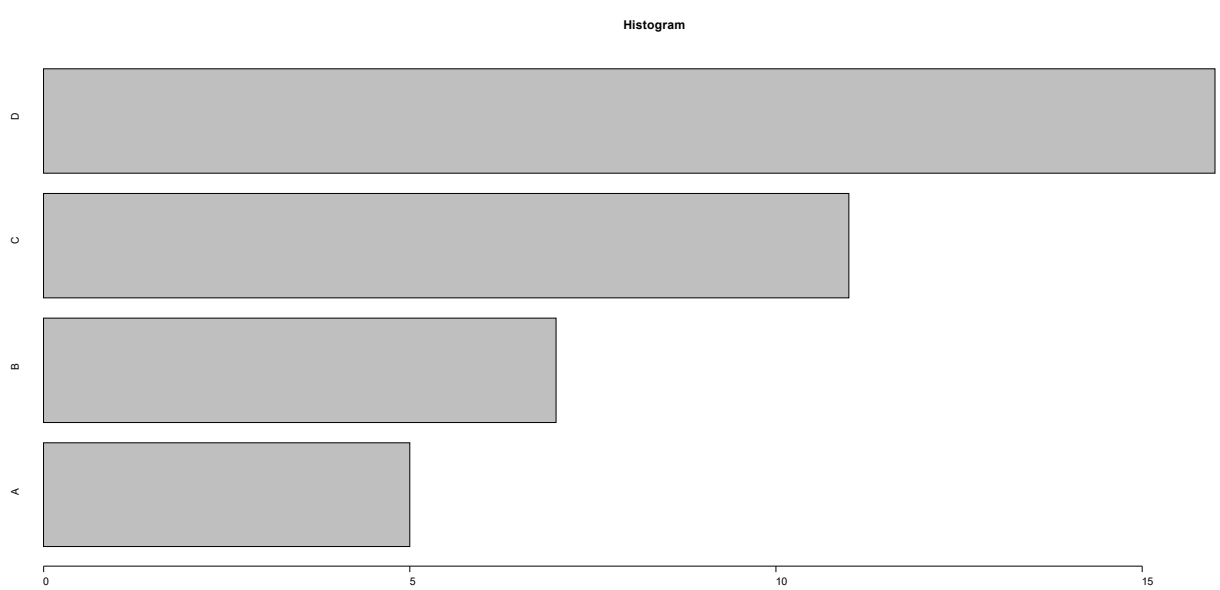
Create barplots with the **barplot(*height*)** function, where ***height*** is a **vector or matrix** and (optionally) a vector of labels for each bar. If the vector has names for the elements, the names will automatically be used as labels:

Example 3.5

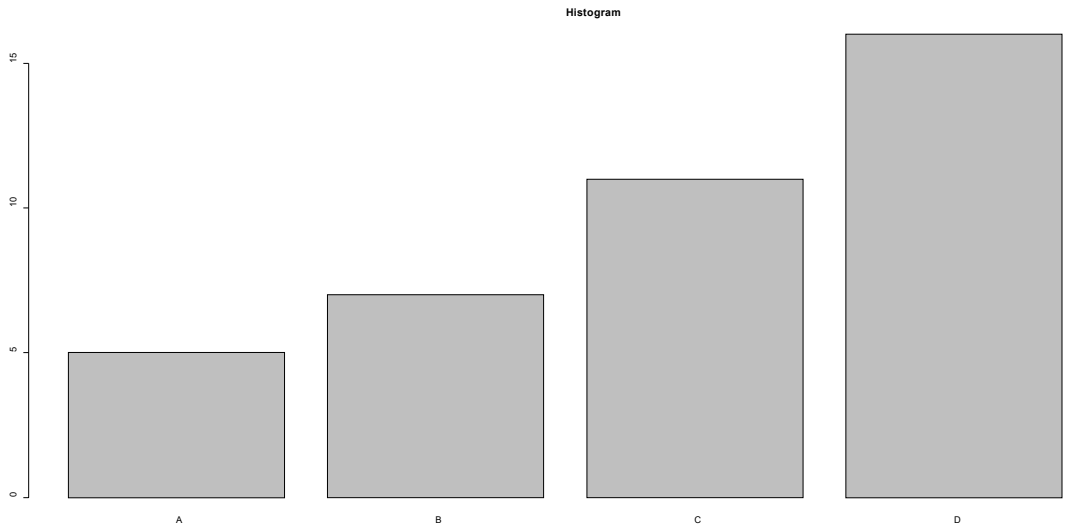
Horizontal Bars

It can be useful to draw the bars of bar-chart horizontally. In R, this is done by specifying horiz=TRUE.

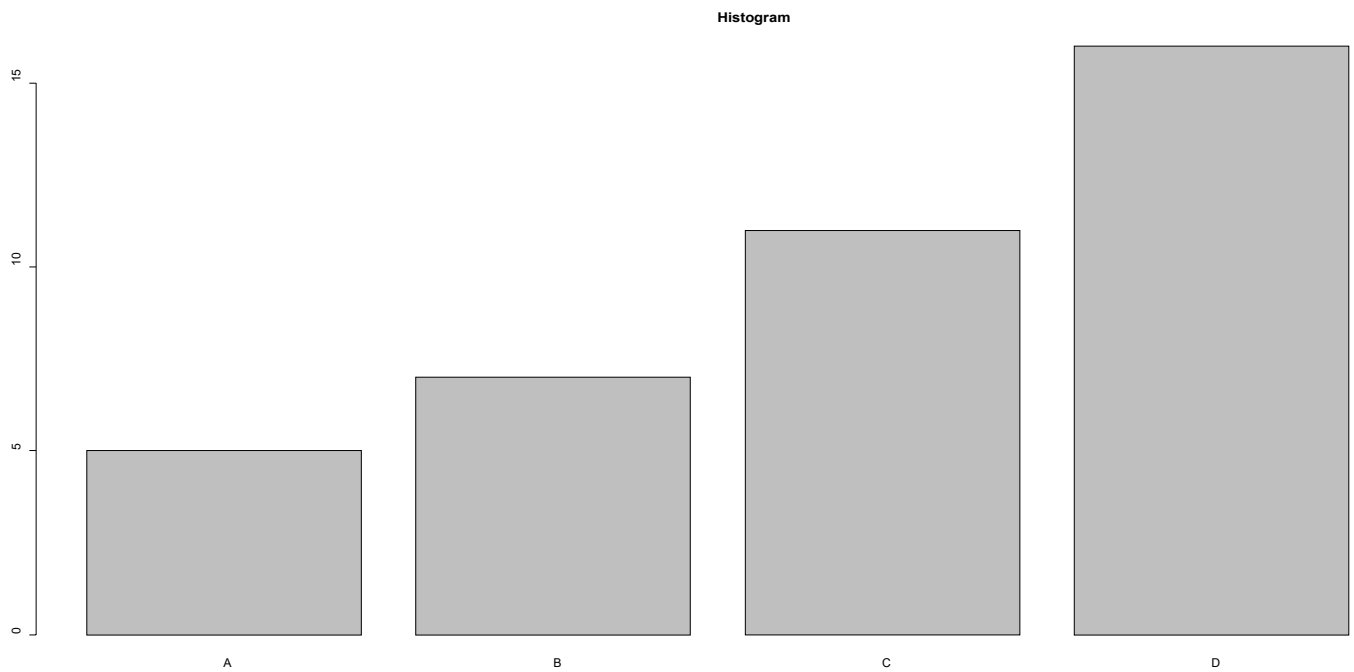
```
# Yearly sales
> count <- c(5, 7, 11, 16)

# 4 companies

> barplot(count, main="Histogram",horiz=TRUE, names.arg=c("A", "B", "C","D"))
```




```
> barplot(count, main="Histogram",horiz=FALSE, names.arg=c("A", "B", "C","D"))
```




```
# Without horiz=FALSE

> barplot(count, main="Histogram",names.arg=c("A", "B", "C","D"))
```



Example 3.6

Car_Brand	Quantity
Totota	5
BWM	7
BENX	10

The above Excel file is stored in:

```
D:\R Teaching Notes\EXCEL DATASETS\carbrand.xlsx
```

In the R Console, type the following command to install the **readxl** package:

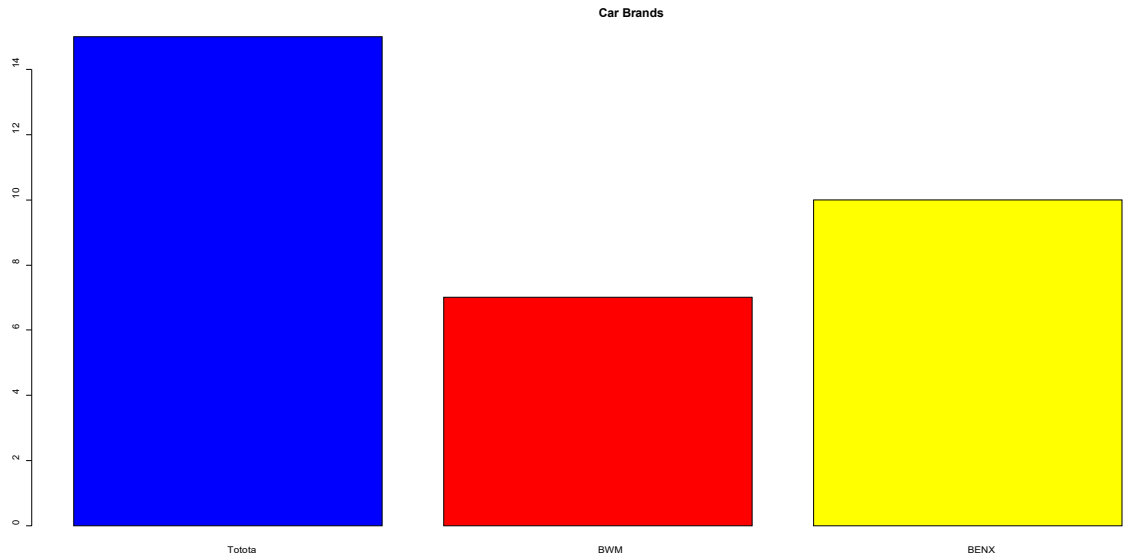
```
> install.packages("readxl")
```

In order to import your file, you'll need to apply the following template in the R Editor:

