MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

SIMON KUZNETS KHARKIV NATIONAL UNIVERSITY OF ECONOMICS

PROBABILITY THEORY AND MATHEMATICAL STATISTICS

Guidelines to laboratory work based on the R software

for Bachelor's (first) degree students of all specialities

Kharkiv
S. Kuznets KhNUE
2018

UDC 519.2(07.034)
P93

Compiled by: L. Malyarets
O. Tyzhnenko

Затверджено на засіданні кафедри вищої математики і економіко-математичних методів.
Протокол № 13 від 07.03.2018 р.

Самостійне електронне текстове мережеве видання

Probability Theory and Mathematical Statistics : guidelines to P93 laboratory work based on the R software for Bachelor's (first) degree students of all specialities [Electronic resource] / compiled by L. Malyarets, O. Tyzhnenko. – Kharkiv : S. Kuznets KhNUE, 2018. – 114 p. (English)

The methodology for studying the probability and statistics problems with the help of R programming by solving textbook problems has been studied. Some basic principles of using R for solving probability and statistics problems have been introduced based on real examples. This approach will help students having basic knowledge of the calculus and mathematical methods in economics to quickly start using R for mathematical calculations. So, the guidelines can be used as a background for students of economics studying probability and statistics courses with the use of the computer program R.

For Bachelor's (first) degree students of all specialities.

UDC 519.2(07.034)

Contents

Introduction

The objective of this methodical edition is to test the statistical software R for possible educational use at S. Kuznets Kharkiv National University of Economics for solving statistical textbook problems. This work introduces some basic features of R for statistics with some real examples. They will help students of economics having basic knowledge of mathematics start using R in probability and statistics courses. It can be also used as a tutorial for engineering students when studying statistics courses with computer program in R.

R is a popular language and environment that allows powerful and fast manipulation of data, offering many statistical and graphical options. Graphical representation of data is pivotal when one wants to present scientific results, in particular in publications. R allows you to build top quality graphs (much better than Excel for example).

These guidelines, however, focus on the statistical possibilities of R. Whatever package you use, you need some basic statistical knowledge if only you want to design your experiments correctly.

This work is mainly divided into three parts. Part 1 will tell you where to get the software and how to install it on your PC. The purpose of Part 2 is to give some familiarity with the R sessions and R language essentials. Part 3 is devoted to the probability distributions with R. In the rest part of this work, some statistical textbook problems are solved with the R statistical software. With the help of these examples, you will further familiarize yourself with the application of R to solving the real statistics problems.

These guidelines were written to follow the version 3.2.5(2016-04-14) that is one of the latest versions of R.

1. Obtaining and installing R

One way to download R is from the main website for R: http://cran.r-project.org/. There are lots of mirror sites worldwide. You can choose a closer site to get the faster download time. There are three pre-complied versions for Linux, Mac Os and Windows and you can select a proper version for variants of platforms. The most convenient way to use R is at a graphics workstation running a windowing system.

The binaries distribution installation is usually quite straightforward and is similar to other software. The binaries distribution can be obtained in two versions:

1. A 23Mb file *rw2001.exe*. Just run this for a Windows-XP style installer. It contains all the R components, and you can select what you want to install.
2. Files *miniR.exe* and *miniR-1.bin* to *miniR-7.bin*. This is a small installation, containing text help and the introduction to R and Data Import/Export manuals in PDF.

For more details, including command-line options for the installers and how to uninstall, see the *rw-FAQ* from CRAN.

For Microsoft Windows platform, select the set up file *rw2001.exe* and double-click with the mouse and then follow the on-screen instructions. When the process is complete, you will have an entry under Programs on the start menu for invoking R, as well as a desktop icon.

The *README.rw2001* offers the detailed instructions on installation for your machine.

2. R language essentials (quick introduction to R)

2.1. Starting the R session and session management

Starting R is straightforward, but the method will depend on your computing platform. You can launch R from a system menu by a double-click on the icon, or by input of the command "R" in the system command line. Once R is started, you will see the information as: R : Copyright 2004, The R Foundation for Statistical Computing, Version 2.0.1 (2004-11-15), ISBN 3-900051-07-0.

R is free software and comes with absolutely no warranty. You are welcome to redistribute it under certain conditions. Type *license*() for distribution details.

R is a collaborative project with many contributors. Type *contributors*() for more information and *citation*() on how to cite R or R packages in publications.

Type *demo*() for some demos, *help*() for on-line help, or *help.start*() for a HTML browser interface to help.

Type *q*() to quit R.

The R program prints a prompt ">" when it is ready for input. R works fundamentally by the question-and-answer model: you can enter the command then press the enter button, the program will do something, then print the result if relevant. If a command is not complete at the end of a line, R will give a different prompt with "+" to expect to read the input until the command is syntactically complete.

R is an expression language with a very simple syntax. It is case sensitive, so the letter "A" and "a" is different symbols and refers to different variables. Normally all alphanumeric symbols are allowed.

Comments can be put almost anywhere, following with a hash mark "#". Any comment character after the "#" is ignored by R. Normally parentheses "()" are for functions, and square brackets "[]" are for vector arrays and lists.

All variables created in R, are stored in a common workspace. The function *ls* (***l*****i*****s***t) is used to display the contents of the workspace. You can also use the function *rm* (***r***e***m***ove) to delete some of the objects from your workspace. To clear the entire workspace you can use the command:

```
> rm(list = ls()).
```

When you exit, you will be asked whether to save the workspace image. It is also possible to save the workspace to a file with any name using the command:

```
> save.image().
```

It will be saved to a file with *.Rdata* extension in your working directory. The files with *.Rdata* extension will be loaded by default when R is started in its directory. Other saved files can be loaded into your workspace using the function *load*().

2.2. Getting help with functions and features

R has an inbuilt help facility. From the command line, you can always use the following commands to get the information on any specific named function.

To get help on the *solve*() function:

```
> help(solve)
```

or

```
> ?solve.
```

Another command we usually use to get help is *apropos*(). This command is convenient when you think you know the function's name but you are not sure. We can use this command to get a list of function names that contain a given pattern.

```
> apropos("solve")
[1] "backsolve"      "forwardsolve"   "qr.solve"       "solve"
[5] "solve.default" "solve.qr".
```

On most R installations the help is available in HTML format by running

```
> help.start()
```

which will launch a Web browser that allows the help pages to be browsed with hyperlinks. The "Search Engine and Keywords" link in the page loaded by *help.start*() is particularly useful as it contains a high-level concept list which searches though available functions. It can be a great way to get your bearings quickly and to understand the breadth of what R has to offer.

2.3. Entering data into R

An R installation contains a library of packages. Some of these packages are part of the basic installation, others can be downloaded from the website: http://cran.r-project.org/. To load the package into R we should use the command *library*(). The loaded packages will be dropped if you terminate your R session. So you have to load it again when you start a new session with the saved workspace.

R has a number of built-in data sets. Sometimes we need to read in a built-in dataset. But at the first we need to load the package, and then ask to load the data. Here are the commands used for reading in a built-in dataset:

- use the command *library*() to list all available packages;
- use the command *data*() without any arguments to list all available datasets;
- use *data*(package = "package name") to list all data sets in a given package;
- use *data*("dataset name") to read in a dataset.

It is very convenient to use built-in data sets, sometimes we want to enter data into the session from outside of R. There are several ways to read data from outside.

2.3.1. Using *c*()

The most useful R command for quickly entering in small data sets is the *c*-function. It is short for "concatenate". This function combines, or concatenates terms together, for example, stores the values 1, 2, 3, 4 into *x*.

```
> x = c(1,2,3,4)
> x
[1] 1 2 3 4.
```

The values are assigned to the variable *x* by the assignment operator "=". The value of *x* doesn't automatically print out. We can input the variable name to indicate the values. The values are prefaced with "[1]". This indicates that the value is a vector.

2.3.2. Using scan

The function *scan*() can do the same thing as *c*():

```
> x = scan()
1 2 3
4.
```

Notice, when we start typing the numbers in, if we hit the return key once, we continue on a new row, if we hit it twice, *scan*() stops.

2.3.3. Using *scan*() with a file

If we have our numbers stored in a text file, then the function *scan*() can be used to read them in. We need to pass the file name to the function *scan*():

```
> x = scan(file = "ReadWithScan.txt")
```

This command will read the contents of the file *ReadWithScan.txt* into the R session.

2.3.4. Editing the data frame

The *data.entry*() command is used to edit the existing variables and data frames with a spreadsheet-like interface. A simple usage is:

```
> data.entry(x)              # x already defined;
> data.entry(x = c(NA))      # if x not defined yet.
```

When the window is closed, the values are saved. The R command edit will also open a simple window to edit data. This makes the edit functions easier, but the results of the edit will not be stored when you close the window.

2.3.5. Reading data in tables

If you want to enter multivariate sets of data, you can do any of the above for each variable. However, it may be more convenient to read in tables of data at once. The command *read.table*() will read the data in and store the results in a data frame. A data frame is a special matrix where all the variables are stored as columns and each has the same length. (Notice we need to specify that the headers be there in this case.)

```
> y = read.table("person.txt",header = TRUE)
> y
      Age    Weight    Height   Gender
1      18       150        65        F
2      21       160        68        M
3      45       180        65        M
4      54       205        69        M
```

2.4. Functions and arguments

In the R environment many things are done through the function calls. R function calls are the commands that contain one or several variables, for example:

```
> plot (height, weight)
```

The function name is plot and the arguments are height and weight. These are the *actual arguments* that only apply to the current call. The functions have a large selection of arguments in order to be able to modify symbols, for example, the plot function has line width, titles, axis type, and so on.

There are two kinds of argument specification used in R functions: *positional matching* and *keyword matching*, that is the arguments can be specified in arbitrary order with the specified keyword of the function (generally, the functions have a large selection of arguments). For example, we can write:

```
> plot (y = weight, x = height, pch = 2)
```

This is the same plot as

```
> plot ( x = height, y = weight, pch = 2)
```

The keyword *pch* was used to say that the argument is a specification of the plotting character.

2.5. Missing values

In real data analysis, a data point is frequently unavailable. R allows vectors to contain a special NA value. This value is carried through in computations so that operations on NA yield NA as the result.

2.6. Functions that create vectors

There are several functions used to create vectors in R: *c, seq, rep, and gl*. The *c*() has already been introduced. The second function, *seq* (*seq*uence), is used for equidistant series of numbers. It is always needed for graphics and ANOVA. Like,

```
> seq (4, 10, 2)
[1] 4 6 8 10,
```

which prints integers from 4 to 10 with increment of 2.

The function *rep*(), replicate, is used to generate repeated values. It is used in two variants, depending on whether the second argument is a vector or a single number. For example:

```
> rep ( c(4,5,6),3)
[1] 4 5 6 4 5 6 4 5 6,

> rep ( c(4,5,6),1:3)
[1] 4 5 5 6 6 6.
```

The first of the above function calls repeats the vector (4, 5, 6) three times. The second one repeats each value of the vector (4, 5, 6) with relevant times, which is indicated by the second argument 1:3 (means 1, 2, 3).

The function *gl*(***g***enerate factor ***l***evels) is used to generate factors by specifying the pattern of their levels. For example:

```
> gl(3,3,9,labels = c("15°F", "70°F", "125°F"))
[1] 15°F  15°F  15°F  70°F  70°F  70°F  125°F  125°F  125°F
Levels: 15°F  70°F  125°F.
```

This command generates factors for a temperature variable. The result gives three factor levels (first argument "3"), three times of replications (second argument "3"), total length of the factors (third argument "9"), and the labels of the factor levels (the last one is optional).

2.7. Matrices and arrays

An array can be considered as a multiply subscribed collection of data entries. A dimension vector is a vector of non-negative integers. A matrix is just a two-dimensional array of numbers. The dimensions are indexed from one up to the values given in the dimension vector. A vector can be used as an array if it has a dimension vector as its dim attribute. For example,

```
> x = 1:12
> dim(x) = c(3,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

This assignment gives the vector (1:12) the dim attribute that asks it to be treated as a 3-row by 4-column matrix.

There is a convenient way to create matrices in R using the function *matrix*().

```
> matrix(1:12, nrow = 3, byrow = T)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12.
```

The argument *nrows* indicates the number of rows of the vector, the argument *byrow* causes the matrix to be filled in a rowwise or columnwise fashion.

2.8. Data frames

A data frame is a list of variables of the same length with unique row names, given class *data.frame*. A data frame can be displayed in matrix form, and its rows and columns extracted using matrix indexing conventions.

The function *data.frame*() converts each of its arguments to a data frame. Objects passed to *data.frame* should have the same number of rows.

```
> d = data.frame(Temperature, Pressure)
> d
  Temperature Pressure
1         100       25
2         125       25
3         150       25
4         100       30
```

5	125	30
6	150	30
7	100	35
8	125	35
9	150	35
10	100	40
11	125	40
12	150	40
13	100	45
14	125	45
15	150	45

The above function creates a data frame list of 2 components using Temperature and Pressure variables.

The simplest way to construct a data frame is to use the *read.table*() function to read an entire data frame from an external file. This has been discussed in Section 2.3.

2.9. Graphics

One of the most important aspects of presentation and analysis of data is the generation of proper graphics. R has a simple model for constructing plots. The command is

> plot (x, y,…).

A single argument *x* can be provided to the *plot* function alternatively. But when argument *y* is missing, *x*-coordinate should be defined as a reasonable way to the *plot*() function.

You might want to modify the drawing in various ways. There are a lot of plotting parameters that you can set. Basically, a standard *x-y* plot has *x* and *y* title labels generated from the expressions being plotted. You may, however, override these labels and also add two further titles, a main title above the plot and a subtitle at the very bottom, in the plot call. You can also change the plotting symbol, plotting color, plotting type, and plotting range by passing corresponding parameters to the plot function. Let us see a plotting example.

Define the *x* coordinate:

```
> x = 1:10
```

Define the *y* coordinate:

```
> y = function(x) (1/10)*exp(-x/10).
```

Call the plot function to construct a scatter plot (Fig. 2.1):

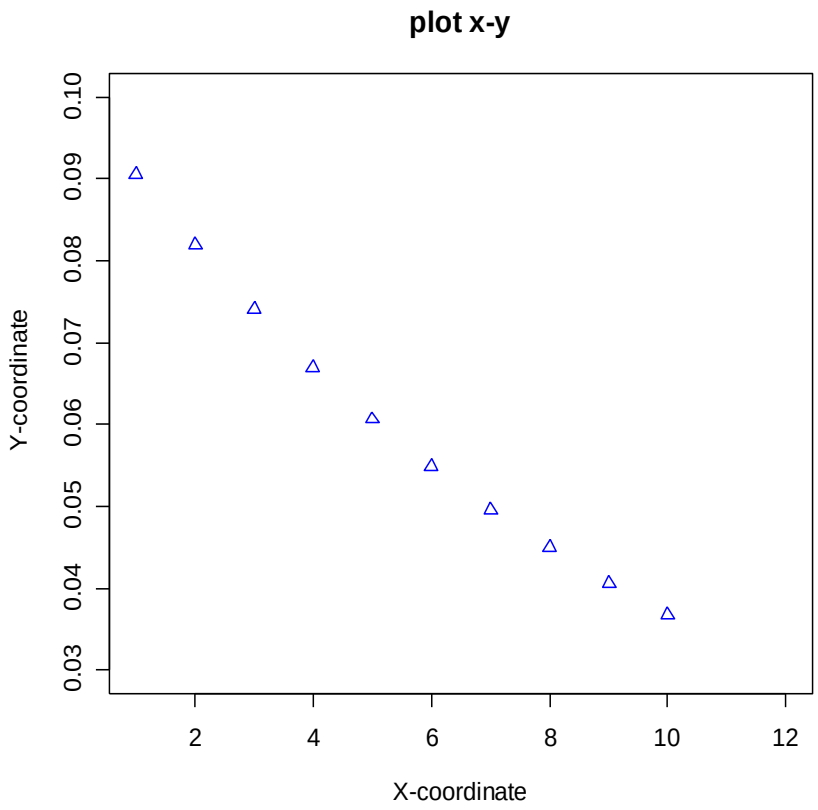


Fig. 2.1. **A scatter plot with pch = 2 and col = "blue".**

There are a lot of arguments that can be set to the *plot*(). See the following commands:

```
> plot(x,y(x), pch = 2, col = "blue", xlim = c(1,12), ylim = c(0,0.1), main = "plot x-y", xlab = "X-coordinate", ylab = "Y-coordinate")
> plot(x,y(x), pch = 2, col = "blue", xlim = c(1,12), ylim = c(0.03,0.1), main = "plot x-y", xlab = "X-coordinate", ylab = "Y-coordinate").
```

The *x* and *y* arguments provide the *x*- and *y*-coordinates for the plot. The *pch*(***p***lotting ***ch***aracter) sets the symbols for the plot. The *col* (***col***or) sets the colors for lines or points. The *xlim* (***x lim***its) and *ylim* (***y lim***its) set the *x*

and *y* limits of the plot. The *main* (***main*** title), *xlab* (***x lab***el), and *ylab* (***y lab***el) set the main title, the label of *x*-coordinate and *y*-coordinate.

The function *lines*() is used for adding connected line segments to a plot. See the command

> lines(x,y(x)-0.01, lty = 2, col = "green"),

which add a dashed line to the existing plot diagram. The argument *lty* (***l***ine ***ty***pe) and *lwd* (***l***ine ***wid***th) set the type and width of the plotting lines.

The function *abline*() can be used to add one or more straight lines to a plot. The argument *h* and *v* forms draw horizontal and vertical lines at the specified coordinates.

> abline(v = 4, h = 0.05).

This command is for adding the horizontal and vertical lines as shown in Fig. 2. 2. The function *points*() is used to add points to a plot.

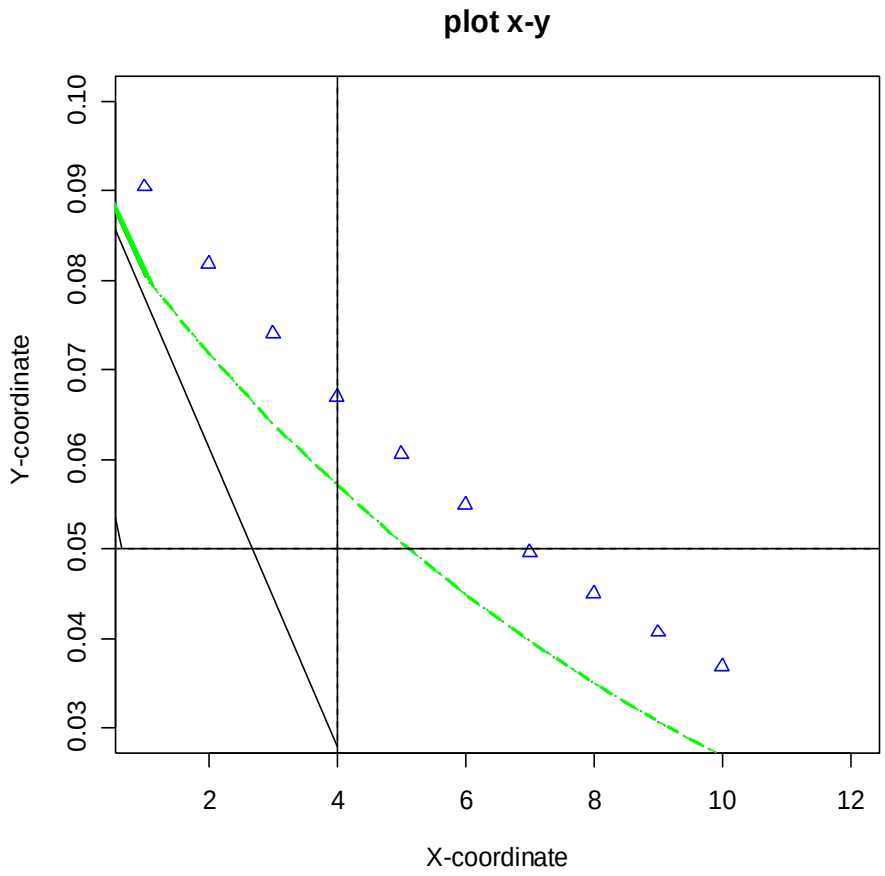


Fig. 2.2. **Adding the reference curve and straight lines, using *lines*() and *abline*()**

It is difficult to describe the graphic parameters completely at this point. You can see the further graphical parameters in the *par*() (parameters). This function is used to set or query graphical parameters. We will return to them as they are used for specific plots.

3. Probability distributions

3.1. The built-in distributions in R

The concepts of randomness and probability are central to statistics. R provides a comprehensive set of statistical distributions:

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df1, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n.

Four fundamental items can be calculated for a statistical distribution:

- density or point probability;
- cumulated probability, distribution function;

- quantile;
- pseudo-random numbers.

For all distributions implemented in R, there is a function for each of the four items listed above. The prefix name is given by *d* for the ***d***ensity, *p* for the cumulated ***p***robability distribution function (CDF), *q* for the ***q***uantile function, and *r* for pseudo-***r***andom numbers. For example, for the normal distribution, these functions are named *dnorm*, *pnorm*, *qnorm*, and *rnorm*, respectively.

Here is an example of binomial distribution.

Example 3.1.1. Albino rats used to study the hormonal regulation of a metabolic pathway are injected with a drug that inhibits body synthesis of protein. The probability that a rat will die from the drug before the experiment is over is .2.

a) What is the probability that at least eight will survive?

```
> pbinom(2,10.2)
[1] 0.6777995.
```

There are three arguments to the *pbinom*() function. The first argument is the yield value that indicates at least 2 rats will die. The following one specifies the number of trials. The last argument is the probability that a rat will die from the drug.

b) Would you be surprised if at least five died during the course of the experiment?

```
> 1-pbinom(4,10.2)
[1] 0.0327935.
```

Consider an example for calculating the binomial probability.

Example 3.1.2. Geophysicists determine the age of a zircon by counting the number of uranium fission tracks on a polished surface. A particular zircon is of such an age that the average number of tracks per square centimeter is five. What is the probability that a 2-centimeter-square sample of this zircon will reveal at most three tracks, thus leading to underestimations of the age of the material?

```
> ppois(3,10)
[1] 0.01033605.
```

There are two arguments to the *ppois()* function here. The first one is the yield value and the last arguments specify the number of trials.

Consider now an example for calculating the normal probability.

Example 3.1.3. Most galaxies take the form of a flattened disc with the major part of the light coming from this very thin fundamental plane. The degree of flattening differs from galaxy to galaxy. In the Milky Way Galaxy most gases are concentrated near the center of the fundamental plane. Let *X* denote the perpendicular distance from this center to a gaseous mass. This *X* is normally distributed with mean 0 and standard deviation 100 parsecs.

a) Sketch a graph of the density for *X*. Find the probability that a gaseous mass is located within 200 parsecs (Fig. 3.1).

```
> plot(function(x) dnorm(x,0,100), -300, 300)
```

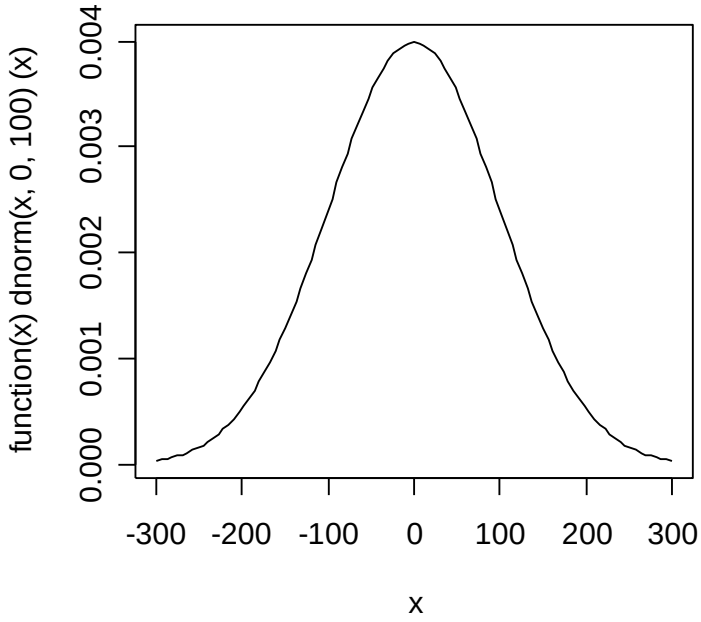


Fig. 3.1. **The plot of the density of the normal distribution**

```
> pnorm(200,0,100)-pnorm(-200,0,100)
[1] 0.9544997.
```

Here we use three arguments in the *pnorm*() function. These arguments specify the yield value (200), the mean (0), and the standard deviation (100).

The function *plot()* is used to sketch a graph of the density for *X* in the area within -300 and 300. There are many parameter arguments that can be used for the plot call.

The function of density for *X* is created further. The R language allows creating your own function. A function is defined by an assignment of the form:

```
> username <- function(arg_1, arg_2, ...) {expression}
```

The *expression* is an R expression, (usually a grouped expression), that uses the arguments *arg_i* to calculate a value. The value of the expression is the value returned for the function. A call to the function then usually takes the form *username*(expr_1, expr_2...) and may occur anywhere a function call is legitimate. For example:

```
myfunction <- function(arg1, arg2, ... ){
statements
return(object)
}
```

The objects in the function are local to the function. The object returned can be any data type. Here is an example:

```
# function example – get measures of central tendency,
# and spread for a numeric vector x. The user has a
# choice of measures and whether the results are printed.

mysummary <- function(x,npar = TRUE, print = TRUE) {
   if (!npar) {
      center <- mean(x); spread <- sd(x)
   } else {
      center <- median(x); spread <- mad(x)
   }
   if (print & !npar) {
      cat("Mean=", center, "\n", "SD=", spread, "\n")
   } else if (print & npar) {
      cat("Median=", center, "\n", "MAD=", spread, "\n")
   }
   result <- list(center=center,spread=spread)
   return(result)
}.
```

Invoking a function:

```
# invoking the function
set.seed(1234)
x <- rpois(500, 4)
y <- mysummary(x)
Median = 4
MAD = 1.4826
```

```
# y$center is the median (4)
# y$spread is the median absolute deviation (1.4826)

y <- mysummary(x, npar = FALSE, print = FALSE)
# no output
# y$center is the mean (4.052)
# y$spread is the standard deviation (2.01927).
```

b) Approximately what percentages of the gaseous masses are located more than 250 parsecs from the center of the plane?

```
> 1-(pnorm(250,0,100)-pnorm(-250,0,100))
[1] 0.01241933.
```

To get the percentage of over 250 parsecs from the center of the plane, the percentage of within 250 should be calculated:

```
pnorm(250,0,100)-pnorm(-250,0,100).
```

c) What distance has the property that 20 % of the gaseous masses are at least this far from the fundamental plane?

```
> qnorm(.1,0,100)
[1] -128.1552
> qnorm(.9,0,100)
[1] 128.1552.
```

3.2. Descriptive statistics and graphics

3.2.1. Summary statistics for a single group

R comes with many built-in functions that can apply to the data. The characteristics most used to measure the center and spread of data are the mean and standard deviation. Here are some R commands needed when measuring a data distribution:

```
> mean(x)      #find the average for  data x.
> var(x)       #find the variance for data x.
> median(x)    #find the median for data x.
> sd(x)        #find the standard deviation for data x.
```

There are a large number of ways to examine the distribution of the data set. The simplest way is to examine the numbers. There is a function frequently used for summaries of data: *summary*(). Here is an example:

```
> x = rnorm(50)
> summary(x)
      Min    1st Qu     Median      Mean    3rd Qu      Max
-1.58300  -0.69710   0.10180   0.04832   0.63240   2.29200.
```

In this example the function *rnorm*() generates an artificial data vector *x* of 50 normally distributed observations. The *summary*() function displays a numeric variable, in which 1st *Qu* and 3rd *Qu* refer to the empirical quartiles (0.25 and 0.75 quantiles).

3.2.2. The graphical display of distributions

The stem-and-leaf diagram is very useful for seeing the shape of the distribution if the data set is relatively small. The number on the left of the bar is the stem, the number on the right is the digit. You put them together to find the observation. The R command for constructing a stem-and-leaf diagram is

```
> stem().
```

If there is too much data, you can try another graphical visualization of data. The most common one is a histogram. The histogram defines a sequence of breakpoints and then counts the number of observations in the bins formed by these break points. (This is identical to the features of the *cut*() function.) It plots these with a bar similar to the bar chart, but in a histogram the bars are touching. The height of the bars can be the frequencies, or the proportions. The command for constructing a histogram is

```
> hist().
```

It can take arguments that control the specific form of the display.

Let us consider a simple example. The observations have to be pre-saved in the file named "6-11.txt", for example.

Example 3.2.1. Some efforts are currently being made to make textile fibers out of peat fibers. This would provide a source of cheap feedstock for the textile and paper industries. One variable being studied is *X*, the percentage

of the ash content of a particular variety of peat moss. Assume that a random sample of 50 mosses yields these observations in the data file "6-11.txt":

0.5	1.8	4.0	1.0	2.0
1.1	1.6	2.3	3.5	2.2
2.0	3.8	3.0	2.3	1.8
3.6	2.4	0.8	3.4	1.4
1.9	2.3	1.2	1.9	2.3
2.6	3.1	2.5	1.7	5.0
1.3	3.0	2.7	1.2	1.5
3.2	2.4	2.5	1.9	3.1
2.4	2.8	2.7	4.5	2.1
1.5	0.7	3.7	1.8	1.7.

Input the data from the data file "6-11.txt":

```
> x = scan("6-11.txt")
```

a) Read 50 items:

```
> x
[1] 0.5 1.8 4.0 1.0 2.0 1.1 1.6 2.3 3.5 2.2 2.0 3.8 3.0 2.3 1.8 3.6 2.4 0.8 3.4
[20] 1.4 1.9 2.3 1.2 1.9 2.3 2.6 3.1 2.5 1.7 5.0 1.3 3.0 2.7 1.2 1.5 3.2 2.4 2.5
[39] 1.9 3.1 2.4 2.8 2.7 4.5 2.1 1.5 0.7 3.7 1.8 1.7
```

b) Construct a stem-and-leaf diagram for these data

```
> stem(x)
```

The decimal point is the sign "|":

```
  0 | 578
  1 | 012234
  1 | 55677888999
  2 | 00123333444
  2 | 556778
  3 | 001124
  3 | 5678
  4 | 0
  4 | 5
  5 | 0
```

c) Break these data into six categories.

Find the range of data, which is the difference between the largest observation data *max(x)* and the smallest data *min(x).* Divide the range by the number of categories (6) to get the minimum length required covering this range:

```
> (max(x)-min(x))/6
[1] 0.75.
```

Set the boundaries and use the *cut*() function to break these data into six categories:

```
> breaks = c(seq(min(x)-0.05,max(x),.8),max(x)+.05)
> breaks
[1] 0.45 1.25 2.05 2.85 3.65 4.45 5.05
> cats = cut(x, breaks = breaks)
```

The function *seq*() is used further to generate the sequence number from the lower boundary "(min(x)-0.05)" to the maximum value (*max(x)*) by the increment of "minimum length(.8)".

d) Construct a frequency table and a relative frequency histogram for these data (Fig. 3.2).

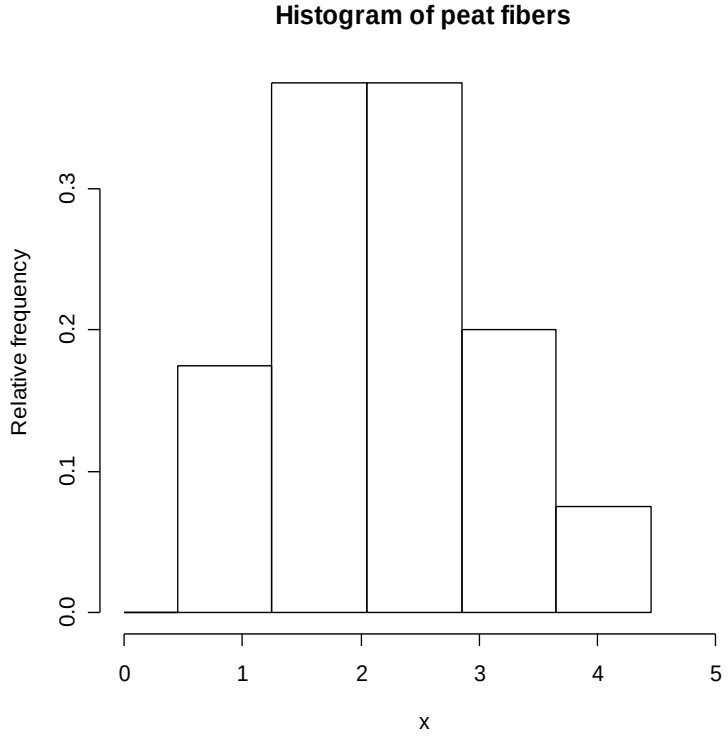


Fig. 3.2. **The relative frequency histogram for the sample of Example 3.2.1.**

The command *table*() can use the cross-classifying factors to build a contingency table of the counts at each combination of factor levels:

```
> table(cats)
cats
(0.45,1.25] (1.25,2.05] (2.05,2.85] (2.85,3.65] (3.65,4.45] (4.45,5.05]
          7          15          15           8           3           2
```

Use the function *hist*() to construct a histogram. The argument "breaks" gives the breakpoints between histogram cells.