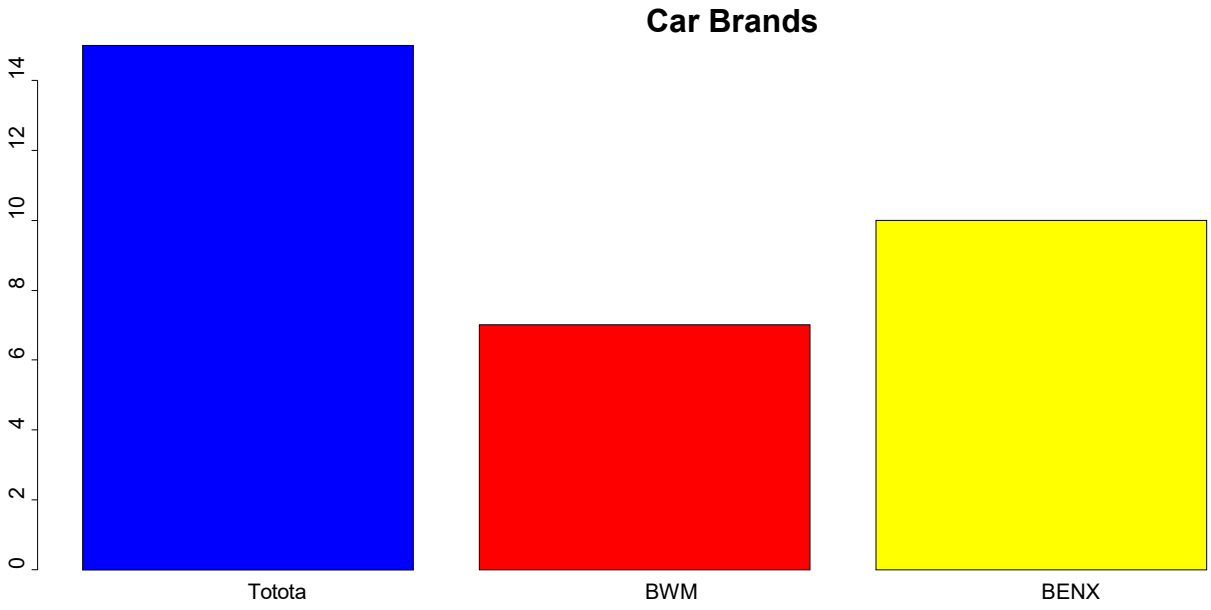
```
# the library is only available in R version 4.0.2

> library("readxl")

> brand <- read_excel("D:\\R Teaching Notes\\EXCEL
DATASETS\\carbrands.xlsx")
```

```
> brand
# A tibble: 3 x 2
  Car_Brand Quantity
  <chr>        <dbl>
1 Totota          15
2 BWM              7
3 BENX            10
```

```
> barplot(brand$Quantity, col = c("blue","red","yellow"),
names.arg=brand$Car_Brand, main="Car Brands")
```




```
# cex.axis (font size for the frequency scale on the y-axis)
# cex.names (font size for the names we gave to the columns)
# cex.main (font size for the title)

> barplot(brand$Quantity, col = c("blue","red","yellow"),
names.arg=brand$Car_Brand,
main="Car Brands", cex.axis = 2, cex.names=2, cex.main=3)
```



3.5 Two-category Bar Charts

A set of data is cross-classified by two factors, gender and alcoholic drinker.

% of New Zealand Population with Potentially Hazardous Drinking Pattern

Gender	15 - 24	25 - 44	45 - 64	65 or Above
Male	41	28	21	9
Female	26	9	4	0.5

It is also possible to input this data directly as a matrix.

```
> alcohol = matrix(c(41, 26, 28, 9, 21, 4, 9, 0.5), nrow = 2)
```

```
> dimnames(alcohol) = list(Sex = c("Male", "Female"), Age = c("15-24",
"25-44", "45-64", "65+"))
```

```
> alcohol
        Age
Sex      15-24 25-44 45-64 65+
  Male      41    28    21 9.0
  Female    26     9     4 0.5
```

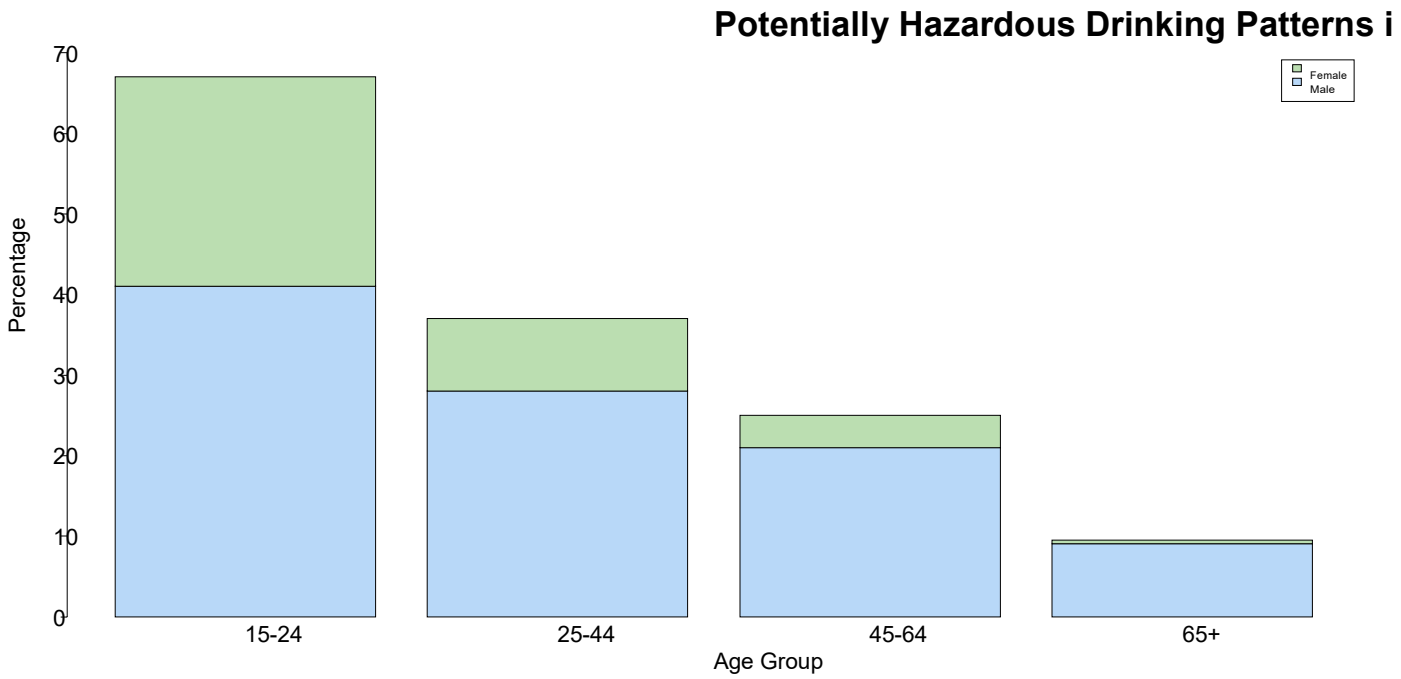
This is a 2x4 matrix with row and column labels.

3.5.1 Stacked Bar Chart

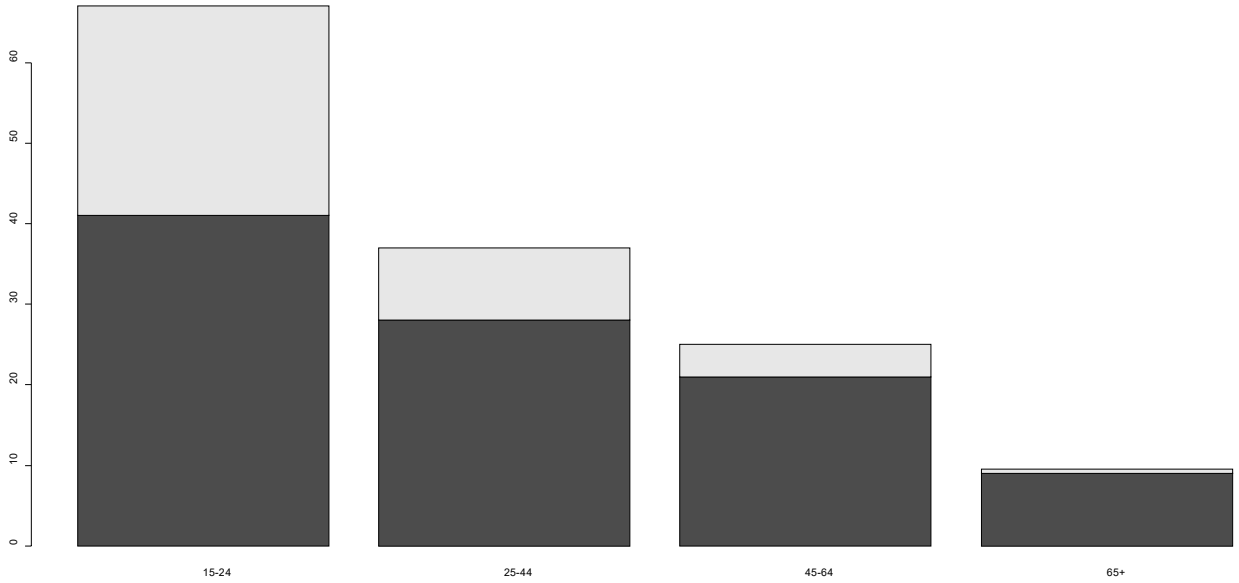
<u>Example 3.7</u>

We create a stacked bar chart for the alcohol data above.

```
> main = c("Potentially Hazardous Drinking Patterns in New Zealand
Population")
> barplot(alcohol,
legend = rownames(alcohol),
col = hcl(c(240, 120)),
ylim = c(0, 70),
las = 1,
main = main,
xlab = "Age Group",
ylab ="Percentage", cex.axis = 2, cex.names=2, cex.main=3, cex.lab=2)

# Note the use of las=1 here to rotate the y axis labels.

# cex.lab: Size of axis labels (the text describing the axis)
```




```
# Create a bar chart for alcohol data

> barplot(alcohol)
```