```
> hist(x, breaks = breaks, ylab = "Relative frequency", main = "Histogram of peat fibers")
```

e) Construct a cumulative frequency table and a relative cumulative frequency ogive for these data.

```
> cumtable = cumsum(table(cats))
> cumtable
(0.45,1.25] (1.25,2.05] (2.05,2.85] (2.85,3.65] (3.65,4.45] (4.45,5.05]
          7          22          37          45          48          50
```

The command *cumsum*() returns a vector whose elements are the cumulative sums.

4. One- and two-sample tests

Now we will focus on the actual statistical analysis applications. Some of the most used statistical tests deal with comparing continuous data, either between two samples or against the a priori stipulated values.

4.1. Comparing the variance of two samples

R provides the *var.test*() (***va***riance comparison ***test***ing) function for testing two-variance comparison. This function implements an F-test on the ratio of the group variances. The command is:

```
> var.test(x, ...)
```

R also provides a function for each of the four items similar to the distribution mentioned in Section 3.1 as the F-distribution:

```
> df(x, df1, df2, log = FALSE)
> pf(q, df1, df2, ncp=0, lower.tail = TRUE, log.p = FALSE)
> qf(p, df1, df2,          lower.tail = TRUE, log.p = FALSE)
> rf(n, df1, df2)
```

Let's consider an example for comparing the variance of two samples. In this example, we don't know the sample data so we cannot use the simple function *var.test*() for comparing the variance of two samples. Instead, we can calculate the *f*-value first, then call the function *pf*() to construct the *p*-value of the testing statistics. Then compare the *p*-value with the confidence level α. We can also use another function *qf*() to find the critical point instead of calculating the *p*-value, then compare the observed *f*-value with the critical points.

Example 4.1.1. Test for equality of variances at the indicated level: $n_1 = 10$, var1 = .25; $n_2 = 8$, var2 = .05; $\alpha = .20$.

Input the data:

```
> n1 = 10
> var1 = .25
> n2 = 8
> var2 = .05
> alpha = .10
```

Calculate the observed value *f*:

```
> f = var1/var2
> f
[1] 5.
```

Calculate the critical points using command *qf*() (***q***uantiles ***f*** distribution). Set the first argument to indicate the level, the following two arguments to indicate the degree of freedom:

```
> qf(alpha, n1-1, n2-1)
[1] 0.3991517
> qf(1-alpha, n1-1, n2-1)
[1] 2.724678.
```

The observed value 5 is larger than the upper critical 2.72. Calculate the *p*-value using the *pf*() (***p***robabilities ***f*** distribution) command. The first argument in this function is the vector of quantiles, the second and third ones are the degrees of freedom, and the last one is specified to the upper tail. We choose the upper tail ("lower.tail = F" – FALSE) for the absolute value of *f* and double it to get the two-sided *p*-value.

```
> 2*pf(f, n1-1, n2-1, lower.tail = F)
[1] 0.04542107.
```

The inference: reject H_0 (the null hypothesis) because the *p*-value is smaller than α = .05.

4.2. Comparing the means of two samples when the variances are equal

The function *t.test*() is used for comparing the sample means:

```
> t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"), mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)
```

There is a number of optional arguments in the *t.test*() function. Three of them are relevant in one-sample problems. *mu* is to specify the mean value μ under the null hypothesis (default is *mu* = 0). In addition, you can specify one of the alternative hypotheses to "alternative" (default is "two.sided"). The third item that can be specified is the confidence level used for the confidence intervals; you would write *conf.level* = 0.90 to get a 90 % interval. To get the usual *t*-test for two-sample problems, it must be specified that variances are the same. This is done via the optional argument *var.equal* = *T*.

R command has a function for each of four items, *dt* (***d***ensity ***t*** distribution), *pt* (cumulated ***p***robability ***t*** distribution), *qt* (***q***uantiles ***t*** distribution), and *rt* (***r***andom number generate ***t*** distribution).

```
> dt(x, df, ncp = 0, log = FALSE)
> pt(q, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)
> qt(p, df,          lower.tail = TRUE, log.p = FALSE)
> rt(n, df)
```

Consider an example of comparing the means of two samples.

Example 4.2.1. A study of reports written by engineers has been conducted. A scale that measures the intelligibility of engineers' English has been devised. This scale, called an "index of confusion", is devised so that low scores indicate high readability.

Journals	Unpublished reports
1.79	2.39
1.87	2.56
1.62	2.36
1.96	2.62
1.65	2.51
1.75	2.29
1.74	2.58
2.06	2.41
1.69	2.86
1.67	2.49
1.94	2.33
1.33	1.94
1.70	2.14

The observations have to be stored as a table in the data file "10-13.txt" with the titles: the "header = TRUE" specifies that the first line is the header containing the names of variables contained in the file.

```
> x = read.table("10-13.txt", header = TRUE)
> x
```

If we did everything right, we will get:

	Journals	Unpublished reports
1	1.79	2.39
2	1.87	2.56
3	1.62	2.36
4	1.96	2.62
5	1.65	2.51
6	1.75	2.29
7	1.74	2.58
8	2.06	2.41
9	1.69	2.86

10	1.67	2.49
11	1.94	2.33
12	1.33	1.94
13	1.70	2.14.

When we need to get the component of the table individually, a useful way is giving the component name followed by "$" to the variable that is stored in the table.

Get sample 1:

```
> a = x$Journals
```

```
> a
 [1] 1.79 1.87 1.62 1.96 1.65 1.75 1.74 2.06 1.69 1.67 1.94 1.33 1.70.
```

Get sample 2:

```
> b = x$Unpublishedreports
> b
 [1] 2.39 2.56 2.36 2.62 2.51 2.29 2.58 2.41 2.86 2.49 2.33 1.94 2.14.
```

a) Test H_0: $\sigma 1^2 = \sigma 2^2$ at the $\alpha = .2$ level to be sure that pooling is appropriate.

```
> var.test(a,b)
```

We use further the *F*-test to compare two variances.

Data: *a* and *b*.

F = 0.6477, the numerator *df* = 12, the denominator *df* = 12, the *p*-value = = 0.463.

The alternative hypothesis: the true ratio of the variances is not equal to 1.

The 95 percent confidence interval:

```
[1] (0.1976  2.1227).
```

Sample estimates: the ratio of the variances is equal to

```
[1] 0.6477.
```

The inference: since the *p*-value 0.463 is larger than $\alpha = .2$, pooling is appropriate.

b) Find sp^2.

There is no function to get the *sp* value. We have to give the expression for calculating the value *sp*:

```
> sp2 = (12*var(a)+12*var(b))/24
> sp2
[1] 0.0432641
```

c) Find a 90 % confidence interval on μ1 - μ2, using the *t.test*() function:

```
> t.test(a,b,var.equal = T, conf.level = .9)
```

A two-sample *t*-test

Data: *a* and *b*.

t = -8.2124, *df* = 24, *p*-value = 1.979e-08.

The alternative hypothesis: the true difference in the means is not equal to 0.
The 90-percent confidence interval:

(-0.8095813 -0.5304187).

Sample estimates: the mean of *x*; the mean of *y*:

1.7515; 2.4215.

d) Does it appear to be a difference between μ1 and μ2? One can use for this the *t.test*() function.

Data: *a*.

t = 34.2422, *df* = 12, the *p*-value = 2.449e-13.

The alternative hypothesis: the true mean is not equal to 0.
The 90-percent confidence interval:

(1.660372; 1.842705).

Sample estimates: the mean of *x*:

1.751538

```
> t.test(b, conf.level = .9)
```

A one-sample *t*-test

Data: *b*.

t = 38.1, *df* = 12, the *p*-value = 6.877e-14.

The alternative hypothesis: the true mean is not equal to 0.
The 90-percent confidence interval:

2.308261 2.534816

Sample estimates: the mean of *x:*

2.421538.

In the next example, the data of two samples are not given. We only know the means and the variances of the samples. So we cannot simply use the *t.test*() and the *var.test*() function for testing the statistics.

Example 4.2.2. Environmental testing is an attempt to test a component under conditions that closely simulate the environment in which the component will be used. An electrical component is to be used in two different locations in Alaska. Before environmental testing can be conducted, it is necessary to determine the soil composition in the localities. These data are obtained on the percentage of SiO_2 by weight of the soil:

Anchorage: $n_1 = 10$, $\mu 1 = 64.94$, var1 = 9.
Kodiak: $n_2 = 16$, $\mu 2 = 57.06$, var2 = 7.29.

Input the data:

```
> n1= 10
> mu1 = 64.94
> var1 = 9
> n2 = 16
> mu2 = 57.06
> var2 = 7.29
```

a) Test H_0: var1 = var2 at the $\alpha = .2$ level.
Calculate the observed value:

```
> f = var1 / var2
> f
[1] 1.234568.
```

Calculate the critical points, use the function *qt* (***q***uantiles ***f***-distribution). The first argument is the vector of probabilities; the second and third ones are the degrees of freedom of observations.

```
> qf(.9,n1-1,n2-1)
[1] 2.086209
> qf(.1,n1-1,n2-1)
[1] 0.4274191.
```

The observed value lies within the critical points.

Calculate the *p*-value, use the function *pf* (***p***robabilities ***f***-distribution). The first argument is the vector of quantiles, the following two arguments are the degrees of freedom of observations, and the last one is to indicate that we want to see the upper tail of the distribution. We double it to get the two-sided *p*-value.

```
> 2*pf(f, n1-1, n2-1, lower.tail = F)
[1] 0.6899627.
```

The inference: we cannot reject H_0 since the *p*-value is larger than $\alpha = 0.2$.

b) Find *sdpool*2.

```
> sdpool = sqrt(((n1 - 1) * var1 + (n2 - 1) * var2)/(n1 + n2 -2))
> sdpool
[1] 2.816248
> sdpool^2
[1] 7.93125
```

c) Find a 99 % confidence interval on $\mu 1 - \mu 2$.

The lower point:

```
> mu1-mu2+qt(0.995,n1+n2-2)*sqrt(sdpool^2*(1/n1+1/n2))
[1] 11.05527
```

The upper point:

```
> mu1-mu2+qt(0.005,n1+n2-2)*sqrt(sdpool^2*(1/n1+1/n2))
[1] 4.704731.
```

The inference: a 99 % confidence interval on $\mu 1 - \mu 2$ is (4.7047; 11.0553).

4.3. Comparing the means of two samples when the variances are unequal

The function *t.test*() also performs the two-sample *t*-test when we assume the variances unequal. This is done without specifying the optional argument *var.equal* in which the default value is False. Let's consider an example.

Example 4.3.1. A study is conducted to compare the tensile strength of two types of roof coatings. It is thought that, on the average, butyl coatings are stronger than acrylic coatings. The following data have been gathered:

Tensile strength, lb/in^2

Acrylic				Butyl			
246.3	247.7	287.5	248.3	340.7	263.4	272.6	271.4
255.0	246.3	284.6	243.7	270.1	341.6	332.6	303.9
245.8	214.0	268.7	276.7	371.6	307.0	362.2	324.7
250.7	242.7	302.6	254.9	306.6	319.1	358.1	360.1

a) Input the data obtained:

```
> Acrylic = scan("acrylic.data")
1: 246.3 255.0 245.8 250.7 247.7 246.3 214.0 242.7 287.5 284.6 268.7 302.6
13: 248.3 243.7 276.7 254.9
17:
```

The function *scan*() reads 16 items.

```
> Butyl = scan("autyl.data")
1: 340.7 270.1 371.6 306.6 263.4 341.6 307.0 319.1 272.6 332.6 362.2 358.1
13: 271.4 303.9 324.7 360.1
17:
```

The function *scan*() reads 16 more items.

b) Is pooling appropriate?

```
> var.test(Acrylic, Butyl)
```

An F-test to compare two variances

Data: Acrylic and Butyl.

F = 0.3633, the numerator *df* = 15, the denominator *df* = 15, the *p*-value = = 0.05878.

The alternative hypothesis: the true ratio of the variances is not equal to 1.

The 95-percent confidence interval:

(0.1269485 1.0399078).

Sample estimates

The ratio of variances:

0.3633.

The inference: because the p-value is very small, pooling is not appropriate.

c) Can H_0 be rejected? Explain, based on the *p*-value of your test.

Because it is a one-sided test, we specified the test is desired against alternatives less than μ (mean) by setting the optional argument: *alternative = "less"*.

```
> t.test(Acrylic, Butyl, alternative = "less")
```

The Welch's two-sample t-test

Data: Acrylic and Butyl.

t = -5.8755, *df* = 24.629, the *p*-value = 2.094e-06.

The alternative hypothesis: the true difference in the means is less than 0.

The 95-percent confidence interval:

(-Inf ; -43.86719).

Sample estimates: the mean of *x*, the mean of *y*:

257.2188; 319.0813.

The inference: the *p*-value = 2.094e-06 is very small, so the null hypothesis is rejected.

Example 4.3.2. The aseptic packaging of juices is a method of packaging that entails rapid heating followed by quick cooling to room temperature in an air-free container. Such packaging allows the juices to be stored unrefrigerated. Two machines used to fill aseptic packages are compared. These data are obtained in the number of containers that can be filled per minute:

Machine I: $n_1 = 25$, $\mu 1 = 115.5$, var1 = 25.2
Machine II: $n_2 = 25$, $\mu 2 = 112.7$, var2 = 7.6

Input the data:

```
> n1 = 25
> mu1 = 115.5
> var1 = 25.2
> n2 = 25
> mu2 = 112.7
> var2 = 7.6.
```

a) Is pooling appropriate?
Calculate the observed value of test statistics:

```
> f = var1 / var2
> f
[1] 3.315789.
```

Calculate the critical points using the function *qf*() (***q***uantiles ***f*** distribution):

```
> qf(.05,n1-1,n2-1)
[1] 0.5040933
> qf(.95,n1-1,n2-1)
[1] 1.983760.
```

The inference: the observed value of the test statistics is bigger than the upper critical point 1.984, so pooling is not appropriate.

Calculate the *p*-value:

```
> 2*pf(f, n1-1, n2-1, lower.tail = F)
[1] 0.004712739.
```

The inference: the *p*-value is very small, so pooling is not appropriate.

b) Construct a 90 % confidence interval on μ1 - μ2. For this, calculate the number of degrees of freedom:

```
> s1 = var1/n1
> s2 = var2/n2
> gamma =(s1+s2)^2/(s1^2/(n1-1)+s2^2/(n2-1))
> gamma
[1] 37.26928.
```

We obtain that the number of degrees of freedom is 37.

Find a 90 % confidence interval on μ1 - μ2. The function *qt*() (***q***uantiles ***t***-distribution) is used in this case. The first argument of the function is the probabilities and the second one is the degrees of freedom 37:

```
> mu = mu1-mu2
> q = qt(.05, 37)
> qt(0.05,37)
[1] q = -1.687094
> mu - q*sqrt(s1+s2)
[1] 4.73244
> mu + q*sqrt(s1+s2)
[1] 0.8675596.
```

The inference: the 90 % confidence interval on μ1 - μ2 is (0.8676; 4.7324).

4.4. A one-sample *t*-test

We will focus on the one-sample *t*-test in this section. If we only knew the mean and standard deviation of a sample, how could we conduct the *t*-test using the theoretical mean? We can use the command *pt* (probabilities *t*-distribution). In Example 4.4.1, we first construct the *t.value*, then calculate the *p*-value. In Example 4.4.2, the sample data is given, so we can use the easier way for statistical testing.

Example 4.4.1. A low-noise transistor for use in computing products is being developed. It is claimed that the mean noise level will be below the 2.5-dB level of products currently in use.

A sample of 16 transistors yields the mean $\bar{x}$ = 1.8 with the standard deviation *s* = .8. Find the *p*-value for the *t*-test. Could we think that H_0 should be rejected? What assumption should we make concerning the distribution of the random variable *x*, the noise level of a transistor?

Input the data:

```
> sample.mean = 1.8
> sample.sd = 0.8
> n = 16
> mu = 2.5.
```

Calculate the *t*-value:

```
> t.value = (sample.mean - mu)/(sample.sd/sqrt(n))
> t.value
[1] -3.5.
```

Calculate the *p*-value using the R-function *pt*() (***p***robabilities ***t*** distribution). Set the vector of quantiles (*t.value*) to the first argument and set the degrees of freedom to the second one:

```
> p.value = pt(t.value, n - 1)
> p.value
[1] 0.001611765.
```

The inference: since the *p*-value is very small, H_0 can be rejected.

Example 4.4.2. Clams, mussels, and other organisms that adhere to the water intake tunnels of electrical power plants are called macrofoulants. These organisms can, if left unchecked, inhibit the flow of water through the tunnel.

Various techniques have been tried to control this problem, among them increasing the flow rate and coating the tunnel with teflon, wax, or grease. In a year's time at a particular plant an unprotected tunnel accumulates a coating of macrofoulants that averages 5 inches in thickness over the length of the tunnel.

A new silicone oil paint is being tested. It is hoped that this paint will reduce the amount of macrofoulants that adhere to the tunnel walls. The tunnel is cleaned, painted with the new paint, and put back into operation under normal working conditions.

At the end of a year's time the thickness in inches of the macrofoulants coating is measured at 16 randomly selected locations within the tunnel. These are the data:

4.2	4.5	4.1	4.6
4.4	4.0	4.7	4.3
5.0	6.2	3.6	4.5
5.1	3.5	3.0	2.8.

Input the data in the R environment:

```
> sample = scan("8-43.txt")
```

Read 16 items:

```
> sample
[1] 4.2  4.5  4.1  4.6  4.4  4.0  4.7  4.3  5.0  6.2  3.6  4.5  5.1  3.5  3.0  2.8
```

a) Do these data support the contention that the new paint reduces the average thickness of the macrofoulants within this tunnel? Explain, based on the *p*-value of the test.

In this case, there are three arguments. The argument "mu = 5" attaches a value to the formal argument "mu", which represents the Greek letter μ conventionally used for the theoretical mean. Normally, the *t.test*() uses the default value "mu = 0", if it is not specified otherwise. The "alternative = "less" indicated that the test is designed against alternatives less than μ:

```
> t.test(sample, mu = 5, alternative = "less")
```

A one-sample *t*-test

Data: the sample for which t = -3.4721, df = 15, the p-value = 0.001707.
The alternative hypothesis: the true mean is less than 5.
The 95-percent confidence interval:

(-Inf; 4.644142).

Sample estimates: the mean of x:

[1] 4.28125.

The inference: because the *p*-value based on the *t.test*() is very small, the H_0 can be rejected.

b) If α had been preset at .05, would H_0 have been rejected?
Because the *p*-value = 0.0017 < alpha = 0.05, H_0 can be rejected.

4.5. Comparing the variance of a sample with a known value

Example 4.5.1. Incompatibility is always a problem when working with computers. A new digital sampling frequency converter is being tested. It takes the sampling frequency from 30 to 52 kilohertz, word lengths of 14 to 18 bits and arbitrary formats and converts it to the output sampling frequency.

The conversion error is thought to have a standard deviation of less than 150 picoseconds. These data are obtained on the sampling error made in 20 tests of the device:

133.2	-11.5	-126.1	17.9	139.4
-81.7	314.8	147.1	-70.4	104.3
56.9	44.4	1.9	-4.7	96.1
-57.3	-43.8	-95.5	-1.2	9.9.

For these data: *mean*(x) = 28.69, *std*(x) = 104.93.

a) Test H_0: $\mu = 0$, H_1: $\mu \neq 0$, at the $\alpha = .1$ level.

Input the data:

```
> sample = scan("8-49.txt")
```

Read 20 items:

```
> sample
 [1]  133.2  -11.5 -126.1   17.9  139.4  -81.7  314.8  147.1  -70.4  104.3
[11]   56.9   44.4    1.9   -4.7   96.1  -57.3  -43.8  -95.5   -1.2    9.9
```

A one-sample *t*-test

```
> t.test(sample)
```

Data: a sample.

t = 1.2225, df = 19, the p-value = 0.2365.

The alternative hypothesis: the true mean is not equal to 0.

The 95-percent confidence interval:

[1] -20.42544 77.79544

Sample estimates: the mean of *x*:

[1] 28.685.

The inference: because the *p*-value is bigger than *alpha* = .1, H_0 cannot be rejected.

b) Test H_0: $\sigma = 150$, H_1: $\sigma < 150$ at the $\alpha = .1$ level.

Calculate the observed value of the test statistics chi_2:

```
> sample.sd = sd(sample)
> sample.sd
[1] 104.9336
> pop.sd = 150
> n = 20
> chi2 = (n - 1) * sample.sd^2/pop.sd^2
[1] 9.2982.
```

Calculate the *p*-value using the function *pchisq*() (***p***robabilities ***chisq***uared distribution). The first argument is the vector of quantiles and the second one is the degrees of freedom:

```
> pchisq((n - 1) * sample.sd/pop.sd, n - 1)
[1] 0.1767.
```

The inference: because the *p*-value is larger than *alpha* = .1, H_0 cannot be rejected.

c) Calculate the critical point, use the function *qchisq*() (***q***uantiles ***chisq***uared distribution). Attach the vector probabilities to the first argument and the degrees of freedom to the second one:

```
> qchisq(0.1,n-1)
[1] 11.65091.
```

The inference: the observed value 9.3 is smaller than the critical point 11.7. The way of comparing the variance of samples with a known value is shown in the answer of question c).

When we test the hypothesis for the value of variance, the test statistic used to test the hypothesis is known to follow a chi-squared distribution. R has a function for the chi-squared distribution. In this example, we use the functions *pchisq*() and *qchisq*() to calculate the *p*-value and critical points for the statistic of the chi-squared distribution. It is called in a similar way as other distributions introduced before.

5. Regression analysis

5.1. Simple linear regression

For linear regression analysis, the function *lm*() is used (***l***iner ***m***odel). The *lm*() function can be used to carry out the simple and multiple linear regression analysis. The next example will show how to use the R command for linear regression analysis.

Example 5.1.1. An investigation was conducted to study gasoline mileage in automobiles when used exclusively for urban driving. Ten properly tuned and serviced automobiles manufactured during the same year were used in the study. Each automobile was driven for 1000 miles and the average number of miles per gallon (mi/gal) (*y*) and the weight of the car in tons (*x*) was recorded at different ambient temperatures in K^0(*z*). The data obtained are:

Car number:	1	2	3	4	5	6	7	8	9	10
Miles per gallon (*y*)	17.9	16.5	16.4	16.8	18.8	15.5	17.5	16.4	15.9	18.3
Weight in ton (*x*):	1.35	1.90	1.70	1.80	1.30	2.05	1.60	1.80	1.85	1.40
K^0(z)	90	30	80	40	35	45	50	60	65	30

Input the data in R:

```
> weight = c(1.35, 1.90, 1.70, 1.80, 1.30, 2.05, 1.60, 1.80, 1.85, 1.40)
> temperature = c( 90, 30, 80, 40, 35, 45, 50, 60, 65, 30)
> miles = c(17.9, 16.5, 16.4, 16.8, 18.8, 15.5, 17.5, 16.4, 15.9, 18.3)
```

Linear regression analysis

The linear regression function in R:

```
> lm(miles~weight+temperature)
```

The call:

```
lm(formula = miles ~ weight + temperature)
```

The coefficients:

(Intercept)	weight	temperature
24.74887	-4.15933	-0.01490

The argument to *lm*() is a model formula that describes the model to be fit. In the formula, the tilde symbol "~" should be read as "described by".

The output of *lm*() is very brief. All you see is the coefficients, which indicate the estimated intercept and the estimated slope for each variable. The best fitting straight line can be obtained from these estimated values, but no other information is given from the output. In fact, the *lm*-object contains much more information than you see when it is printed.

The *summary*() is an extraction function that is used to print out the desired quantities of a regression analysis:

```
> summary(lm(miles~weight+temperature))
```

The call:

```
lm(formula = miles ~ weight + temperature).
```

The results

The residuals:

Min	1Q	Median	3Q	Max
-0.185929	-0.077793	0.005608	0.105263	0.150812

The coefficients:

	Estimate	Std. error	t-value	Pr(>|t|)
(Intercept)	24.748874	0.348882	70.938	2.91e-11
weight	-4.159335	0.186705	-22.278	9.28e-08
temperature	-0.014895	0.002276	-6.545	0.00032

The inference: the residual standard error: 0.1416 on 7 degrees of freedom;
The multiple R-Squared: 0.9866; the adjusted R-squared: 0.9827;
F-statistic: 257.3 on 2 and 7 DF; the *p*-value: 2.798e-07.

The above is a format that looks more like what other statistical packages would output.

5.2. Residuals and fitted values

There are two further extraction functions (*fitted*() and *resid*()) that can be used to extract information about the results of a regression analysis. Let's construct the residual and fitted values in R for the observations of Example 5.1.1.

Store the value returned by the *lm* function under the name "lmMiles".

```
> lmMiles = lm(miles~weight+temperature)
```

Output the fitted values that you would expect for the given *x*-value according to the # fitting straight line by the function *fitted*():

```
> fitted(lmMiles)
```

The results of applying this function are as follows:

1	2	3	4	5
17.79322	16.39929	16.48640	16.66627	18.82041
6	7	8	9	10
15.55196	17.34919	16.36837	16.08593	18.47895.

Show the difference between the above fitted values and the observed values:

```
> resid(lmMiles)
```

1	2	3	4	5
0.10677943	0.10071238	-0.08640365	0.13372910	-0.02041326
6	7	8	9	10
-0.05196217	0.15081236	0.03162945	-0.18592873	-0.17895490.

5.3. The confidence and prediction interval

There are two kinds of bands around the fitted lines: the confidence interval and the prediction interval. The predicted values for these two bands can be extracted with the function *predict*(). The main arguments in the *predict*() function are:

object: the linear mode object you want to predict values from;

newdata: an optional data frame in the *which*() function to look for variables with *which*() to predict; if omitted, the fitted values are used;

interval: the type of the interval calculation, including "confidence" and "prediction".

level: the tolerance/confidence level.

You can set *interval* = "confidence" or *interval* = "prediction" to obtain the confidence or prediction interval by evaluating the regression functions.

Calculate the confidence and prediction interval for the regression model of Example 5.1.1. The result of the linear regression is saved, for example, in "lmMiles":

```
> predict(lmMiles,interval = "confidence")
```

	fit	lwr	upr
1	17.79322	17.53515	18.05129
2	16.39929	16.21686	16.58171
3	16.48640	16.30320	16.66961
4	16.66627	16.53203	16.80052
5	18.82041	18.59530	19.04553
6	15.55196	15.35477	15.74915
7	17.34919	17.23703	17.46134
8	16.36837	16.24054	16.49620
9	16.08593	15.93504	16.23682
10	18.47895	18.27011	18.68780.

The *fit* is the expected values, and *lwr* and *upr* are the lower and upper confidence limits for the expected values. Because the argument "newdata" is omitted here, the fitted values are used to produce the required bands. If you want to find a confidence or prediction interval on the new data, you need to add this new data in the argument "newdata".

Set the new data:

```
> newdata = data.frame(weight = 1.5, temperature = 40)
```

Find the confidence interval obtained by the regression function in the newdata frame:

```
 > predict(lmMiles, newdata, interval = "confidence")
$fit
             fit              lwr              upr
 [1,] 17.91407       17.76318         18.06496

> predict(lmMiles, newdata, interval = "prediction", level = .90)
             fit              lwr              upr
 [1,] 17.91407       17.61982         18.20832.
```

You may set the confidence or prediction level to the argument "level = = .90".

5.4. Correlation

The function *cor*() (***cor***relation) can be used to compute the correlation between two or more vectors.

Calculate the correlation between two predict or variables, *miles* and *weight*.

```
> cor(miles, weight)
[1] -0.9510329.
```

You can obtain the entire matrix of correlations between all variables in a data frame.

Set the data frame under the name *x*:

```
> x = data.frame(miles,weight,temperature)
> cor(x)
```

	miles	weight	temperature
miles	1.0000000	-0.9510329	-0.1878411
weight	-0.9510329	1.0000000	-0.1022271
temperature	-0.1878411	-0.1022271	1.0000000.

Now you obtain the correlation between all variables. This is exactly the same *p*-value as in the regression analysis in the output of the *summary*() command.

5.5. Testing the hypotheses about the model parameters

R can automatically do a hypotheses test for the assumption $\beta_0 = 0$, $\beta_1=0$, $\beta_2= 0$ … The testing *p*-value is included in the output of the *summary*() command in the column Pr(>|t|).

```
> summary(lmMiles)
```

The call:

```
lm(formula = miles ~ weight + temperature)
```

The residuals:

Min	1Q	Median	3Q	Max
-0.185929	-0.077793	0.005608	0.105263	0.150812

The coefficients:

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	24.748874	0.348882	70.938	2.91e-11
weight	-4.159335	0.186705	-22.278	9.28e-08
temperature	-0.014895	0.002276	-6.545	0.00032

The residual standard error: 0.1416 on 7 degrees of freedom.

The inference: the multiple R-squared: 0.9866; the adjusted R-squared: 0.9827; F-statistic: 257.3 on 2 and 7 DF, the *p*-value: 2.798e-07.

The inference: the *p*-values of the *Pr*(>|t|) column are very small, so all of the hypotheses have to be rejected.

5.6. The criteria for selection of variables

The function *step*() can be used to choose a model by AIC (Akaike' Information Criterion) in a stepwise algorithm. The main arguments to the function are:

object: the object that is used as the initial model in the stepwise search;

direction: the mode of the stepwise search can be one of "both", "backward", or "forward" with the default of "both"; if the "scope" argument is missing, the default for *direction* is "backward";

trace: if positive, information is printed during the running of "step"; larger values may give more detailed information.

steps: the maximum number of steps to be considered; the default is 1000.

Let's use the stepwise method to choose a regression model of Example 5.1.1:

```
> step(lmMiles)
```

Start from AIC = -36.66;

```
miles ~ weight + temperature.
```

The result:

	Df	Sum of Sq	RSS	AIC
<none>			0.140	-36.662
- temperature	1	0.859	0.999	-19.033
- weight	1	9.951	10.091	4.091

The call:

```
lm(formula = miles ~ weight + temperature).
```

The coefficients:

(Intercept)	weight	temperature
24.74887	-4.15933	-0.01490.

The inference: the stepwise-selected model is returned.

5.7. Diagnostics

The *influence*() function provides the basic quantities that are used in forming a wide variety of diagnostics for checking the quality of regression fits.

```
> influence(lmMiles)
$hat
        1         2         3         4         5
0.5940565 0.2968422 0.2993973 0.1607538 0.4520247
        6         7         8         9        10
0.3468339 0.1122063 0.1457563 0.2030800 0.3890490.

$coefficients
   (Intercept)      weight     temperature
1   0.11543054 -0.127260994  2.362584e-03
2  -0.02779725  0.049047656 -7.625658e-04
3   0.05529260 -0.012707478 -8.826730e-04
4  -0.01009853  0.030314626 -4.713150e-04
5  -0.05720109  0.025699706  1.986451e-04
6   0.07205954 -0.050574845  8.948367e-05
7   0.06283837 -0.023070599 -1.372913e-04
8  -0.01507530  0.008648304  8.175270e-05
9   0.15034206 -0.077303183 -8.417170e-04
10 -0.38631183  0.154322657  1.876765e-03.

$sigma
        1         2         3         4         5
0.1367860 0.1448706 0.1470226 0.1408550 0.1525283
        6         7         8         9        10
0.1506740 0.1382816 0.1523038 0.1271291 0.1210590.

$wt.res
         1          2          3          4           5
0.10677943 0.10071238 -0.08640365 0.13372910 -0.02041326
         6          7          8          9          10
0.05196217 0.15081236 0.03162945 -0.18592873 -0.17895490.
```

The result of *influence*() contains the following components:

hat: a vector containing the diagonal of the "hat" matrix;

coefficients: a matrix whose *i*-th row contains the change in the estimated coefficients which results when the *i*-th case is dropped from the regression.

sigma: a vector whose *i*-th element contains the estimate of the residual standard deviation obtained when the *i*-th case is dropped from the regression.

wt.res: a vector of weighted residuals.

There is a set of functions that can be used to compute some of the regression (leave-one-out deletion) diagnostics for linear and generalized linear models.

5.7.1. Studentized deleted residuals

The functions *rstudent*() (***r***esidual ***student***ized) and *rstandard*() (***r***esidual ***standard***ized) are used to obtain the studentized and standardized residuals respectively.