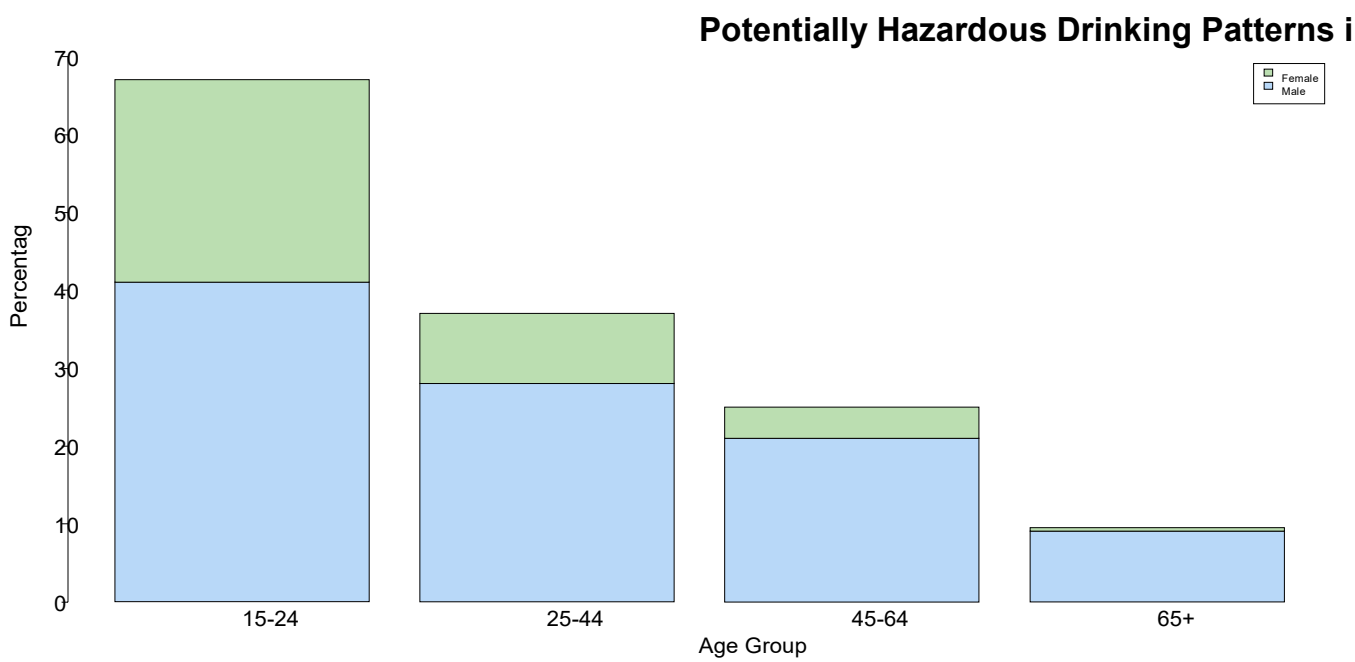


Improving the Default Layout

There are clear problems with a default bar plot:

- The colors are awful.
- The plot needs a *legend* explaining what the two colors mean.
- The plot needs an overall title and labels for the *x* and *y* axes.
- The tick mark labels should be horizontal.
- The tick marks should span the full height of the bars.

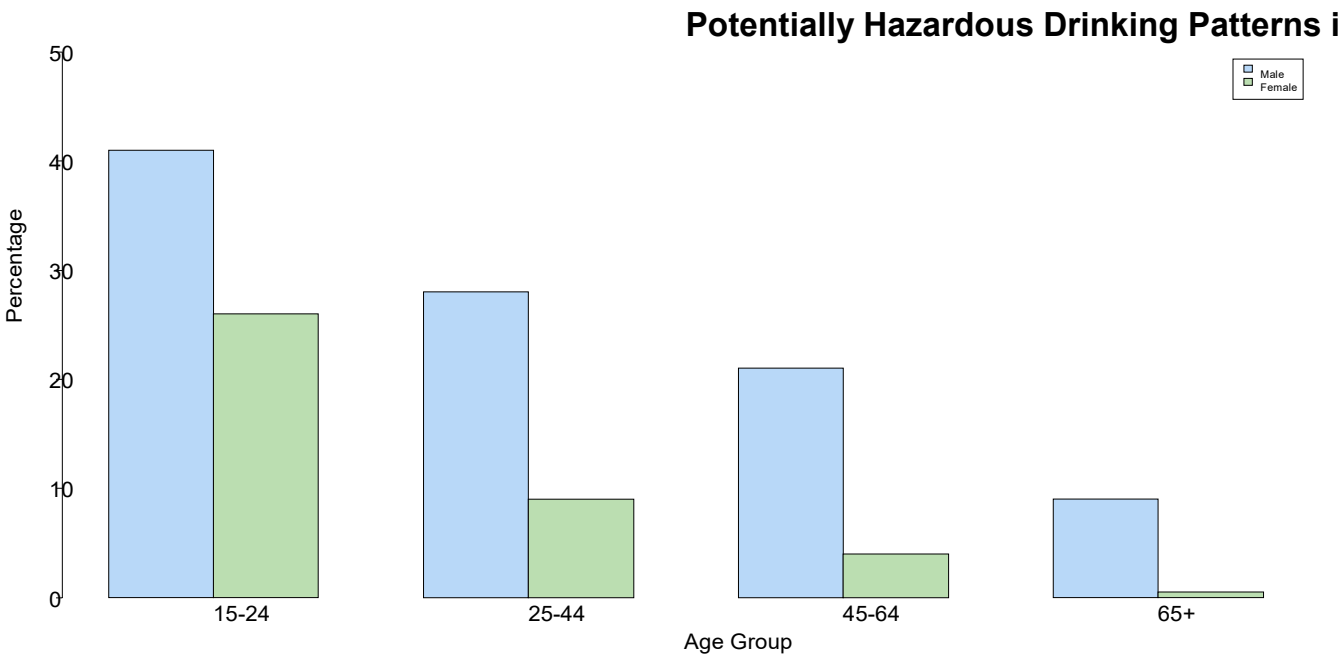
```
> main = c("Potentially Hazardous Drinking Patterns in New Zealand
Population")
> barplot(alcohol,
legend = rownames(alcohol),
col = hcl(c(240, 120)),
ylim = c(0, 70),
las = 1,
main = main,
xlab = "Age Group",
ylab ="Percentage", cex.axis = 2, cex.names=2, cex.main=3, cex.lab=2)
```



3.5.2 Side-by-Side Bars

Instead of drawing the "stacked" form of bar chart, it is also possible to produce a "side-by-side" form.

```
> barplot(alcohol, beside = TRUE,legend = rownames(alcohol),col =
hcl(c(240, 120)),
ylim = c(0, 50),las = 1,main = main,xlab = "Age Group",ylab
="Percentage", cex.axis = 2, cex.names=2, cex.main=3, cex.lab=2)
```



Remarks

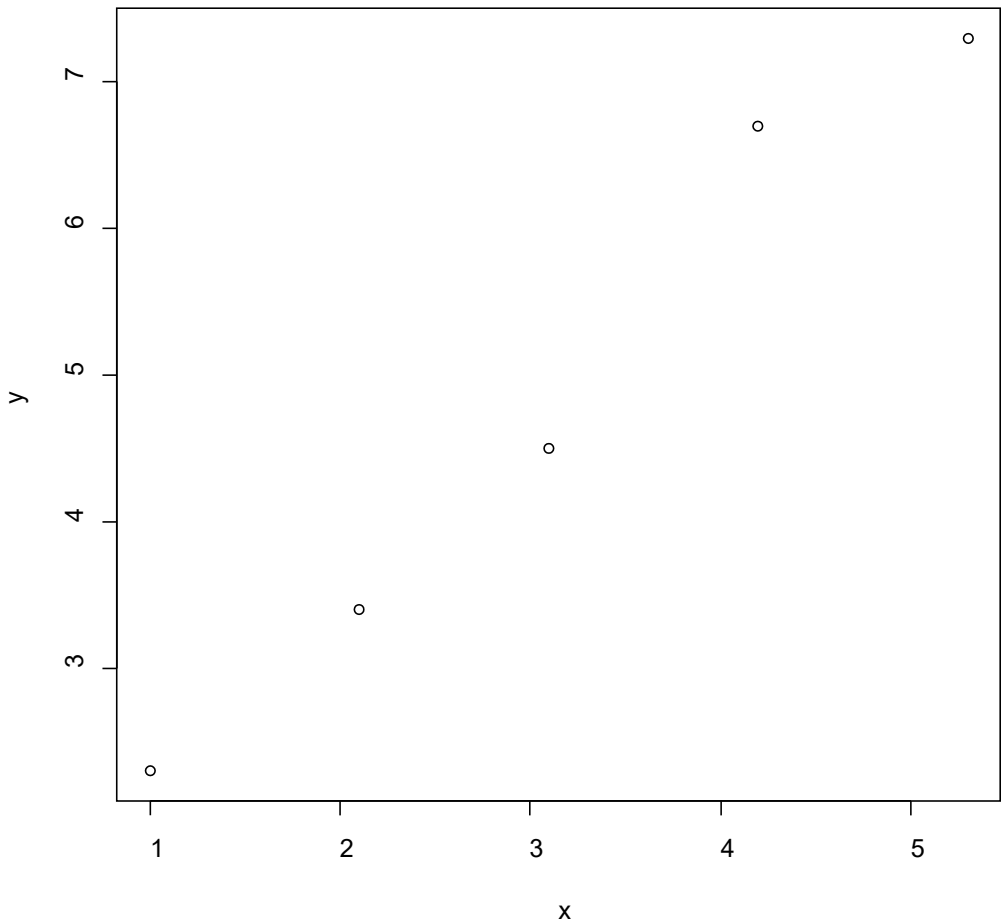
- Perceptual theory tells us that the side-by-side version will work best.
- Occasionally the stacked version is useful—this is when the combined height of the bars is meaningful.

3.6 Scatter Plot

To make a scatter plot below, use plot() and pass it a vector of *x* values followed by a vector of *y* values:

Example 3.8

```
> x <- c(1, 2.1, 3.1, 4.2, 5.3)
> y <- c(2.3, 3.4, 4.5, 6.7, 7.3)
> plot(x, y)
```



Example 3.9

Height	Weight
1.7	100
1.9	230
2.5	196
1.6	201
1.8	178
1.5	154
2.2	209
2.2	221
2.5	240
1.9	231

The above Excel file is stored in:

```
C:\Users\Tony\Desktop\hwdata.xlsx
```

In the R Console, type the following command to install the **readxl** package:

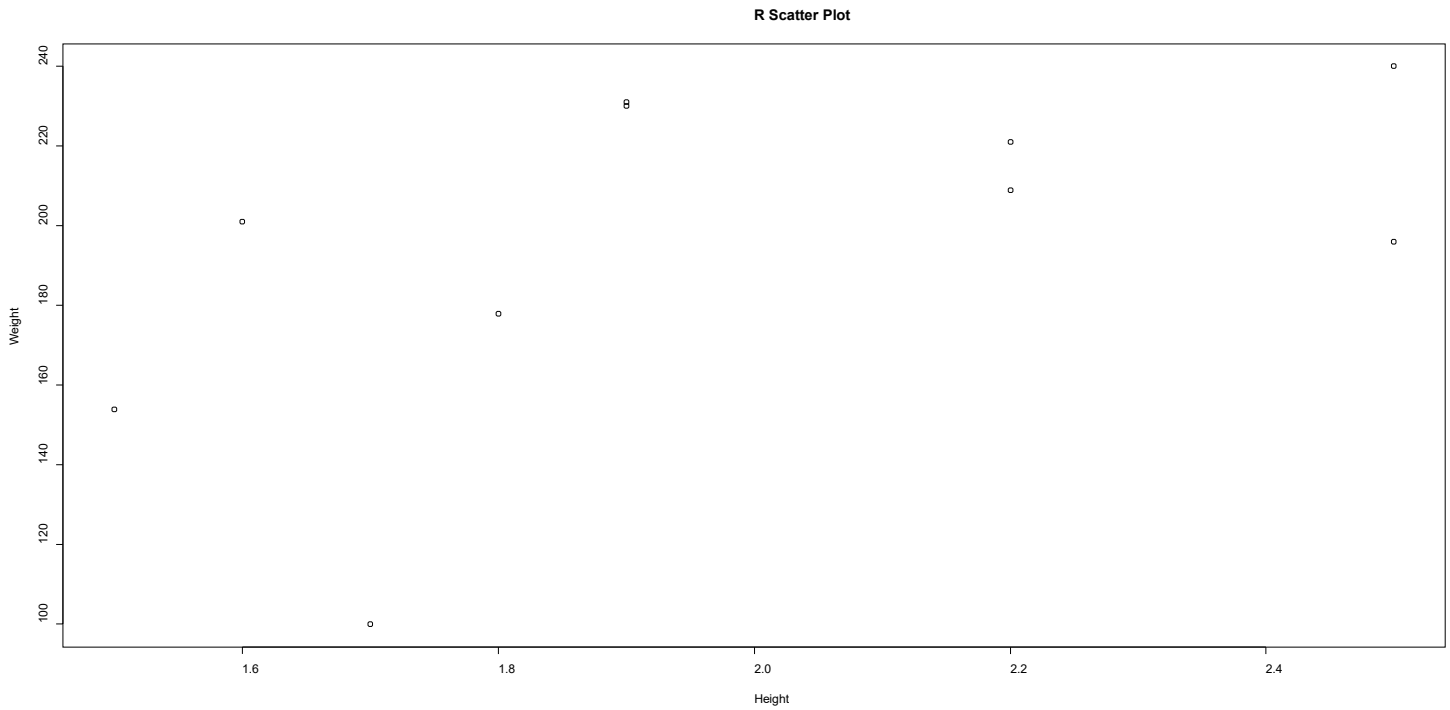
```
> install.packages("readxl")
```

In order to import your file, you'll need to apply the following template in the R Editor:

```
# the library is only available in R version 4.0.2
> library("readxl")

> Mydata <- read_excel("c:\\users\\tony\\desktop\\hwdata.xlsx")

> Mydata
```

```
# A tibble: 10 x 2
   Height  Weight
   <dbl>   <dbl>
 1  1.7     100
 2  1.9     230
 3  2.5     196
 4  1.6     201
 5  1.8     178
 6  1.5     154
 7  2.2     209
 8  2.2     221
 9  2.5     240
10  1.9     231
```

```
> plot(Mydata$Height, Mydata$Weight, xlab = "Height", ylab = "Weight",
main = "R Scatter Plot")

# plot(mydata$Height, Mydata$Weight) cannot work
# plot(Mydata$height, Mydata$weight) cannot work
# plot(mydata$height, mydata$weight) cannot work
```

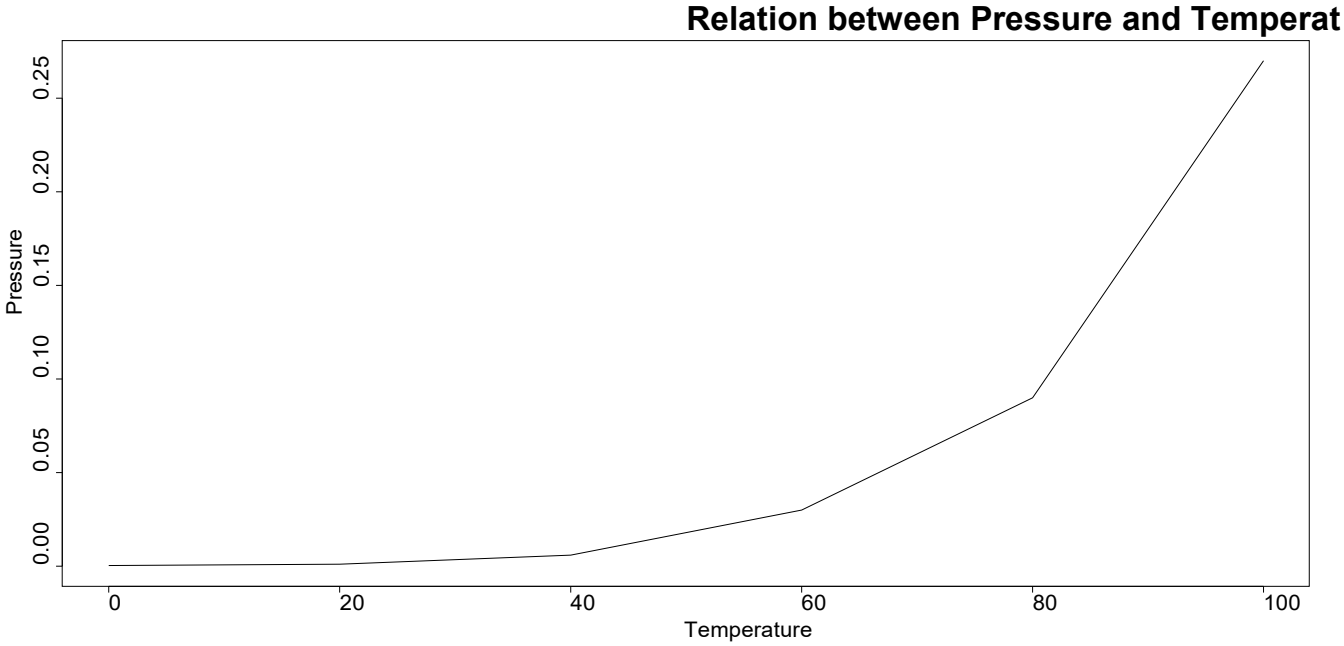


3.7 Line Chart

Line chart is used to look at the changes in variable over time or look at the relationship between two variables. In both cases, *x* axis corresponds to the independent variable (time, days, etc.), *y* axis corresponds to the dependent variable (temperature, income, etc.).

Let's consider the dataset with temperatures and pressures:

```
> temp <- c(0, 20, 40, 60, 80, 100)
> pressure <- c(0.0002, 0.0012, 0.0060, 0.0300, 0.0900, 0.2700)
> main <- c("Relation between Pressure and Temperature")

> plot(temp,pressure, type="l", xlab="Temperature",
    ylab="Pressure",main=main, cex.axis=2, cex.main=3, cex.lab=2)
```



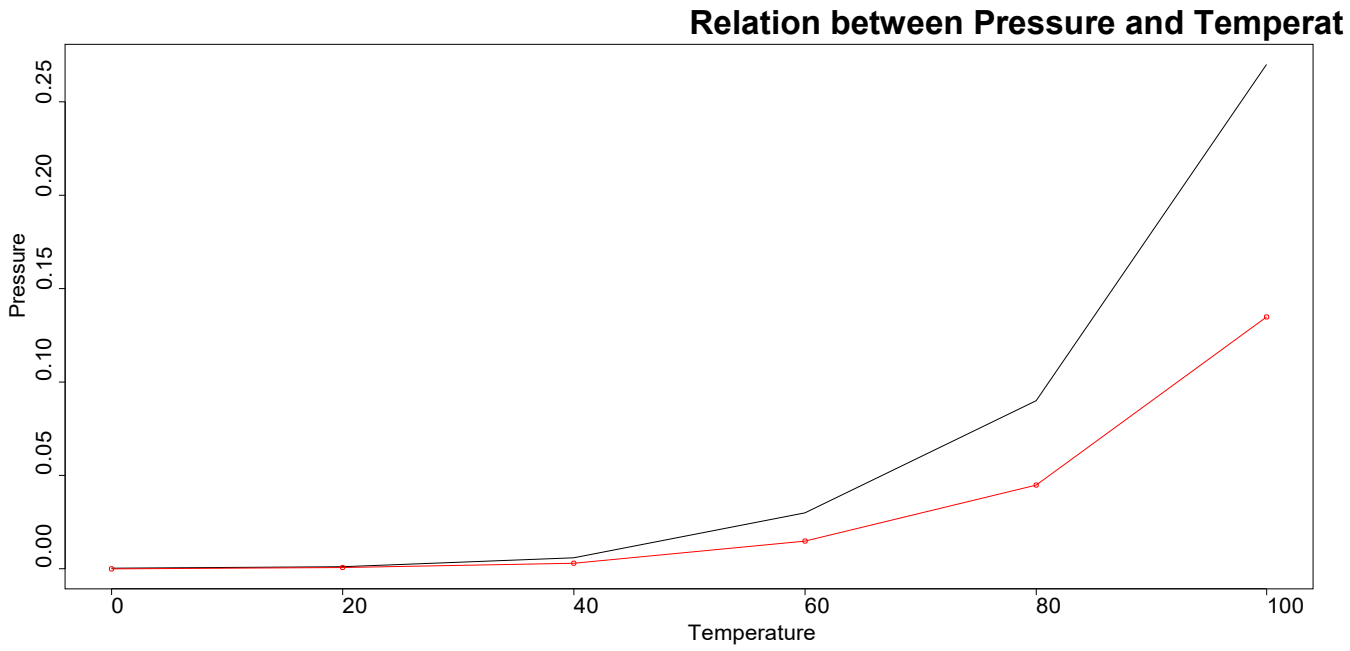
Remark

Drawing on a plot:

To add additional data, we use:

- points(x,y)
- lines(x,y)

```
# add blue points

points(temp, pressure/2, col=="blue")

# add red lines

lines(temp, pressure/2, col="red")

# add points to the lines

points(temp, pressure/2, col="red")

> plot(temp,pressure, type="l", xlab="Temperature",
ylab="Pressure",main=main, cex.axis=2, cex.main=3, cex.lab=2)
lines(temp, pressure/2, col="red")
points(temp, pressure/2, col="red")
```


Relation between Pressure and Temperat
Pressure
0.00
0.05
0.10
0.15
0.20
0.25
0
20
40
60
80
100
Temperature

Chapter 4 Hypothesis Testing

4.1 One-sample Test for Means – Z Test

With regards to **z-test**, there is NO z.test() function in the original R package, unfortunately. However, the package TeachingDemos contains a z.test() function which will be helpful. We therefore start with installing and loading TeachingDemos:

```
> install.packages("TeachingDemos")

> library("TeachingDemos")
```

$$HH_0: \mu\mu = \mu\mu_0 \leftrightarrow \begin{cases} HH_1: \mu\mu \neq \mu\mu_0 \\ HH_1: \mu\mu > \mu\mu_0 \\ HH_1: \mu\mu < \mu\mu_0 \end{cases}$$

Assumptions

(1) The population under study is normally distributed.

(2) The population variance ($\sigma\sigma^2$) or standard deviation ($\sigma\sigma$) is known.

Example 4.1

A student is interesting in estimating how many memes their professors know and love. So they go to class, and every time a professor uses a new meme, they write it down. After a year of classes, the student has recorded the following meme counts, where each count corresponds to a single class they took:

3, 7, 11, 0, 7, 0, 4, 5, 6, 2

The student talks to some other students who've done similar studies and determines that σ = 2 is a reasonable value for the standard deviation of this distribution. We want to test if $\mu\mu \neq 4.7$.

Assumption checking

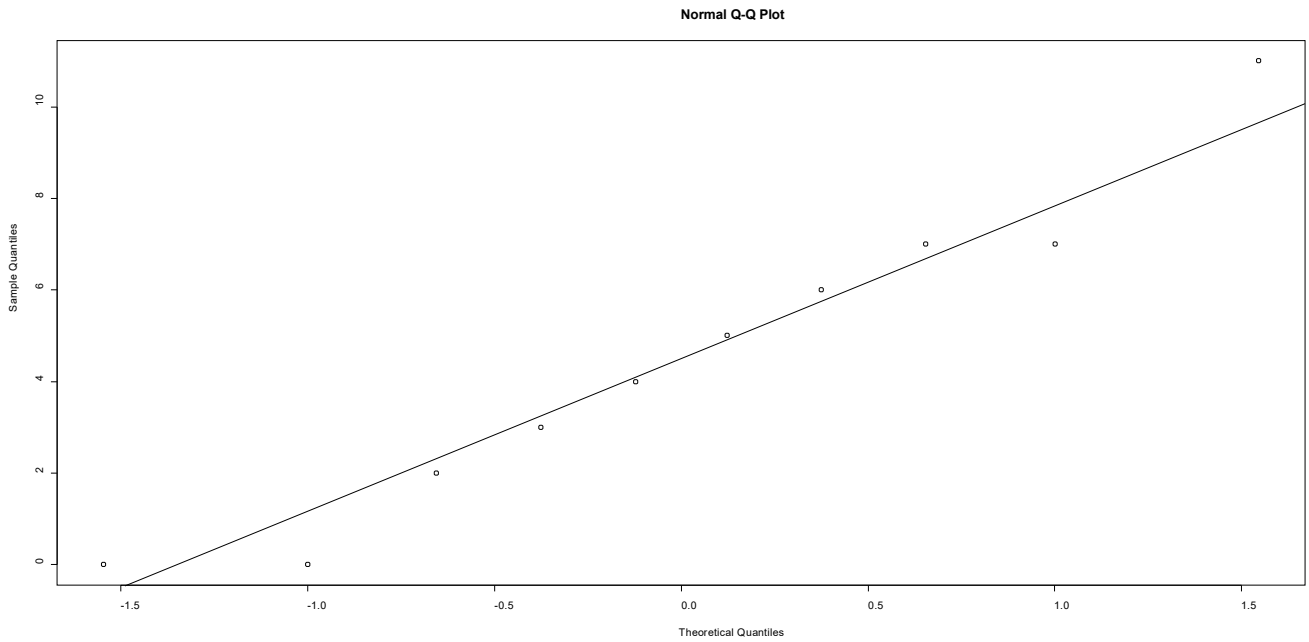
Before we can do a Z-test, we need to make check if we can reasonably treat the mean of this sample as normally distributed. This happens when one of the following hold:

1. The data comes from a normal distribution.
2. We have lots of data. How much? Many textbooks use 30 data points as a rule of thumb.

Since we have a small sample, we should check if the data comes from a normal distribution using a normal quantile-quantile plot.

```
# read in the data
> x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
```

```
# make the qqplot with a straight line at the same time
# run 2 commands simultaneously
> qqnorm(x)
  qqline(x)
```




```
> z.test(x, alternative="two.sided", mu=4.7, stdev=2)

        One Sample z-test

data:  x
z = -0.31623, n = 10.00000, Std. Dev. = 2.00000, Std. Dev. of the sample mean = 0.63246, p-value = 0.7518
alternative hypothesis: true mean is not equal to 4.7
95 percent confidence interval:
 3.26041 5.73959
sample estimates:
mean of x
      4.5
```

Conclusion:
Do not reject HH_0 at the 5% level and there is no evidence that the mean meme counts are different from 4.7.

Another author wrote a library "DescTools" for Z test, "ZTest()":

```
> install.packages("DescTools")
> library(DescTools)
> ZTest(x, alternative="two.sided", mu=4.7, sd_pop=2)
```

```
# Computer output:
```

```
> ZTest(x, alternative="two.sided", mu=4.7, sd_pop=2)

        One Sample z-test

data:  x
z = -0.31623, Std. Dev. Population = 2, p-value = 0.7518
alternative hypothesis: true mean is not equal to 4.7
95 percent confidence interval:
 3.26041 5.73959
sample estimates:
mean of x
      4.5
```

Remark

If you do NOT have the raw data for analysis but only the sample mean, $x\bar{x}$, you can use the hypothesis testing method below:

```
> install.packages("PASWR")
> library("PASWR")
> sample mean of the data is 4.5
> zsum.test(4.5, sigma.x = 2, n.x = 10, alternative="two.sided", mu = 4.7)
```

```
> zsum.test(4.5, sigma.x = 2, n.x = 10, alternative="two.sided", mu = 4.7)

        One-sample z-Test

data:  Summarized x
z = -0.31623, p-value = 0.7518
alternative hypothesis: true mean is not equal to 4.7
95 percent confidence interval:
 3.26041 5.73959
sample estimates:
mean of x
      4.5
```

Example 4.2

The mean breaking strength of a certain type of cord has been established from considerable experience at 18.5N with a standard deviation of 1.4N. A new machine is purchased to manufacture this type of cord. A sample of 120 pieces obtained from the new machine shows a mean breaking strength of 17.2N. Would you say that this sample is inferior on the basis of the 1% level of significance? Assume that breaking strength data are normally distributed.

Solution

HH_0: $\mu\mu = 18.5 \leftrightarrow HH_1$: $\mu\mu < 18.5$

```
> zsum.test(17.2, sigma.x = 1.4, n.x = 120, alternative="less", mu = 18.5)
```

```
> zsum.test(17.2, sigma.x = 1.4, n.x = 120, alternative="less", mu = 18.5)

        One-sample z-Test

data:  Summarized x
z = -10.172, p-value < 2.2e-16
alternative hypothesis: true mean is less than 18.5
95 percent confidence interval:
     -Inf 17.41022
sample estimates:
mean of x
     17.2
```

Conclusion: Reject HH_0 at the 1% level and there is sufficient evidence that the mean breaking strength is less than 18.5N.

4.2 Two-sample Test for Means – Z Test

Example 4.3

Suppose that the sample data from two different populations are as follows:

```
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
y <- c(5, 8, 10, 1, 6, 2, 5, 6, 7, 3)
```

The standard deviations of the two populations are equal to $\sigma\sigma_{xx} = 0.4$ and $\sigma\sigma_{yy} = 0.6$, respectively.

(a) Check if the data from the two populations are normally distributed.
(b) Is there any sufficient evidence that $\mu\mu_1 \neq \mu\mu_2$?

Solution

(a)

```
# Test for normality of data
# p-value > 0.05, data are normally distributed
# p-value < 0.05, data are not normally distributed

> shapiro.test(x)
```

```
> shapiro.test(x)

        Shapiro-Wilk normality test

data:  x
W = 0.95668, p-value = 0.7474
```

∵ p-value > 0.05,
∴ The data, x are normally distributed.

```
> shapiro.test(y)
```

```
> shapiro.test(y)

        Shapiro-Wilk normality test

data:  y
W = 0.97717, p-value = 0.9482
```

∵ p-value > 0.05,
∴ The data, y are normally distributed.

(b)

```
> install.packages("BSDA")
> library("BSDA")
> z.test(x, y, alternative="two.sided", sigma.x=0.4, sigma.y=0.6,
conf.level=0.95)
```

```
> z.test(x, y, alternative="two.sided", sigma.x=0.4, sigma.y=0.6, conf.level=0.95)

        Two-sample z-Test

data:  x and y
z = -3.5082, p-value = 0.0004511
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.2469406 -0.3530594
sample estimates:
mean of x mean of y
      4.5       5.3
```

Since the p-value is less than 0.05, therefore HH_0: $\mu\mu_1 = \mu\mu_2$ is rejected at the 5% level.

4.3 Hypothesis Tests for Mean Differences: Paired Data – t.test

Consider the paired data below that represent cholesterol levels on 10 men before and after taking a certain medication.