```
> rstudent(lmMiles)
        1         2         3         4         5
 1.2252170 0.8290415 -0.7021213 1.0363558 -0.1807929
        6         7         8         9        10
-0.4267142 1.1574880 0.2246932 -1.6383022 -1.8912248
```

5.7.2. Hat matrix leverage

```
> hatvalues(lmMiles)
        1         2         3         4         5
 0.5940565 0.2968422 0.2993973 0.1607538 0.4520247
        6         7         8         9        10
 0.3468339 0.1122063 0.1457563 0.2030800 0.3890490
```

Note: the *hatvalues*() function gives the same result with the values of the hat component in the influence function.

5.7.3. The influence on single fitted values – DFFITS

```
> dffits(lmMiles)
         1          2          3          4          5
1.48215679 0.53865748 -0.45898682 0.45357044 -0.16420334
         6          7          8          9         10
-0.31094671 0.41149928 0.09281384 -0.82702815 -1.50918391.
```

5.7.4. The influence on all fitted values – Cook's distance

```
> cooks.distance(lmMiles)
          1           2           3           4           5
0.683339996 0.101239673 0.075706564 0.067857705 0.010428696
          6           7           8           9          10
0.036493352 0.053830944 0.003322094 0.183778982 0.554937168.
```

5.7.5. The influence on the regression coefficients – DFBETAS

```
> dfbetas(lmMiles)
   (Intercept)      weight temperature
1   0.34249727 -0.70559225  1.07471017
2  -0.07787532  0.25676649 -0.32752396
3   0.15263745 -0.06555043 -0.37356120
4  -0.02909806  0.16322254 -0.20820198
5  -0.15220605  0.12778442  0.08103511
6   0.19410255 -0.25456366  0.03695314
7   0.18443299 -0.12653035 -0.06177668
8  -0.04017292  0.04306461  0.03339930
9   0.47996934 -0.46116093 -0.41197125
10 -1.29514655  0.96679126  0.96462535.
```

5.8. Examples

Example 5.8.1. The body fat example. We want to study the relation of the amount of body fat (y) to several possible predictor variables, based on a sample of 20 healthy females 25 – 34 years old. The possible predictor variables are triceps skinfold thickness (x_1), thigh circumference (x_2), and midarm circumference (x_3). The amount of body fat for each of the 20 persons was obtained by a cumbersome and expensive procedure requiring the immersion for the person in water.

1) Read the data from the data file.

```
> BodyFat = scan("CH07TA01.DAT", list(x1 = 0, x2 = 0, x3 = 0, y = 0))
```

Read 20 records. The first argument is the name of the file where the data are reading from. The second argument is a dummy list structure that establishes the mode of the vectors to be read. The result, BodyFat, is a list whose components are the three vectors read in. You can access the vectors separately like: BodyFat$$x_1$, BodyFat$$x_2$, BodyFat$y.

2) List the data:

> BodyFat
$$x_1$
 [1] 19.5 24.7 30.7 29.8 19.1 25.6 31.4 27.9 22.1 25.5 31.1 30.4 18.7 19.7 14.6 29.5 27.7 30.2 22.7 25.2

$$x_2$
 [1] 43.1 49.8 51.9 54.3 42.2 53.9 58.5 52.1 49.9 53.5 56.6 56.7 46.5 44.2 42.7 54.4 55.3 58.6 48.2 51.0

$$x_3$
 [1] 29.1 28.2 37.0 31.1 30.9 23.7 27.6 30.6 23.2 24.8 30.0 28.3 23.0 28.6 21.3 30.1 25.7 24.6 27.1 27.5

$y
 [1] 11.9 22.8 18.7 20.1 12.9 21.7 27.1 25.4 21.3 19.3 25.4 27.2 11.7 17.8 12.8 23.9 22.6 25.4 14.8 21.1

3) Regression of *y* on x_1:

> lm(BodyFat$y~BodyFat$$x_1$))

The call:

lm(formula = BodyFat$y ~ BodyFat$$x_1$).

The coefficients:

(Intercept)	BodyFat$$x_1$
-1.4961	0.8572.

4) Summary of the regression of *y* on x_1, x_2, and x_3:

> summary(lm(BodyFat$y~BodyFat$$x_1$+BodyFat$$x_2$+BodyFat$$x_3$))

The call:

lm(formula = BodyFat$y ~ BodyFat$$x_1$ + BodyFat$$x_2$ + BodyFat$$x_3$).

The residuals:

Min	1Q	Media	3Q	Max
-3.7263	-1.6111	0.3923	1.4656	4.1277

The coefficients:

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	117.085	99.782	1.173	0.258
BodyFat$$x_1$	4.334	3.016	1.437	0.170
BodyFat$$x_2$	-2.857	2.582	-1.106	0.285
BodyFat$$x_3$	-2.186	1.595	-1.370	0.190

The inference: the residual standard error: 2.48 on 16 degrees of freedom; the multiple *R*-squared: 0.8014; the adjusted *R*-squared: 0.7641; *F*-statistic: 21.52 on 3 and 16 *DF*, the *p*-value: 7.343e-06.

5) Testing the assumptions of the model of regression of *y* on x_1 and x_2.

The validity of the model can be checked graphically. We can use the plot function to test if the regression model is significant. We can test for correlations by looking if there are trends in the data. This can be done with plots of the residuals vs time and order. We can test the assumption that the errors have the same variance with plots of residuals vs time order and fitted values.

The plot command will do these tests if we give it the result of the regression. It will plot 4 separate graphs unless you tell R to place 4 graphs on one plot window in advance with the function *par*(mfrow = c(2,2)), Fig. 5.1. The function *par*() (***par***ameters) is used to set the graphical parameters. A vector of the form *c*(2,2) is set to the argument *mfrow*, which tell you the subsequent figures will be drawn in a *nrow*-by-*ncolumn* array on the device by rows.

Save the result of regression *y* on x_1 and x_2 under the name of *lmResult*.

> lmResult = lm(BodyFat$y~BodyFat$$x_1$+BodyFat$$x_2$)

Set the argument of graphical parameters to indicate the subsequent graphs that will be displayed in 2 rows by 2 columns fashion on the same plot window.

> par(mfrow = c(2,2))

Plot four graphs of the regression model.

```
> plot(lmResult)
```

This is different from the *plot*(x,y), which produces a scatter plot. There are four plots produced by *plot* (lmResult):

- Residuals vs fitted. This plots the fitted values against the residuals. Look for spread around the line $y = 0$ and no obvious trend.
- Normal *Q-Q* plot. The residuals are normal if this graph falls close to a straight line.
- Scale-Location plot. This plot shows the square root of the standardized residuals. The tallest point has the largest residuals.
- Cook's distance plot. This plot identifies which plot has a lot of influence in the regression line.

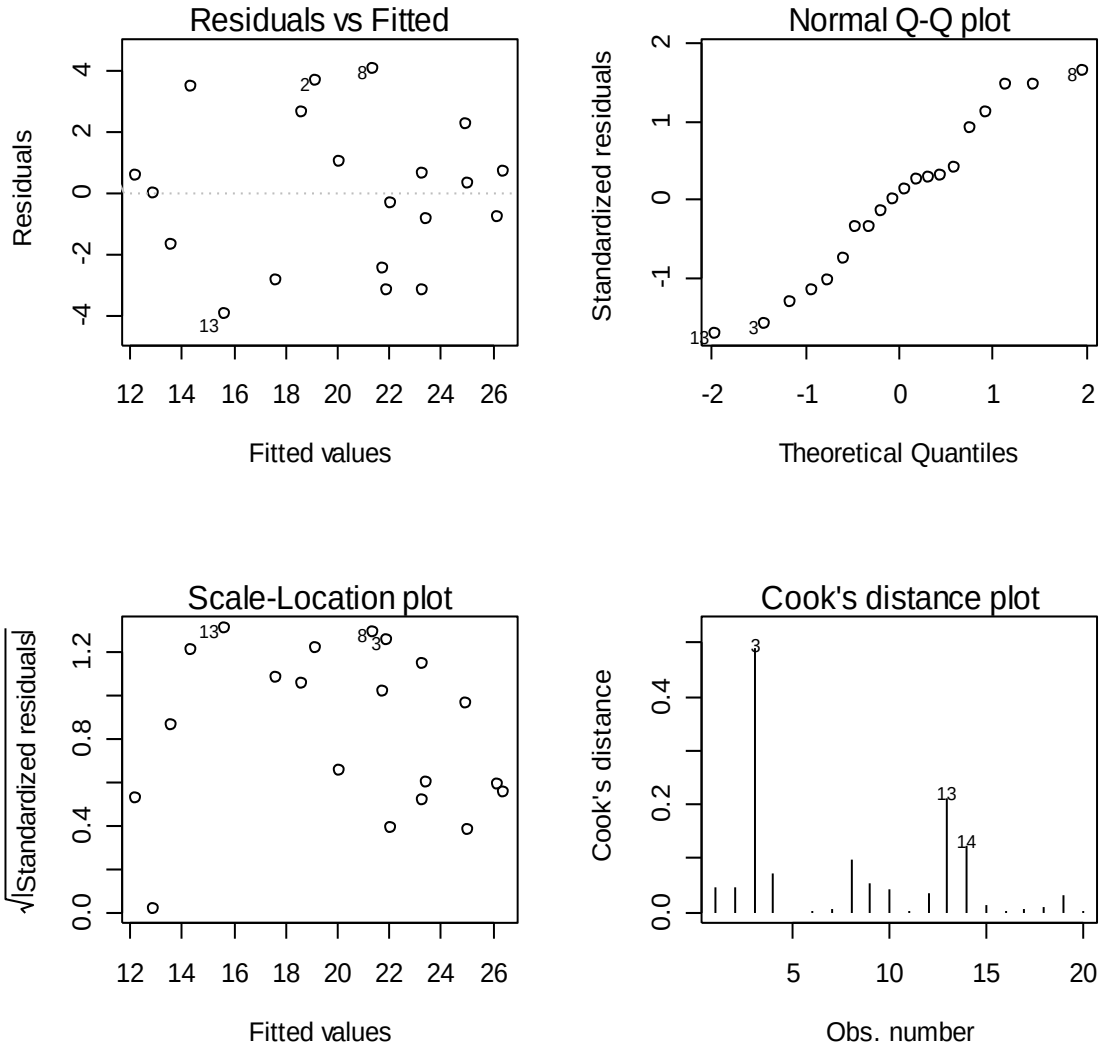


Fig. 5.1. **Four scatter plot graphs for the regression model**

6) Examine whether there are outlying *y* observations with two predictor variables (x_1, x_2).

Calculate the studentized residuals for the regression model:

```
> rstudent(lmResult)
          1           2           3           4
-0.7299854027  1.5342541325 -1.6543295725 -1.3484842072
          5           6           7           8
-0.0001269809 -0.1475490938  0.2981276214  1.7600924916
          9          10          11          12
 1.1176487404 -1.0337284208  0.1366610657  0.9231785040
         13          14          15          16
-1.8259027246  1.5247630510  0.2671500921  0.2581323416
         17          18          19          20
-0.3445090997 -0.3344080836 -1.1761712768  0.4093564171.
```

Cases 3, 8, and 13 have the largest absolute *studentized* residuals. Now we use the Bonferroni test procedure with a significance level α = .10 to test if case 13, which has the largest absolute studentized residual, is an outlier.

The function *qt*() (***q***uantile ***t***-distribution) gives the quantile value of *t*-distribution. The first argument is the significance level, the second one is the degrees of freedom.

```
>qt(.9975,16)
[1] 3.251993
```

The inference: since $|t_{13}| = 1.825 \leq 3.252$, we conclude that case 13 is not an outlier.

7) Identifying outlying observations.

Calculate the hat matrix for identifying outlying observations.

```
> hatvalues(lmResult)
         1          2          3          4          5
0.20101253 0.05889478 0.37193301 0.11094009 0.12861620
         6          7          8          9         10
0.24801034 0.15551745 0.09628780 0.11463564 0.11024435
        11         12         13         14         15
0.12033655 0.10926629 0.17838181 0.14800684 0.33321201
```

16	17	18	19	20
0.09527739	0.10559466	0.19679280	0.06695419	0.05008526.

The two largest leverage values are $h_{3,3} = .372$ and $h_{15,15} = .333$. Both exceed the criterion of the two mean leverage values, $2p/n = 2(3)/20 = .30$, and both are separated by a substantial gap from the next largest leverage value, $h_{55} = .248$ and $h_{11} = .201$. Having identified cases 3 and 15 as outlying in terms of their *x*-values, obtain the influence on the single fitted value – DFFITS:

```
> dffits(lmResult)
```

1	2	3	4
-3.661472e-01	3.838103e-01	-1.273067e+00	-4.763483e-01
5	6	7	8
-7.292347e-05	-5.668650e-02	1.279371e-01	5.745212e-01
9	10	11	12
4.021649e-01	-3.638725e-01	5.054583e-02	3.233366e-01
13	14	15	16
-8.507812e-01	6.355141e-01	1.888521e-01	8.376829e-02
17	18	19	20
-1.183735e-01	-1.655265e-01	-3.150707e-01	9.399706e-02.

The only DFFITS value that exceeds the guideline for a medium-size data set is for case 3, where $|(\text{DFFITS})_3| = 1.273$. This value is somewhat larger than the guideline of 1. However, the value is close enough to 1, so that the case may not be influential enough to require remedial action.

Obtain the influence on all fitted values – Cook's distance:

```
> cooks.distance(lmResult)
```

1	2	3	4
4.595055e-02	4.548118e-02	4.901567e-01	7.216190e-02
5	6	7	8
1.883399e-09	1.136518e-03	5.764939e-03	9.793853e-02
9	10	11	12
5.313352e-02	4.395704e-02	9.037986e-04	3.515436e-02
13	14	15	16
2.121502e-01	1.248925e-01	1.257530e-02	2.474925e-03
17	18	19	20
4.926142e-03	9.636470e-03	3.236006e-02	3.096787e-03.

Case 3 has the largest Cook's distance value, with the next largest distance measure D_{13} = .212 being substantially smaller. To assess the magnitude of the influence of case 3 (D_3 = .490), we refer to the corresponding *F*-distribution.

The function *pf*() (***p***robability of ***f***-distribution) gives the *F* distribution function. Here we set the probability = .490 to the first argument and the degrees of freedom 3 and 7 to the second and third arguments of function *qt*().