	Cholesterol Levels in mg/dL										mean	s^2	s
Before (x)	237	289	257	228	303	275	262	304	244	233	263.2	811.1	28.5
After (y)	194	240	230	186	265	222	242	281	240	212	231.2	864.0	29.4
$d = x - y$	43	49	27	42	38	53	20	23	4	21	32.0	238.0	15.4

t.test uage:

Test a claim about $\mu\mu_{dd} = \mu\mu_{xx} - \mu\mu_{yy}$ (the population mean difference)

Below, `mu` is the value of in HH_0.

(i) Two-tailed test R Command

```
> t.test(x, y, paired=TRUE, mu = ,)
```

(ii) Right-tailed test R Command

```
> t.test(x, y, paired=TRUE, mu= , alternative = "greater")
```

(iii) Left-tailed test R Command

```
> t.test(x, y, paired=TRUE, mu= , alternative = "less")
```

<u>Example 4.4</u>

Test the claim that, on average, the drug lowers cholesterol in all men. In other words, test the claim that $\mu\mu_{dd} > 0$. Test at $\alpha\alpha = 5\%$.

```
> before <- c(237, 289, 257, 228, 303, 275, 262, 304, 244, 233)
> after <- c(194, 240, 230, 186, 265, 222, 242, 281, 240, 212)
> t.test(before, after, paired=TRUE, mu= , alternative="greater")
```

```
        Paired t-test

data:  before and after
t = 6.5594, df = 9, p-value = 5.202e-05
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 23.05711      Inf
sample estimates:
mean of the differences
                     32
```

We can reject HH_0 at 5% level and support the claim because the P-value (0.00005202) is less than the 5% significance level.

Remarks

(1) we can extract the t-statistic or some other quantity of interest) from the output of the t.test function.

```
< ttest = t.test(dr1, dr2, paired=T, mu= , alternative="greater")
< names(ttest)
```

```
> ttest = t.test(dr1, dr2, paired=T, mu= , alternative="greater")
> names(ttest)
 [1] "statistic" "parameter" "p.value" "conf.int" "estimate" "null.value" "stderr" "alternative" "method" "data.name"
```

The value we want is named "statistic". To extract it, we can use the dollar sign notation, or double square brackets:

```
< ttest$p.value
```

```
> ttest$p.value
[1] 5.201946e-05
```

(2) If the data are stored in a text file, we can perform t test as follows:

```
< data <- read.table("D:\\R Teaching Notes\\chol.txt", header = F)
< dr1 <- data[,1]
< dr2 <- data[,2]
< t.test(dr1, dr2, paired=T, mu= , alternative="greater")
```

```
> data <- read.table("D:\\R Teaching Notes\\chol.txt", header = F)
> data
    V1  V2
1  237 194
2  289 240
3  257 230
4  228 186
5  303 265
6  275 222
7  262 242
8  304 281
9  244 240
10 233 212
> dr1 <- data[,1]
> dr2 <- data[,2]
> t.test(dr1, dr2, paired=T, mu= , alternative="greater")

        Paired t-test

data:  dr1 and dr2
t = 6.5594, df = 9, p-value = 5.202e-05
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 23.05711      Inf
sample estimates:
mean of the differences
                     32
```

4.4 Hypothesis Tests for Two Means: Independent Data – t.test

We test for a difference in means.

Consider the following data for cholesterol levels of 20 different men. Ten of them took the drug for a year and 10 of them did not.

	Cholesterol Levels in mg/dL										mean	s^2	s
No Drug (x_1)	237	289	257	228	303	275	262	304	244	233	263.2	811.1	28.5
Drug (x_2)	194	240	230	186	265	222	242	281	240	212	231.2	864.0	29.4

t.test usage:

Test about a claim about $\mu\mu_1 - \mu\mu_2$. We just need to remove `paired = TRUE`.

Below mu is the value of $\mu\mu_1 - \mu\mu_2$ in HH_0.

(i) Two-tailed test R Command

```
> t.test(x, y, mu = ,)
```

(ii) Right-tailed test R Command

```
> t.test(x, y, mu= , alternative = "greater")
```

(iii) Left-tailed test R Command

```
> t.test(x, y, mu= , alternative = "less")
```

Example 4.5

	Cholesterol Levels in mg/dL										mean	s^2	s
No Drug (x_1)	237	289	257	228	303	275	262	304	244	233	263.2	811.1	28.5
Drug (x_2)	194	240	230	186	265	222	242	281	240	212	231.2	864.0	29.4

Test the claim that the mean cholesterol level for all men who use the drug is less than the mean for those who do not use the drug. Assume that both populations are normally distributed and use a 5% significance level.

```
> no_drug <- c(237, 289, 257, 228, 303, 275, 262, 304, 244, 233)
> drug <- c(194, 240, 230, 186, 265, 222, 242, 281, 240, 212)
> t.test(no_drug, drug, mu=0, alternative="greater")
```

```
        Welch Two Sample t-test

data:  no_drug and drug
t = 2.4725, df = 17.982, p-value = 0.01181
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 9.556077      Inf
sample estimates:
mean of x mean of y
    263.2     231.2
```

We can reject HH_0 at 5% level and support the claim because the P-value (0.01181) is less than the 5% significance level.

Remark

Not specifying an alternate hypothesis defaults to a two-tailed test.

Example 4.6

Tony is teaching two sections of statistics, with 15 and 19 students, respectively. The grades on an exam are as follows.

Section 1: 100, 95, 90, 90, 90, 90, 85, 83, 80, 79, 71, 71, 70, 66, 48
Section 2: 100, 100, 100, 100, 98, 98, 98, 93, 93, 90, 86, 83, 81, 79, 79, 76, 61, 48, 41

One of the classes asks if they did significantly better or worse on the exam than the other class. Using $\alpha\alpha = 10\%$, what should Tony tell them?

Data input

```
> S1 <- c(100, 95, 90, 90, 90, 90, 85, 83, 80, 79, 71, 71, 70, 66, 48)

> S2 <- c(100, 100, 100, 100, 98, 98, 98, 93, 93, 90, 86, 83, 81, 79, 79,
76, 61, 48, 41)

# Perform two-sample t test
# We use a two-sided test since there was no initial inclination that
section 1 was superior or inferior to section 2.
# "var.equal = FALSE" means we do not assume that the two sets of
data have the same variance (same standard deviation).
# This is a technical issue in calculating the t-statistic; many
textbooks would call this assumption "not pooling the data".
# In any event, since the p-value is not less than alpha, we do not
reject the null hypothesis and conclude there's no significant
difference between the exam results in the two session.

> t.test(S1,S2,paired=FALSE, var.equal=FALSE)
```

Output:

```
> S1 <- c(100, 95, 90, 90, 90, 90, 85, 83, 80, 79, 71, 71, 70, 66, 48)
> S2 <- c(100, 100, 100, 100, 98, 98, 98, 93, 93, 90, 86, 83, 81, 79, 79, 76, 61, 48, 41)
> t.test(S1,S2,paired=FALSE, var.equal=FALSE)

        Welch Two Sample t-test

data:  S1 and S2
t = -0.72844, df = 31.977, p-value = 0.4716
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -14.759210   6.983771
sample estimates:
mean of x mean of y
 80.53333  84.42105
```

Conclusion

We do not reject HH_0 at 10% level as the p-value (0.4716) is greater than the significance level (0.1).

Remarks

(1) 'var.equal' automatically defaults to "FALSE", as these are the most conservative assumptions.

(2) When using "`t.test(S1,S2,paired=FALSE)`" without "`var.equal=FALSE`", the result is the same as "`t.test(S1,S2,paired=FALSE, var.equal=FALSE)`". That is, the default assumption is unequal variance.

Output:

```
> t.test(S1,S2,paired=FALSE)

        Welch Two Sample t-test

data:  S1 and S2
t = -0.72844, df = 31.977, p-value = 0.4716
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -14.759210   6.983771
sample estimates:
mean of x mean of y
 80.53333  84.42105
```

(3) If equal variance is assumed, the following R code can be used:

```
> t.test(S1,S2,paired=FALSE, var.equal=T)
```

Output:

```
> t.test(S1,S2,paired=FALSE, var.equal=T)

        Two Sample t-test

data:  S1 and S2
t = -0.70546, df = 32, p-value = 0.4856
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -15.113092   7.337654
sample estimates:
mean of x mean of y
 80.53333  84.42105
```

Conclusion

When equal variance is assumed, we still do not reject HH_0 at 10% level as the p-value (0.4856) is greater than the significance level (0.1).

Chapter 5 Linear Correlation & Simple Regression Analysis

5.1 Introduction

R organizes many linear models into the lm function:

- Regression Analysis (Unit 4)
- Analysis of Variance, ANOVA (Unit 5)

5.2 Specifying Model Formula

Functional Form	R Code	Remark
$yy = ff(xx) = \beta\beta_0 + \varepsilon\varepsilon$	$yy \sim 1$	A "1" represents the y-intercept
$yy = ff(xx) = \beta\beta_0 + \beta\beta_1 xx_1 + \varepsilon\varepsilon$	$yy \sim xx\,0000\,yy \sim xx + 1$	The y-intercept will be assumed if you omit it.
$yy = ff(xx) = \beta\beta_1 xx_1 + \varepsilon\varepsilon$	$yy \sim xx - 1$	But you can specify no y-intercept, i.e. $ff(0) = 0$.
$yy = ff(xx) = \beta\beta_0 + \beta\beta_1 xx_1 + \beta\beta_2 xx_2 + \varepsilon\varepsilon$	$yy \sim xx1 + xx2$	Include as many terms as you want.

<u>Example 5.1</u>

The weekly advertising expenditure (x) and weekly sales (y) are presented in the following table:

x	41	54	63	54	48	46	62	61	64	71
y	1250	1380	1425	1425	1450	1300	1400	1510	1575	1650

(a) Fit the trivial mean model: $yy = \beta\beta_{00} + \varepsilon\varepsilon$

```
> x = c(41, 54, 63, 54, 48, 46, 62, 61, 64, 71)
> y = c(1250, 1380, 1425, 1425, 1450, 1300, 1400, 1510, 1575, 1650)
```

> $ffffff = llll(yy{\sim}1)$

```
> summary(fit)
```

```
> x = c(41, 54, 63, 54, 48, 46, 62, 61, 64, 71)
> y=c(1250, 1380, 1425, 1425, 1450, 1300, 1400, 1510, 1575, 1650)
> fit=lm(y~1)
> summary(fit)

Call:
lm(formula = y ~ 1)

Residuals:
   Min     1Q Median     3Q    Max 
-186.5  -51.5  -11.5   58.5  213.5 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1436.50      37.79   38.01    3e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 119.5 on 9 degrees of freedom
```

$\therefore yy$ = 1436.50

```
> mean(y)
```

```
> mean(y)
[1] 1436.5
```

Confirmation: the fitted parameter value of $\beta\beta_0$, β is simply the mean of y values, y

(b) Fit the full model: $yy = \beta\beta_{00} + \beta\beta_{11}xx + \varepsilon\varepsilon$

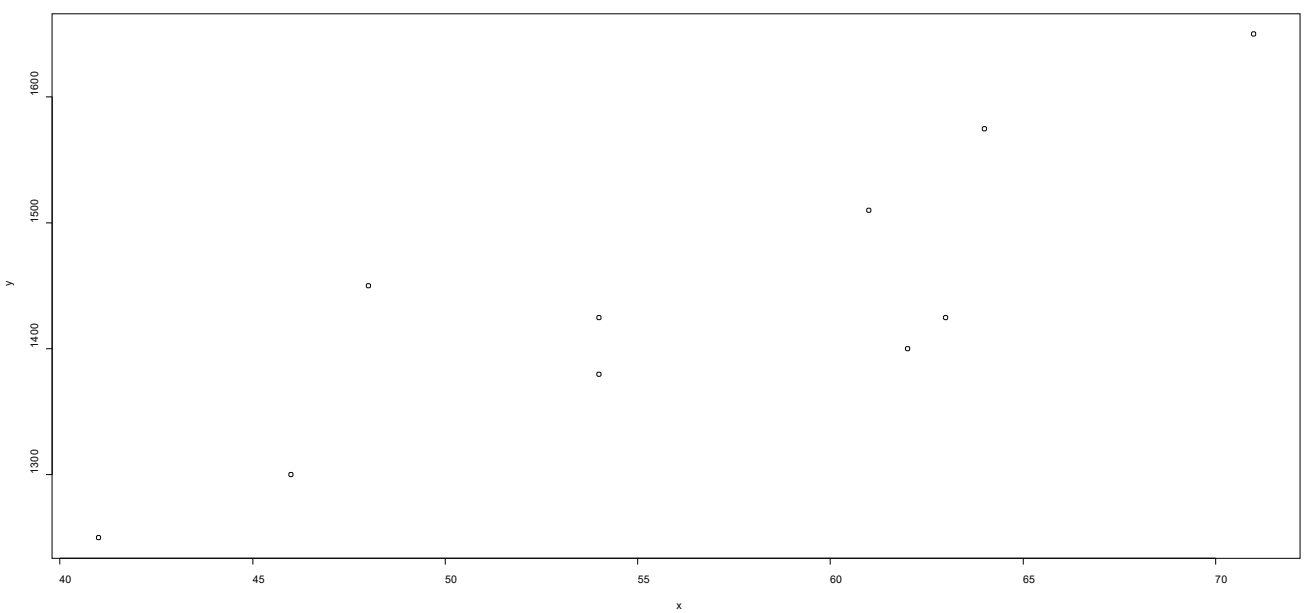
> $ffffff = llll(yy{\sim}xx)$

```
> summary(fit)
```

```
>  fit=lm(y~x)
>  summary(fit)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-96.906 -29.038  -2.998  47.980 104.109 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  828.127    136.129   6.083 0.000295 ***
x             10.787      2.384   4.525 0.001938 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 67.19 on 8 degrees of freedom
Multiple R-squared:  0.719,	Adjusted R-squared:  0.6839 
F-statistic: 20.47 on 1 and 8 DF,  p-value: 0.001938
```

Therefore, the fitted regression equation is yy = 828.127 + 10.787xx.

```
> plot(x,y)
```



A linear and positive relationship between x and y is apparent in the scatter plot above.

(c) Fit the model with y-intercept: $yy = \beta\beta_{11}xx + \varepsilon\varepsilon$

> $ffffff = llll(yy{\sim}xx - 1)$

```
> summary(fit)
```

```
> fit=lm(y~x-1)
> summary(fit)

Call:
lm(formula = y ~ x - 1)

Residuals:
     Min       1Q   Median       3Q      Max 
-157.062 -107.763    1.054  125.869  244.619 

Coefficients:
  Estimate Std. Error t value Pr(>|t|)    
x  25.1121     0.8322   30.18 2.36e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 150.3 on 9 degrees of freedom
Multiple R-squared:  0.9902,	Adjusted R-squared:  0.9891 
F-statistic: 910.6 on 1 and 9 DF,  p-value: 2.357e-10
```

Therefore, the fitted regression equation w/o y-intercept is yy = 25.1121xx.

5.3 Output of lm()

`fit$coef`: shows the list of best-fit coefficient(s)

```
> fit$coef
 (Intercept)           x 
   828.12689    10.78676 
```

`fit$resid`: shows a vector of residuals of all observations

```
> fit$resid
        1         2         3         4         5         6         7         8         9        10 
-20.38394 -30.61178 -82.69260  14.38822 104.10876 -24.31772 -96.90584  23.88092  56.52064  56.01334 
```

`fit$fitted`: shows the vector of predicted y values of all observations

```
> fit$fitted
        1         2         3         4         5         6         7         8         9        10 
1270.384 1410.612 1507.693 1410.612 1345.891 1324.318 1496.906 1486.119 1518.479 1593.987 
```

`summary(fit)`: shows the summary statistics

```
> summary(fit)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-96.906 -29.038  -2.998  47.980 104.109 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  828.127    136.129   6.083 0.000295 ***
x             10.787      2.384   4.525 0.001938 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 67.19 on 8 degrees of freedom
Multiple R-squared:  0.719,	Adjusted R-squared:  0.6839 
F-statistic: 20.47 on 1 and 8 DF,  p-value: 0.001938
```

`confint(fit)`: confidence interval(s) of the parameter(s)

```
> confint(fit)
                 2.5 %     97.5 %
(Intercept) 514.213779 1142.04000
x             5.289146   16.28437
```

`predict(fit, newdata= new.x)`: predicts the response of the hypothetical data points of x

```
> new.x <- data.frame(x = c(72, 73, 74))
> new.x
> predict(fit, newdata=new.x)
```

```
> new.x <- data.frame( x = c(72, 73, 74) )
> predict(fit, newdata=new.x)
       1        2        3 
1604.773 1615.560 1626.347 
> new.x
   x
1 72
2 73
3 74
> new.x <- data.frame(x = c(72, 73, 74))
> new.x
   x
1 72
2 73
3 74
> predict(fit, newdata=new.x)
       1        2        3 
1604.773 1615.560 1626.347 
```

Direct Way of Finding Correlation Between x and y

By default, the correlation function in R is as follows:

```
> cor(x, y = NULL, use = "everything", method = c("pearson",
"kendall", "spearman"))

# by default it uses Pearson method

# cor(variable1,variable2) cor(cars$mpg,cars$engine)
```

Example 5.2

The weekly advertising expenditure (x) and weekly sales (y) are presented in the following table:

x	41	54	63	54	48	46	62	61	64	71
y	1250	1380	1425	1425	1450	1300	1400	1510	1575	1650

```
> x = c(41, 54, 63, 54, 48, 46, 62, 61, 64, 71)
> y = c(1250, 1380, 1425, 1425, 1450, 1300, 1400, 1510, 1575, 1650)
> cor(x, y)
```

```
> x = c(41, 54, 63, 54, 48, 46, 62, 61, 64, 71)
> y = c(1250, 1380, 1425, 1425, 1450, 1300, 1400, 1510, 1575, 1650)
> cor(x, y)
[1] 0.84795
```

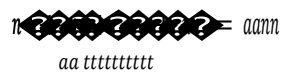
The value is the same as before: $r = \sqrt{R^2} = \sqrt{0.719} = 0.8479$

```
> summary(fit)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-96.906 -29.038  -2.998  47.980 104.109

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  828.127    136.129   6.083 0.000295 ***
x             10.787      2.384   4.525 0.001938 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 67.19 on 8 degrees of freedom
Multiple R-squared:  0.719,     Adjusted R-squared:  0.6839
F-statistic: 20.47 on 1 and 8 DF,  p-value: 0.001938
```

Chapter 6 Analysis of Variance (ANOVA)

Let a = the total number of treatment groups;
n = the number of observations per treatment;
N = the total number of observations/experimental units;
= $aa \times nn$.

Our goal is to compare $aa \geq 2$ treatments. As available resources we have N experimental units that we assign *randomly* to the a different treatment groups having n observations each, i.e. we have



This is a one-way analysis of variance. If all the treatment groups have the same number of experimental units, we call the design **balanced**.

<u>Example 6.1</u>
The tensile strengths (unit: Newton) of 3 different brands of electrical wires are shown below:

Brand 1	Brand 2	Brand 3
18.2	17.4	15.2
20.1	18.7	18.8
17.6	19.1	17.7
16.8	16.4	16.5
18.8	15.9	15.9
18.7	18.4	17.1
19.1	17.7	16.7

Test at 5% level if the mean tensile strengths of the 3 brands of wires are different.

Step 1: Form the vectors of observations by treatment group

```
> y1 = c(18.2, 20.1, 17.6, 16.8, 18.8, 18.7, 19.1)
> y2 = c(17.4, 18.7, 19.1, 16.4, 15.9, 18.4, 17.7)
> y3 = c(15.2, 18.8, 17.7, 16.5, 15.9, 17.1, 16.7)
```

Step 2: Combine yy_1, yy_2 *aannaa* yy_3 into ONE long vector,

```
> y = c(y1, y2, y3)
```

```
> y
 [1] 18.2 20.1 17.6 16.8 18.8 19.7 19.1 17.4 18.7 19.1 16.4 15.9 18.4 17.7 15.2 18.8 17.7 16.5 15.9 17.1 16.7
```

Step 3: Form a second vector, *group* which we can identify group membership

n = 7 observations per treatment group
a = 3 treatments

```
> n = rep(7,3)

> group = rep(1:3,n)
```

```
> n = rep(7,3)
> n
[1] 7 7 7
> group = rep(1:3,n)
> group
 [1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3
```

This means that:

(i) The 1st 7 observations belong to 1st treatment group, i.e. y1.
(ii) The 2nd 7 observations belong to 2nd treatment group, i.e. y2.
(iii) The 3rd 7 observations belong to 3rd treatment group, i.e. y3.

Step 4: Create a table of observations with observation identities (Group 1, Group 2, Group 3)

```
> data = data.frame(y = y, group = factor(group))
```

```
> data = data.frame(y = y, group = factor(group))
> data
      y group
1  18.2     1
2  20.1     1
3  17.6     1
4  16.8     1
5  18.8     1
6  19.7     1
7  19.1     1
8  17.4     2
9  18.7     2
10 19.1     2
11 16.4     2
12 15.9     2
13 18.4     2
14 17.7     2
15 15.2     3
16 18.8     3
17 17.7     3
18 16.5     3
19 15.9     3
20 17.1     3
21 16.7     3
```

Step 5: Generate the ANOVA table

```
> fit = lm(y ~ group, data)
> anova(fit)
```

```
> fit = lm(y ~ group, data)
> anova(fit)
Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value  Pr(>F)
group      2  9.2829  4.6414  3.5328 0.05079 .
Residuals 18 23.6486  1.3138
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$HH_0: \mu\mu_1 = \mu\mu_2 = \mu\mu_3 \quad \leftrightarrow \quad HH_1: nnooff\ HH_0$

where $\mu\mu_{ii} = ffhee\ ffffh\ ppoopppplaaffffoonn\ lleeaann\ ffeenttffllee\ ttffooeennssffh\ ooff\ wwffooee, ff = 1, 2, 3.$

Conclusion:

∵ p-value < 5%, we reject HH_0 at 5% level.

We conclude that the mean tensile strengths of the 3 brands of wires are significantly different.

<u>Alternative Function for One Way ANOVA: aov( )</u>

```
# Create a response variable vector, y, using the data from Brand 1,
Brand 2 and Brand 3, in order:

> y = c(18.2, 20.1, 17.6, 16.8, 18.8, 18.7, 19.1, 17.4, 18.7, 19.1,
16.4, 15.9, 18.4, 17.7, 15.2, 18.8, 17.7, 16.5, 15.9, 17.1, 16.7)

# Create an identity vector, group, for each obserevation of y

> group = c(1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,3)
```

Create a table of observations with observation identities (Group 1, Group 2, Group 3)

```
> mydata = data.frame(y = y, group = factor(group))

> results = aov(y ~ group, data=mydata)

> anova(results)
```

```
> fit = lm(y ~ group, data)
> anova(fit)
Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value  Pr(>F)
group      2  9.2829  4.6414  3.5328 0.05079 .
Residuals 18 23.6486  1.3138
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The same result is the same as before.

Chapter 7 Non-parametric Tests

7.1 Table of Parametric & Non-parametric Tests

Test	One-sample	Two Independent Samples	Paired Samples
Parametric	t test	Unpaired t test	Paired t test
Non-parametric	Wilcoxon signed rank test	Unpaired Wilcoxon rank sum test	Paired Wilcoxon signed rank test

7.2 One-sample Wilcoxon Signed Rank Test

Example 7.1

Consider the weights (kg) of 10 dogs below:

Dog1	Dog2	Dog3	Dog4	Dog5	Dog6	Dog7	Dog8	Dog9	Dog10
17.6	20.6	22.2	15.3	20.9	21.0	18.9	18.9	18.9	18.2

We want to know if the median weights of dogs differs from 25kg. Take $\alpha\alpha = 5\%$.

$$HH_0: \mu\mu = 25 \leftrightarrow \quad HH_1: \mu\mu \neq 25.$$

Input the data

```
> y=c(17.6, 20.6, 22.2, 15.3, 20.9, 21.0, 18.9, 18.9, 18.9, 18.2)
> result = wilcox.test(y, mu=25)
```

OR

```
> result = wilcox.test(y, mu=25, altenative="two.sided")
```

```
> result = wilcox.test(y, mu=25)
Warning message:
In wilcox.test.default(y, mu = 25) : cannot compute exact p-value with ties
> result

        Wilcoxon signed rank test with continuity correction

data:  y
V = 0, p-value = 0.005793
alternative hypothesis: true location is not equal to 25
```

The p-value is 0.005793 which is less than 5%.
We reject HH_0 at the 5% level and conclude that median weight of dogs is significantly different from 25kg.

Remark

A warning message, saying that "cannot compute exact p-value with ties". It comes from the assumption of a Wilcoxon test that the responses are continuous. We can suppress this message by adding another argument `exact=FALSE`, but the result will be the same.

7.3 Wilcoxon Signed Rank Test for Paired/Dependent Samples

Using the Wilcoxon signed rank test, we can decide whether 2 data population distributions are identical ***without*** assuming them to follow the normal distribution.

Paired Samples

Two data samples are ***matched (paired or dependent)*** if they come from ***repeated observations of the same subjects***. Otherwise, they are independent samples.

Example 7.2

A data set contains the weights (grams) of 10 chickens before and after feeding a certain feed, as shown below:

Weight Before	200.1	190.9	192.7	213.0	241.4	196.9	172.2	185.5	205.2	193.7
Weight After	392.9	393.2	345.1	393.0	434.0	427.9	422.0	383.9	392.3	352.2

Test if there is any significant difference in the median weights before and after taking the feed at the 5% level of significance.

$$H_0: \mu_{?} = \mu_{?} \quad \leftrightarrow \quad H_1: \mu_{?} \neq \mu_{?}$$

```
# Create 2 vectors: before-weight and after-weight

> before=c(200.1,190.9,192.7,213,241.4,196.9,172.2,185.5,205.2,193.7)

> after=c(392.9,393.2,345.1,393,434,427.9,422,383.9,392.3,352.2)

# Create a data frame

> mydata=data.frame(group=rep(c("before","after"),each=10),weight=c(before,after))
```

Result

```
> before=c(200.1,190.9,192.7,213,241.4,196.9,172.2,185.5,205.2,193.7)
> after=c(392.9,393.2,345.1,393,434,427.9,422,383.9,392.3,352.2)
> mydata=data.frame(group=rep(c("before","after"),each=10),weight=c(before,after))
> mydata
    group weight
1  before  200.1
2  before  190.9
3  before  192.7
4  before  213.0
5  before  241.4
6  before  196.9
7  before  172.2
8  before  185.5
9  before  205.2
10 before  193.7
11  after  392.9
12  after  393.2
13  after  345.1
14  after  393.0
15  after  434.0
16  after  427.9
17  after  422.0
18  after  383.9
19  after  392.3
20  after  352.2
```

```
# Perform paired Wilcoxon signed rank test - Method 1
# The data are saved in two different numeric vectors.

> result = wilcox.test(before, after, paired=TRUE)

> result


# Computer output
```

```
> result = wilcox.test(before, after, paired=TRUE)
> result

        Wilcoxon signed rank exact test

data:  before and after
V = 0, p-value = 0.001953
alternative hypothesis: true location shift is not equal to 0
```

The p-value is 0.001953 which is less than 5%. We reject HH_0 at the 5% level.
We conclude that the median weight of the mice before and after taking the feed is significantly different.

```
# Perform paired Wilcoxon signed rank test - Method 2
# The data are saved in a data frame

> result <- wilcox.test(weight ~ group, data=mydata, paired=TRUE)

> result

# Computer output
```

```
> result <- wilcox.test(weight ~ group, data=mydata, paired=TRUE)
> result

        Wilcoxon signed rank exact test

data:  weight by group
V = 55, p-value = 0.001953
alternative hypothesis: true location shift is not equal to 0
```

The result is the same as that obtained in Method 1.

Remarks

(1) If we want to test whether the median weight before treatment is **less** than the median weight after treatment, use the following code:

```
> wilcox.test(weight~group,data=mydata,paired=TRUE,alternative="less")
```

(2) If we want to test whether the median weight before treatment is **greater** than the median weight after treatment, use the following code:

```
> wilcox.test(weight~group,data=mydata,paired=TRUE,alternative="greater")
```

7.4 Wilcoxon Signed Rank Test for Unpaired/Independent Samples

Example 7.3

A data set contains the weights of 18 individuals (9 women and 9 men), as shown in the table below:

Group	Weight	Gender
1	38.9	Woman
2	61.2	Woman
3	73.3	Woman
4	21.8	Woman
5	63.4	Woman
6	64.6	Woman
7	48.4	Woman
8	48.8	Woman
9	48.5	Woman
10	67.8	Man
11	60.0	Man
12	63.4	Man
13	76.0	Man
14	89.4	Man
15	73.3	Man
16	67.3	Man
17	61.3	Man
18	62.4	Man

Test H_0: $\mu_{?} = \mu_{?}$ $\leftrightarrow$ H_1: $\mu_{?} \neq \mu_{?}$ at 5% level.

Construct Women weight vector

```
> Wweight = c(38.9, 61.2, 73.3, 21.8, 63.4, 64.6, 48.4, 48.8, 48.5)
```

Construct Men weight vector

```
> Mweight = c(67.8, 60, 63.4, 76, 89.4, 73.3, 67.3, 61.3, 62.4)
# Create a data frame
> mydata = data.frame(weight = c(Wweight, Mweight),group = rep(c("Woman", "Man"),
each = 9))
```

Output

```
> mydata = data.frame(weight = c(Wweight, Mweight),group = rep(c("Woman", "Man"), each = 9))
> mydata
   weight group
1    38.9 Woman
2    61.2 Woman
3    73.3 Woman
4    21.8 Woman
5    63.4 Woman
6    64.6 Woman
7    48.4 Woman
8    48.8 Woman
9    48.5 Woman
10   67.8   Man
11   60.0   Man
12   63.4   Man
13   76.0   Man
14   89.4   Man
15   73.3   Man
16   67.3   Man
17   61.3   Man
18   62.4   Man
```

```
# Compute unpaired two-samples Wilcoxon signed rank test

> result = wilcox.test(Wweight, Mweight, alternative="two.sided")

> result
```

Computer output

```
> result = wilcox.test(Wweight, Mweight, alternative="two.sided")
Warning message:
In wilcox.test.default(Wweight, Mweight, alternative = "two.sided") :
  cannot compute exact p-value with ties
> result

        Wilcoxon rank sum test with continuity correction

data:  Wweight and Mweight
W = 15, p-value = 0.02712
alternative hypothesis: true location shift is not equal to 0
```

The p-value is 0.02712 which is less than 5%. We reject HH_0 at the 5% level.
We conclude that men's median weight is significantly different from women's median weight.