```
> pf(.490,3,17)
[1] 0.3061611.
```

The influence on the regression coefficients – *dfbetas*():

```
> dfbetas(lmResult)
```

	(Intercept)	BodyFat$$x_1$$	BodyFat$$x_2$$
1	-3.051821e-01	-1.314856e-01	2.320319e-01
2	1.725732e-01	1.150251e-01	-1.426129e-01
3	-8.471013e-01	-1.182525e+00	1.066903e+00
4	-1.016120e-01	-2.935195e-01	1.960719e-01
5	-6.372122e-05	-3.052747e-05	5.023715e-05
6	3.967715e-02	4.008114e-02	-4.426759e-02
7	-7.752748e-02	-1.561293e-02	5.431634e-02
8	2.614312e-01	3.911262e-01	-3.324533e-01
9	-1.513521e-01	-2.946556e-01	2.469091e-01
10	2.377492e-01	2.446010e-01	-2.688086e-01
11	-9.020885e-03	1.705640e-02	-2.484518e-03
12	-1.304933e-01	2.245800e-02	6.999608e-02
13	1.194147e-01	5.924202e-01	-3.894913e-01
14	4.517437e-01	1.131722e-01	-2.977042e-01
15	-3.004276e-03	-1.247567e-01	6.876929e-02
16	9.308463e-03	4.311347e-02	-2.512499e-02
17	7.951208e-02	5.504357e-02	-7.609008e-02
18	1.320522e-01	7.532874e-02	-1.161003e-01
19	-1.296032e-01	-4.072030e-03	6.442931e-02
20	1.019045e-02	2.290797e-03	-3.314146e-03.

Case 3 is the only case that exceeds the guideline of 1 for the medium-size data sets for both x_1 and x_2. Thus, case 3 is again tagged as potentially influential. However, the DFBETAS values do not exceed 1 by very much so that case 3 may not be so influential as to require remedial action.

Example 5.8.2. Chemical shipment. The data to follow, taken on 20 incoming shipments of chemicals in drums arriving at a warehouse, show the number of drums in the shipment (x_1), the total weight of the shipment (x_2, in hundred pounds), and the number of minutes required to handle the shipment (y).

Input the data:

> ChemiShip = scan("CH06PR09.DAT", list(y = 0, x_1 = 0, x_2 = 0))

Read 20 records. Show the data set:

```
> ChemiShip
$y
 [1] 58 152 41 93 101 38 203 78 117 44 121 112 50 82 48 127 140 155 39
[20] 90

$x1
 [1] 7 18 5 14 11 5 23 9 16 5 17 12 6 12 8 15 17 21 6 11

$x2
 [1] 5.11 16.72 3.20 7.03 10.98 4.04 22.07 7.03 10.62 4.76 11.02 9.51
[13] 3.79 6.45 4.60 13.86 13.03 15.21 3.64 9.57.
```

a) Obtain the studentized deleted residuals and identify any outlying y observations. Use the Bonferroni's outlier test procedure with $\alpha = 0.05$. State the decision rule and conclusion.

Save the result of the regression model:

> lmOut = lm (ChemiShip$y~ChemiShipx1+ChemiShipx2)

Obtain the studentized deleted residuals:

```
> rstudent(lmOut)
        1         2         3         4         5
0.42675254 -0.80047414 0.48150671 0.24531968 0.08475496
        6         7         8         9        10
-0.89457378 0.20652191 0.92718720 -0.10385591 -0.44555083
       11        12        13        14        15
-0.45241008 3.62623361 0.90389470 0.13072481 -1.75127470
       16        17        18        19        20
-0.60782399 1.22203071 -0.95432624 -1.02583519 -0.61414823.
```

In the upper list of the studentized deleted residuals, cases 12, 15, and 17 are the most outlying ones. We test case 12, which has the largest absolute studentized deleted residual (3.626) with $\alpha = 0.05$.

Use the function *qt*() (*q*uantiles of Student *t*-distribution) to calculate the quantile value with $\alpha = 0.05$. The probability of Student's *t*-distribution and the degrees of freedom are set to the arguments:

```
> qt(1-0.05/40,16)
[1] 3.580522
```

The inference: since $|t_{12}| = 3.626 > 3.580$, we conclude that case 12 is an outlier.

The second largest case $|t_{15}| = 1.75 < 3.58$, so this case is not an outlier.

b) Identify any outlying *x*-observations.

Obtain the hat matrix leverage:

```
> hatvalues(lmOut)
```

1	2	3	4	5
0.09133349	0.19376540	0.13099986	0.26847934	0.14900387
6	7	8	9	10
0.14056367	0.42868587	0.06651639	0.13453992	0.16452717
11	12	13	14	15
0.17857817	0.05138802	0.11031467	0.15597401	0.0953742
16	17	18	19	20
0.12815463	0.09698045	0.23049569	0.11180602	0.07251911.

The inference: the largest leverage value is $h_{77} = 0.429$. It exceeds the criterion of twice the mean leverage value, $2p/n = 2(3) / 20 = .30$. The next two largest leverage values are case 4 and case 18, but both of them are much smaller than the value of case 7. So we identify case 7 as an outlying *x* observation.

c) Management wishes to predict the number required to handle the next shipment containing $x_1 = 15$ drums whose total weight is $x_2 = 8$ (hundred pounds). Construct a scatter plot of x_2 against x_1 and determine visually whether this prediction involves an extrapolation beyond the range of the data.

Add the predictor values to x_1 and x_2:

```
> addx1 = c(ChemiShip$x1,15)
> addx2 = c(ChemiShip$x2, 8)
```

Construct a scatter plot of x_2 against x_1:

> plot (addx_1, addx_2)

R provides a useful function *identify*() to find the index of the closest (*x*; *y*) coordinates to the mouse click. The function *identify* reads the position of the graphics pointer when the (first) mouse button is pressed. It then searches the coordinates given in *x* and *y* for the point closest to the pointer. If this point is close to the pointer, its index will be returned as part of the value of the call.

Three arguments are set to the function: the coordinates of the points in a scatter plot and the number of the points we want to identify. We identify the three outliers and find the corresponding index:

> identify(addx_1, addx_2, n = 3)
[1] 16 21 4

In the obtained scatter plot, the predictor value case 21 falls within the range of the scatter area, so this prediction involves an extrapolation beyond the range of the data.

Read the data from the data frame separately:

> y = ChemiShip$y
> x_1 = ChemiShip$$x_1$
> x_2 = ChemiShip$$x_2$

Create a data frame for setting the estimation data set:

> new = data.frame(x_1 = 15, x_2 = 8)

Compute the prediction interval for all the regression data sets:

> predict(lm(y~x_1+x_2), interval = "prediction")

	fit	lwr	upr
1	55.65775	43.27632	68.03918
2	156.08097	143.13151	169.03043
3	38.41952	25.81509	51.02395
4	91.78733	78.43879	105.13587
5	100.54737	87.84301	113.25173
6	42.68637	30.02876	55.34399
7	202.09731	187.93088	216.26374
8	72.94678	60.70694	85.18662

9	117.55927	104.93512	130.18341
10	46.34368	33.55379	59.13357
11	123.35921	110.49239	136.22603
12	96.84849	84.69576	109.00121
13	45.18459	32.69595	57.67323
14	81.30495	68.56211	94.04779
15	56.83527	44.43094	69.23960
16	130.24902	117.66045	142.83759
17	133.56918	121.15576	145.98260
18	159.71512	146.56796	172.86229
19	44.42265	31.92563	56.91967
20	93.38515	81.11091	105.65939.

Compute the prediction interval for the estimated data set:

> predict(lm(y~x_1+x_2), new, interval = "prediction")

	fit	lwr	upr
[1,]	100.4826	87.1526	113.8127.

d) Case 7 appears to be an outlying *x*-observation and case 12 an outlying *y*-observation. Obtain the DFFITS, DFBETAS, and the Cook's distance values for each of these cases to assess their influence. With these, one can conclude the following.

Obtain the DFFITS:

> dffits(lmOut)

1	2	3	4	5
0.13529722	-0.39242323	0.18695102	0.14861906	0.03546501
6	7	8	9	10
-0.36178109	0.17889501	0.24750183	-0.04094805	-0.19771969
11	12	13	14	15
-0.21094213	0.84399998	0.31828507	0.05619611	-0.56863730
16	17	18	19	20
-0.23303720	0.40047521	-0.52230327	-0.36396220	-0.17173031.

The largest DFFITS value is 0.843 of case 12, but this value does not exceed 1, which is the guideline for the medium data set, so there is no influence case in this measurement.

Obtain the DFBETAS:

```
> dfbetas(lmOut)
```

	(Intercept)	ChemiShip$$x_1$$	ChemiShip$$x_2$$
1	0.122855552	-0.04327413	0.0098720883
2	0.061012751	0.14958419	-0.2528778589
3	0.176732390	-0.05463481	-0.0008834814
4	-0.025976645	0.13138441	-0.1317456914
5	0.017823424	-0.02787250	0.0287067303
6	-0.351927749	0.19100547	-0.0943824289
7	-0.060748101	-0.05685992	0.1123483643
8	0.198609085	-0.08507044	0.0451016524
9	0.012636770	-0.03157524	0.0264043658
10	-0.187199214	0.13499584	-0.0892584612
11	0.080846469	-0.17385681	0.1449559800
12	0.353791892	-0.12188051	0.1378580288
13	0.286042063	-0.05861754	-0.0316941000
14	0.001831733	0.04311705	-0.0463174805
15	-0.370513959	-0.12638404	0.2571081607
16	-0.021673104	0.11848066	-0.1618347451
17	-0.138884614	0.16427966	-0.0672002726
18	0.312765306	-0.35456723	0.2165300133
19	-0.320519790	0.04866628	0.0553440075
20	-0.099571789	0.09480214	-0.0927223402

The inference: there is no case's DFBETAS value that exceeds guide-line 1, so no case influences the regression model.

Obtain the Cook's distance:

```
> cooks.distance(lmOut)
          1           2           3           4
0.0064101777 0.0524401531 0.0122015599 0.0077933871
          5           6           7           8
0.0004452592 0.0441472158 0.0113044124 0.0205890007
          9          10          11          12
0.0005934465 0.0136757853 0.0155601581 0.1384778737
```

13	14	15	16
0.0341358732	0.0011172659	0.0960985497	0.0187994039
17	18	19	20
0.0519524153	0.0914135514	0.0440206341	0.0102042828.

Case 12 has the largest Cook's distance value 0.1385, the next largest distance D_{15} = 0.096 is substantially small. Let's test the influence of case 12.

Compute the F-distribution function using *pf* (***p***robability ***f*** distribution):

```
> pf(0.1385, 3,17)
[1] 0.06439451
```

Case 12 is the 6th percentile of this distribution, so it does not have any influence on the fitted values.

e) Calculate the Cook's distance D_i for each case and prepare an index plot. Are any cases influential according to this measure?

Construct an index plot by using the plot function. The argument is set as the result of the regression *lmOut*. The output of the plot function is 4 separate graphs. You can see the graphs sheet by sheet by pressing the enter button. The index plot of the Cook's distance is the last graph (Fig. 5.2):

```
> plot(lmOut)
```

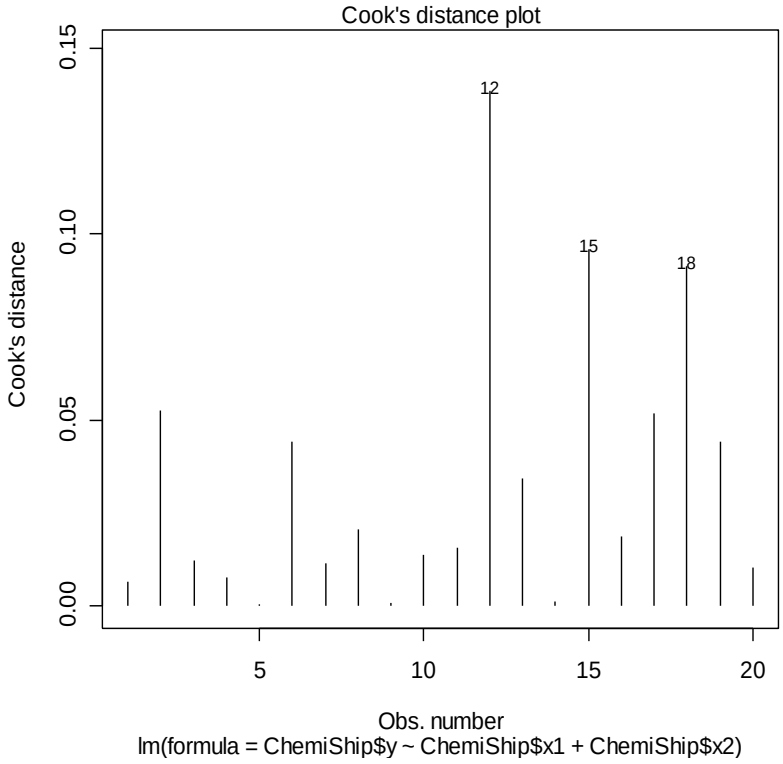


Fig. 5.2. **The index plot for the Cook's distance**

The inference: in this index plot, the most influential case is case 12, and cases 15 and 18 are two next largest influential ones to this regression model.

Example 5.8.3. Cosmetics sales. An assistant in the district sales office of a national cosmetics firm obtained data, shown below, on advertising expenditures and sales last year in the district's 14 territories. X_1 denotes expenditures for point-of-sale displays in beauty salons and department stores (in thousand dollars), and X_2 and X_3 represent the corresponding expenditures for local media advertising and the prorated share of national media advertising, respectively. Y denotes sales (in thousand cases). The assistant was instructed to estimate the increase in the expected sales when X_1 is increased by 1 thousand dollars and X_2 and X_3 are held constant, and was told to use an ordinary multiple regression model with a linear term for the predictor variables and with independent normal error terms:

i	Yi	X_1i	X_2i	X_3i
1	8.26	4.2	4.0	3.0
2	14.70	6.5	6.5	5.0
3	9.73	3.0	3.5	4.0
4	5.62	2.1	2.0	3.0
5	7.84	2.9	3.0	4.0
6	12.18	7.2	7.0	3.0
7	8.56	4.8	5.0	4.5
8	10.77	4.3	4.0	5.0
9	7.56	2.6	2.5	5.0
10	8.90	3.1	3.0	4.0
11	12.51	6.2	6.0	4.5
12	10.46	5.5	5.5	5.0
13	7.15	2.2	2.0	4.0
14	6.74	3.0	2.8	3.0

a) State the regression model to be employed and fit it to the data.

Input the data. The data was saved in the file *CH09PR13.DAT*:

> CosmeticsSales = scan ("CH09PR13.DAT", list(Y = 0, X_1 = 0, X_2 = 0, X_3 = 0))

Read 14 records:

```
> CosmeticsSales
$Y
 [1]  8.26 14.70  9.73  5.62  7.84 12.18  8.56 10.77  7.56  8.90 12.51
[12] 10.46 7.15  6.74
[13]

$X1
 [1] 4.2 6.5 3.0 2.1 2.9 7.2 4.8 4.3 2.6 3.1 6.2 5.5 2.2 3.0

$X2
 [1] 4.0 6.5 3.5 2.0 3.0 7.0 5.0 4.0 2.5 3.0 6.0 5.5 2.0 2.8

$X3
 [1] 3.0 5.0 4.0 3.0 4.0 3.0 4.5 5.0 5.0 4.0 4.5 5.0 4.0 3.0.
```

Build the regression model and save the regression result:

```
>lm_Result =
+lm(CosmeticsSales$Y~CosmeticsSales$X1+CosmeticsSales$X2+Cosmetics
Sales$X3)
```

Summary results:

```
> summary(lm_Result)
```

The call:

```
lm(formula = CosmeticsSales$Y ~ CosmeticsSales$X1 + CosmeticsSales$X2 +
+ CosmeticsSales$X3)
```

The residuals:

Min	1Q	Median	3Q	Max
-2.20899	-0.30679	-0.04512	0.55678	1.58152

The coefficients:

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	0.9796	1.7270	0.567	0.5831
CosmeticsSales$x1	0.4096	1.5423	0.266	0.7959
CosmeticsSales$x2	0.8300	1.5633	0.531	0.6071
CosmeticsSales$x3	0.8163	0.4144	1.970	0.0771 .

The inference: the residual standard error: 1.152 on 10 degrees of freedom; the multiple *R*-squared: 0.8402; the adjusted *R*-squared: 0.7922; *F*-statistic: 17.52 on 3 and 10 DF; the *p*-value: 0.0002627.

As a result of the investigation, the fitted regression model is as follows:

$$Y = 0.9796 + 0.4096\ X_1 + 0.83\ X_2 + 0.8163\ X_3.$$

b) Test whether there is a regression relation between sales and the three predictor variables; use the $\alpha = .05$ significance level.

Chose a model using the R command *step*():

> step(lm_Result)

Start: AIC = 7.26.

CosmeticsSales$Y ~ CosmeticsSales$$X_1$ + CosmeticsSales$$X_2$ + + CosmeticsSales$$X_3$.

Result:

	Df	Sum of Sq	RSS	AIC
CosmeticsSales$$x_1$	1	0.0936	13.3699	5.3553
CosmeticsSales$$x_2$	1	0.3742	13.6505	5.6460
CosmeticsSales$$x_3$	1	5.1524	18.4287	9.8479

Step: AIC = 5.36.

CosmeticsSales$Y ~ CosmeticsSales$$X_2$ + CosmeticsSales$$X_3$

	Df	Sum of Sq	RSS	AIC
CosmeticsSales$$x_3$	1	5.065	18.434	7.852
CosmeticsSales$$x_2$	1	52.989	66.359	25.784

The call function:

lm(formula = CosmeticsSales$Y ~ CosmeticsSales$$X_2$ + CosmeticsSales$$X_3$).

The coefficients:

(Intercept)	CosmeticsSales$$X_2$	CosmeticsSales$$X_3$
1.0690	1.2419	0.7978

The inference: based on the upper result, the final model obtained is a two-variable model:

$$Y = 1.0690 + 1.2419X_2 + 0.7978X_3.$$

c) Test each of the regression coefficients β_k (k = 1, 2, 3) individually whether or not β_k = 0, use α = .05 each time. Do the conclusions of these tests correspond to those obtained in part b?

Occasionally an experimenter might suspect that a particular predictor variable is not really very useful. To decide whether or not this is the case, we test the null hypothesis that the coefficient for this variable is 0. That is, we test:

Test if $\beta_1 = 0$:

> summary(lm(formula = CosmeticsSales$Y ~ CosmeticsSales$X$_1$))

The call function:

lm(formula = CosmeticsSales$Y ~ CosmeticsSales$X$_1$).

The residuals:

Min	1Q	Median	3Q	Max
-1.6940	-1.1413	0.1314	0.7602	2.2192

The coefficients:

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	3.9663	0.9292	4.268	0.00109
CosmeticsSales$X$_1$	1.3099	0.2101	6.236	4.34e-05

The results:

the residual standard error: 1.278 on 12 degrees of freedom;

the multiple R-Squared: 0.7642; the adjusted R-squared: 0.7445;

F-statistic: 38.89 on 1 and 12 DF; the p-value: 4.343e-05.

The inference: based on the upper testing result, pt = 0.0000434, it is much smaller than 0.05/2, so this hypothesis is rejected.

The predictor variable X_1 is needed in the model that contains the other predictor variables.

Test if $\beta_2 = 0$.

> summary(lm(formula = CosmeticsSales$y ~ CosmeticsSales$x2))

The call function:

lm(formula = CosmeticsSales$Y ~ CosmeticsSales$X$_2$).

The residuals:

Min	1Q	Median	3Q	Max
-2.0523	-0.9807	0.0863	0.8561	2.0887

The coefficients:

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	3.9488	0.8970	4.402	0.000862
CosmeticsSales$x	2 1.3327	0.2055	6.487	2.99e-05

The results:

the residual standard error: 1.239 on 12 degrees of freedom;

the multiple *R*-squared: 0.7781; the adjusted *R*-squared: 0.7596;

F-statistic: 42.08 on 1 and 12 *DF*; the *p*-value: 2.994e-05.

The inference: based on the upper testing result, *pt* = 0.00002.99, it is smaller than .05/2, so the hypothesis is rejected.

The conclusion: the predictor variable X_2 is useful in predicting the value of the response.

Test if $\beta 3 = 0$:

> summary(lm(formula = CosmeticsSales$Y ~ CosmeticsSales$$X_3$))

The call function:

lm(formula = CosmeticsSales$Y ~ CosmeticsSales$$X_3$).

The residuals:

Min	1Q	Median	3Q	Max
-3.1033	-1.4112	-0.2792	0.4594	4.3331

The coefficients:

	Estimate	Std. Error	*t*-value	*Pr*(>|t|)
(Intercept)	3.622	3.357	1.079	0.302
CosmeticsSales$$X_3$	1.408	0.810	1.739	0.108

The results:

the residual standard error: 2.352 on 12 degrees of freedom;

the multiple *R*-squared: 0.2012; the adjusted *R*-squared: 0.1346;

F-statistic: 3.023 on 1 and 12 *DF*; the *p*-value: 0.1077.

The inference: we get the *p*-value from this result: *pt* = 0.108, it is greater than 0.05/2, so we cannot reject this hypothesis.

The conclusion: the predictor variable X_3 is not useful in predicting the value of the response *Y*. It is not needed in the model that contains the other predictor variables.

d) Obtain the correlation matrix of the *X* variables. Create the data frame for the variables:

>Cosmetics = data.frame(X_1 = CosmeticsSales$$X_1$, X_2 = CosmeticsSales$$X_2$, X_3 = CosmeticsSales$$X_3$)

Use the function *cor*() (***cor***relation) to obtain the correlation matrix of the *X* variables:

> cor(Cosmetics)

	X_1	X_2	X_3
X_1	1.0000000	0.9922085	0.2143812
X_2	0.9922085	1.0000000	0.2365410
X_3	0.2143812	0.2365410	1.0000000.

6. Power analysis

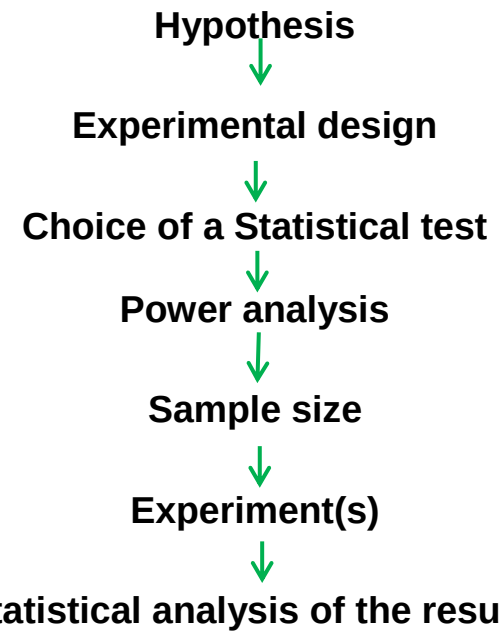
First, the definition of power: it is the probability of detecting a specified effect at a specified significance level. Now this specified effect refers to the effect size which can be the result of an experimental manipulation or the strength of a relationship between 2 variables. And this effect size is "specified" because prior to the power analysis we should have an idea of the size of the effect we expect to see. The "probability of detecting" a bit refers to the ability of a test to detect an effect of a specified size. The recommended power is 0.8 which means we have an 80 % chance of detecting an effect if one truly exists.

The main output of a power analysis is the estimation of a sufficient sample size. This is of pivotal importance of course. If our sample is too big, it is a waste of resources; if it is too small, we may miss the effect ($p > 0.05$) which would also mean a waste of resources.

From a more practical point of view we need to justify our sample size which we can do through a power analysis. Finally, it is all about the ethics of research, which is encapsulated in the Home office's 3 R: Replacement, Refinement and Reduction.

The latter in particular relates directly to power calculation as it refers to the methods which minimize animal use and enable the researcher to obtain comparable levels of information from fewer animals' (NC3Rs website).

When should we run your power analysis? It depends of what we expect from it: the most common output being the sample size, we should run it before doing the actual experiment (*a priori* analysis). The correct sequence from hypothesis to results should be:



The power analysis depends on the relationship between 6 variables: the effect size of biological interest, the standard deviation, the significance level, the desired power, the sample size and the alternative hypothesis.

The significance level is about the *p*-value, it is generally agreed to be 5 % and we will come back to it later. The desired power, as mentioned earlier, is usually 80 %. The alternative hypothesis is about choosing between one and 2-sided tests, it is a technical thing and we will come back to it later as well. So we are left with the 3 variables on which we have pretty much no control or about which we cannot decide arbitrarily: the effect size, the sample size and the standard deviation. To help understand what they are and how much they are connected, here is an example.

Let's make it simple, say we are studying a gene which is expressed in the brain and we are using a mouse model. Our hypothesis is that knocking out (KO) that gene will affect the mouse's behavior. The next step is to design the experiment. We are going to create a KO mouse in which gene A is inoperative and we are going to compare WT (wild type) and KO mice's behavior through a set of tasks.

Let's say that the output of one of these tasks is a quantitative variable, the time taken by the mouse to achieve one task for example. Now we need to translate the hypothesis for this particular task into a statistical question. The hypothesis is that knocking out gene A will affect KO mice behavior which can be quantified by a change in the time taken to achieve the task.

Statistically we need to know: what type of data we are going to collect (time), which test we are going to use to compare the 2 genotypes and how many mice we will need. When thinking about the sample size, it is very important to consider the difference between technical and biological replicates. Technical replicates involve taking several samples from one tube and analyzing them across multiple conditions. Biological replicates are different samples measured across multiple conditions.

First, the *sample size*, the name itself is self-explanatory. The aim of a power analysis is usually to find the appropriate sample size as in the one which will allow us to detect a specified effect. This effect, also called *effect size* of biological interest, can only be determined scientifically not statistically. It is either a difference that would be meaningful biologically, like an increase/ decrease of 10 % or 20 % for a particular variable, or what we expect to get based on preliminary data. The larger the effect size, the smaller the experiment will need to be to detect it.

The Standard Deviation (SD) is basically the noise in our data, the variability we observe between our values. This we get ideally from a pilot study or from previous experiments or even the literature.

Now going back to the effect size, there are actually 2 different ones: the absolute one which is basically the difference between the mean value of, say, our control group and the one of our treatment group, and the relative one, also referred to as Cohen's *d*. This one is the more useful and more widely used one as it accounts for the variability in our data.

Cohen's *d*:

$$\mathrm{d}=\frac{\mathrm{Mean1\text{-}Mean2}}{\mathrm{Pooled\ SD}}.$$

The *significance level* refers to the famous *p*-value which, for a test to be significant, should be below 0.05 (the 5 % threshold). Going back to our experiment about finding out more on gene A, we would define the *p*-value as the probability that a difference as big as the one observed between the WT and the KO could be found even if the knock-out of gene A does not affect the mouse's behavior.

Basically it means that if we find a significant difference ($p < 0.05$) between our 2 groups of mice, so corresponding to the effect size of biological interest, there is less than 5 % chance that we would have been able to observe it if the knocking out of gene A did not affect the behavior of the mouse.

The last variable is the so-called *alternative hypothesis*: a one- or two-sided test. This refers to the distribution of our variable: are we going to look at one side or at the two sides of it. In other words, and again going back to our experiment, do we want to answer the question: does it take longer to KO mice to achieve the task or do we simply want to know if there is a difference at all.

Most of the time, in bench science, we go for the second question, even though we might think of one direction more than the other. We don't have enough evidence to choose to look only at one side of the distribution. It is pretty much only in clinical trials that people go for one-sided tests. They have already tested a particular drug for example in many systems/species so they have plenty of evidence about the effect of the drug.

Finally, it is 2 times easier to reach significance with a one-side test than with a two-side one so a reviewer will always be suspicious if we go for the first one and if he asked for justification, we'd better have one!

The basic idea behind the power analysis is that if we fix any five of the variables, a mathematical relationship can be used to estimate the sixth. So going back one more time to our example, running the power analysis, our question can be: What *sample size* do I need to have an 80 % probability (*power*) to detect an average 5 minutes difference (*effect size* and *standard deviation*) at a 5 % *significance level* using a *two-sided test*? The variables are all linked and will vary as shown in the diagram (Fig. 6.1).

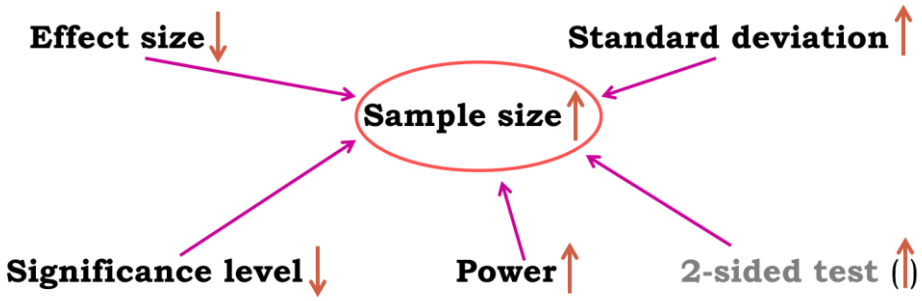


Fig. 6.1. **The power of the *t*-test diagram.**

Now here is the good news, there are packages that can do the power analysis for us, providing of course we have some prior knowledge of the key

parameters. Mostly we need to have some idea of the difference we are expecting to see or that would make sense, together with some information on the standard deviation.

About power analysis, the important message is: after we have designed your experiment, run a power analysis to estimate the appropriate sample size that will allow us to do good and ethical science.

7. Qualitative data

Packages needed for the course can be downloaded by the following command:

```
>install.packages(c("beanplot","gmodels","plotrix","car","pastecs","reshape2"))
```

Let's talk about the important stuff: your data. The first thing you need to do good stats is to know your data inside out. They are generally organized into variables, which can be divided into 2 categories: *qualitative* and *quantitative* and in this chapter we will only look at the former.

Qualitative data are non-numerical data and the values taken can be names (also *nominal* data, e.g. causes of death in a hospital). The values can also be numbers but not numerical (e.g. an experiment number is a numerical label but not a unit of measurement). A qualitative variable with intrinsic order in their categories is *ordinal*. Finally, there is the particular case of qualitative variable with only 2 categories, it is then said to be *binary* or *dichotomous* (e.g. alive/dead or male/female).

We are going to use an example to go through the analysis and the plotting of categorical data.

Example 7.1. (File: *cats.dat*, for example.)

A researcher is interested in whether cats could be trained to line dance. He tries to train them to dance by giving them either food or affection as a reward (training) for dance-like behavior.

At the end of the week a note is made of which animal could line dance and which could not (dance). All the variables are dummy variables (categorical).

The pivotal question is: Is there an effect of training on cats' ability to learn to line dance? You have already designed your experiment and chosen your statistical test: it will be a Fisher's exact test (or a Chi-square test) and the power analysis with qualitative data.

The next step is to run a power analysis. In an ideal world, you would have run a pilot study to get some idea of the type of effect size you are expecting to see. Let's start with this ideal situation. Let's say, in your pilot study, you found that 30 % of the cats did line dance after received affection and 70 % did so after received food.

So we want to compare 2 proportions (0.3 and 0.7), we will be using the function *power.prop.test*() from R:

```
> power.prop.test(p1 = .3, p2 = .7, power = .8, sig.level = 0.05).
```

R tells us that we need 2 samples of 23 cats to reach a power of 80 %. In other words: if we want to be at least 80 % confident to spot a reward effect, if indeed there is one, we will need about 46 cats altogether.

Next, the experiment is run and the data collected:

```
cats.data<-read.table("cats.dat", sep = "\t",header = T).
```

It is always worth having a quick look at the data:

```
> head(cats)
> View(cats)
```

The next step, plotting the data:

```
> plot(cats.data$Training, cats.data$Dance, xlab = "Training", ylab = "Dance")
```

The result of applying this function is presented in Fig. 7.1.

The categorical data are sometimes best presented as contingency tables:

```
> table(cats.data)
```

The result:

```
                        Dance
Training                 No Yes
  Affection as Reward   114  48
  Food as Reward         10  28
```

Percentages can be more informative than raw data. The second line gives the values as percentage integers:

```
> contingency.table <- table(cats.data)
> contingency.table100<-prop.table(contingency.table,1)
> contingency.table100<-round(contingency.table100*100)
> contingency.table100
```

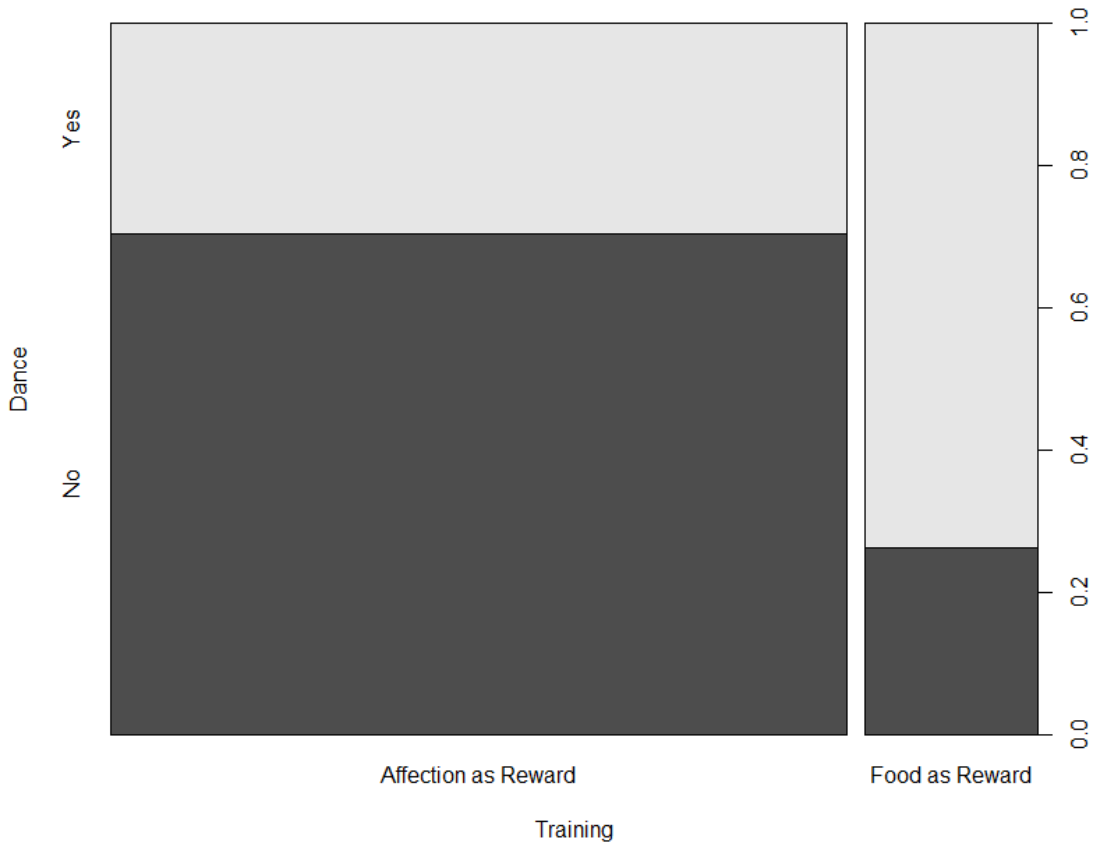


Fig. 7.1. **The initial categorical data**

The result:

Training	Dance No	Yes
Affection as Reward	70	30
Food as Reward	26	74

Then we can plot the data as percentage/proportion:

```
> plot(contingency.table100, col=c("white","darkgrey"),cex.axis=1)
```

The result of applying this function is presented in Fig. 7.2.

As mentioned before, to analyze such data we need to use a Fisher's exact test but we could also use a Chi^2 (χ^2) test.

Both tests will give us the sameish *p*-value for big samples but for small samples the difference can be more important and the *p*-value given by Fisher's exact test is more accurate. Having said that, the calculation of the Fisher's exact test is quite complex whereas the one for χ^2 is quite easy so only the calculation of the latter is going to be presented here. Also, the Fisher's test is often only available for 2x2 tables, so in a way the χ^2 is more general.

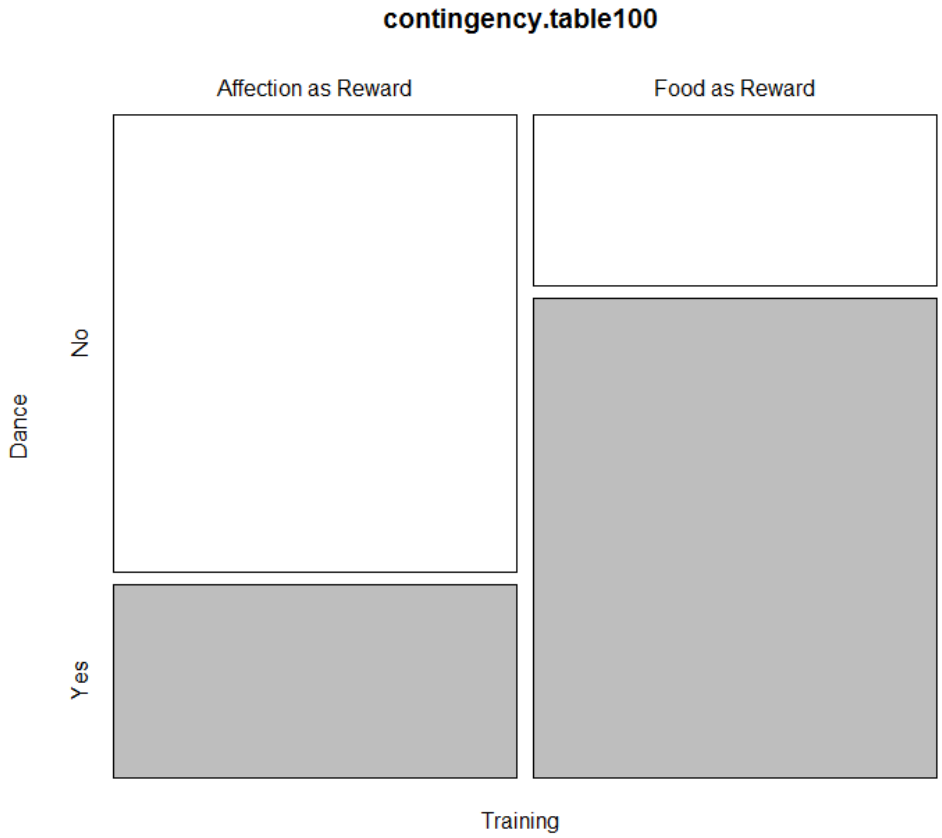


Fig. 7.2. **The data as percentage/proportion**

For both tests, the idea is the same: how different are the observed data from what we would have expected to see by chance i.e. if there were no association between the 2 variables. Or, looking at the table we may also ask: knowing that 76 of the 200 cats did dance and that 162 of them received affection, what is the probability that those 76 dancers would be so unevenly distributed between the 2 types of reward?

8. The null hypothesis and the error types

The null hypothesis (H_0) corresponds to the absence of effect (e.g. the animals rewarded by food are as likely to line dance as the ones rewarded by affection) and the aim of a statistical test is to accept or to reject H_0.

As mentioned earlier, traditionally, a test or a difference are said to be significant if the probability of Type I error is: $\alpha \leq 0.05$ (max $\alpha = 1$). It means that the level of uncertainty of a test usually accepted is 5 %.

It also means that there is a probability of 5 % that we may be wrong when we say that our two means are different, for instance, or we can say that when we see an effect, we want to be at least 95 % confident that something is significantly happening.

Summing: if our *p*-value is between 5 % and 10 % (0.05 and 0.10), one would not reject it too fast.

It is often worth putting this result into perspective and asking ourselves a few questions like:

What does the literature say about what I am looking at?

What if I had a bigger sample?

Have I run other tests on similar data and were they significant or not?

The interpretation of a border line result can be difficult, so it is important to look at the whole picture.

The specificity and the sensitivity of a test are closely related to Type I and Type II errors.

Specificity = Number of True Negatives / (Number of False Positives + + Number of True Negatives). A test with a high specificity has a low Type I error rate.

Sensitivity = Number of True Positives / (Number of False Negatives + + Number of True Positives). A test with a high sensitivity has a low Type II error rate.

9. The chi-squared test

It could be either: A) a one-way χ^2 test, which is basically a test that compares the observed frequency of a variable in a single group with what would be the expected by chance or B) a two-way χ^2 test, the most widely used, in which the observed frequencies for two or more groups are compared with expected frequencies by chance. In other words, in this case, the χ^2 tells you whether or not there is an association between two categorical variables.

An important thing to know about the χ^2, and for the Fisher's exact test for that matter, is that it does not tell us anything about causality; it is simply measuring the strength of the association between two variables and it is our knowledge of the biological system we are studying which will help us to interpret the result. Hence, we generally have an idea of which variable is acting the other.

The χ^2 value is calculated using the formula below:

$$\chi^2 = \frac{(Observed\ frequency - Expected\ frequency)^2}{Expected\ frequency}$$

The observed frequencies are the ones we measured, the values that are in our table. Now, the expected ones are calculated this way:

Expected frequency = (row total)*(column total)/grand total.

So, for example: the expected frequency of cats that would line dance after having received food as reward is:

(76*38)/200 = 14.44.

We can also think in terms of probabilities:

probability of line dancing: 38/200,
probability of receiving food: 76/200.

If the two events are independent, the probability of them occurring at the same time (the expected frequency) will be: 38/200*76/200 = 0.072 and 7.2 % of 200 is 14.4.

Intuitively, one can see that we are kind of looking for 50/50 (random output) results but accounting for the counts we have. If we work out the values for all the cells, we get the following.

To run a χ^2 analysis, we are going to use the *chisq.test*() function. Now, if we use the default version, R will give us the *p*-value with Yates continuity correction. All corrections for statistical tests work the same way: they increase the *p*-value. The reason is that if we are applying a correction, it is because we are using a test on data that do not meet the assumptions for it. Misusing a statistical test means that the output (i.e. the *p*-value) should not be trusted and as a consequence it is very likely that there is an increase of the probability of making the Type I error. So the solution is to increase the *p*-value, hence making it more difficult to reach significance thus reducing the probability of making the Type I error.

There is only one assumption that we have to be careful about when we run a χ^2: with 2x2 contingency tables we should not have cells with an expected count below 5 as if it is the case, it is likely that the test is not accurate (for larger tables, all expected counts should be greater than 1 and no more than 20 % of expected counts should be less than 5).

If we remember the χ^2s formula, the calculation gives us an estimation of the difference between our data and what we would have obtained if there was no association between our variables. Clearly, the bigger the value of the χ^2, the bigger the difference between observed and expected frequencies and the more likely the difference is to be significant.

Now with a 2x2 table, the way to go is usually a Fisher's exact test:

> fisher.test().

As we can see here, the *p*-values vary slightly between the two-sided tests (*p* = 1.31e-06 vs *p* = 4.77e-07) though the conclusion remains the same: cats only care about food. Though the samples are not very big here, the assumption for the χ^2 is met so you can choose either test.

10. Quantitative data

Packages needed for this chapter: *plotrix*, *car*, *pastecs*, *beanplot*, and r*eshape2*.

When it comes to quantitative data, more tests are available but assumptions must be met before applying them. In fact, there are two types of statistical tests: parametric and non-parametric ones. Parametric tests have four assumptions that must be met for the tests to be accurate. Non-parametric tests are based on ranks and they make few or no assumptions about population parameters like normality (e.g. Mann-Whitney test).

10.1. Descriptive statistics

The median: The median is the value exactly in the middle of an ordered set of numbers.

Example 1: 18 27 34 52 54 59 61 68 78 82 85 87 91 93 100, Median = 68
Example 2: 18 27 27 34 52 52 59 61 68 68 85 85 85 90, Median = 60

10.2. The mean

On average, μ = *average* of all values in a column. It can be considered as a model because it summaries the data.

Example 10.1. The number of friends of each member of a group of 5 lecturers: 1, 2, 3, 3 and 4. Mean: (1+2+3+3+4)/5 = 2.6 friends per lecturer: clearly a hypothetical value!

Now, if the values were: 1, 1, 1, 1 and 9 the mean would also be 2.6 but clearly it would not give an accurate picture of the data. So, how can we know that it is an accurate model? We look at the difference between the real data

and our model in Fig. 10.1. To do so, we calculate the difference between the real data and the model created and we make the sum so that we get the total error (or sum of differences).

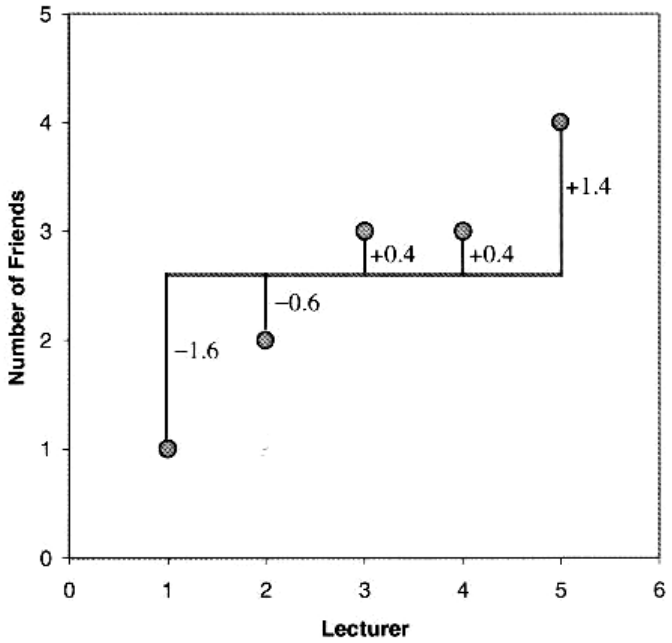


Fig. 10.1. **The difference between the real data and the model**

The difference is:

$$_i(x_i - \mu) = (-1.6) + (-0.6) + (0.4) + (0.4) + (1.4) = 0.$$

And we get no errors! Of course, positive and negative differences cancel each other out. So to avoid the problem of the direction of the error, we can square the differences and instead of the sum of errors, we get the Sum of Squared errors (SS). In our example:

$$SS = (-1.6)^2 + (-0.6)^2 + (0.4)^2 + (0.4)^2 + (1.4)^2 = 5.20.$$

10.3. The variance

This SS gives a good measure of the accuracy of the model but it is dependent upon the amount of data: the more data, the higher the SS. The solution is to divide the SS by the number of observations (*N*). As we are interested in measuring the error in the sample, to estimate the one in the population, we divide the SS by *N*-1 instead of *N* and we get the *variance* (S^2) = SS/(*N*-1). In our example: Variance (S^2) = 5.20 / 4 = 1.3. Why *N*-1 instead of *N*?

If we take a sample of 4 scores in a population, they are free to vary but if we use this sample to calculate the variance, we have to use the mean of

the sample as an estimate of the mean of the population. To do that, we have to hold one parameter constant.

Example 10.2. The mean of the sample is 10. We assume that the mean of the population from which the sample has been collected is also 10. If we want to calculate the variance, we must keep this value constant which means that the 4 scores cannot vary freely. If the values are 9, 8, 11 and 12 (mean = 10) and if we change 3 of these values to 7, 15 and 8, then the final value must be 10 to keep the mean constant.

If we hold 1 parameter constant, we have to use N-1 instead of N. It is the idea behind the *degree of freedom*: one less than the sample size.

10.4. The standard deviation (S.D.)

The problem with the variance is that it is measured in squared units, which is not very nice to manipulate. So for more convenience, the square root of the variance is taken to obtain a measure in the same unit as the original measure: the *standard deviation:* S.D. = $\sqrt{SS/(N-1)} = \sqrt{(S^2)}$, in our example: S.D. = $\sqrt{(1.3)} = 1.14$. So you would present your mean as follows: $\mu = 2.6 \pm 1.14$ friends.

The standard deviation is a measure of how well the mean represents the data or how much our data are scattered around the mean. In Fig. 10.2 the difference between the low and high standard deviations is shown. Small S.D.: data close to the mean: mean is a good fit of the data (graph on the left). Large S.D.: data distant from the mean: mean is not an accurate representation (graph on the right)

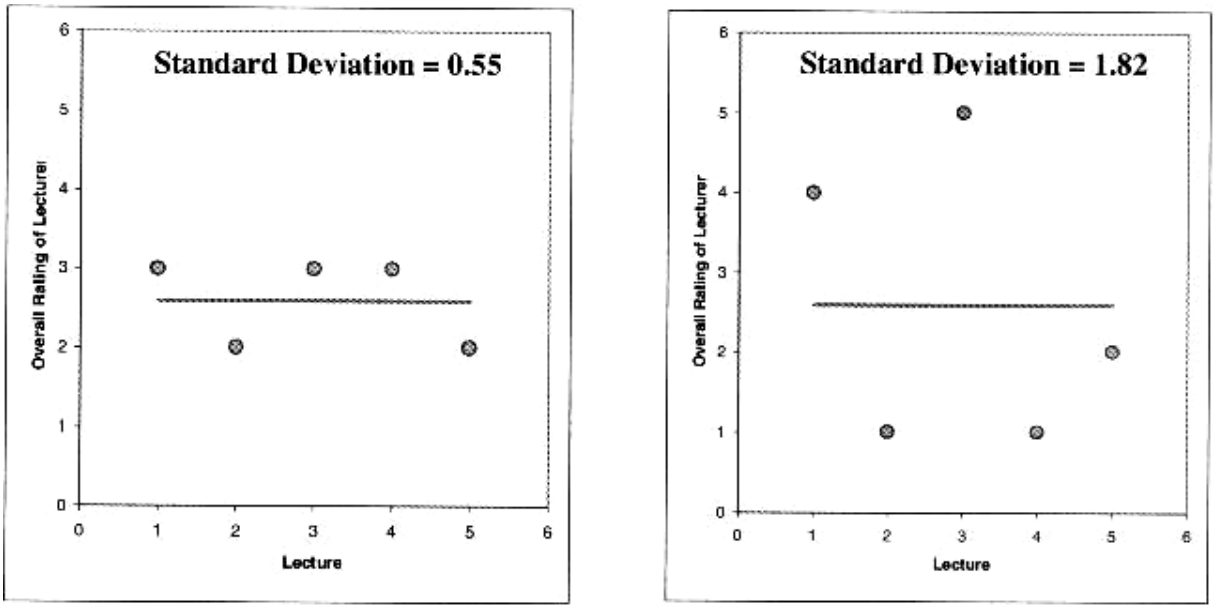


Fig. 10.2. **The low and high standard deviations**

10.5. Standard deviation vs standard error

Many are confused about the difference between the standard deviation (S.D.) and the *standard error of the mean* (S.E.M. = S.D. / √N). This difference is represented in Fig. 10.3.

The S.D. (the graph on the left) quantifies the scatter of the data and increasing the size of the sample does not decrease the scatter (above a certain threshold).

The S.E.M. (the graph on the right) quantifies how accurately we know the true population mean, it's a measure of how much we expect the sample means to vary. So the S.E.M. gets smaller as our samples get larger: the mean of a large sample is likely to be closer to the true mean than is the mean of a small sample.

A small S.E.M. means that most sample means are similar to the population mean and so our sample is likely to be an accurate representation of the population.

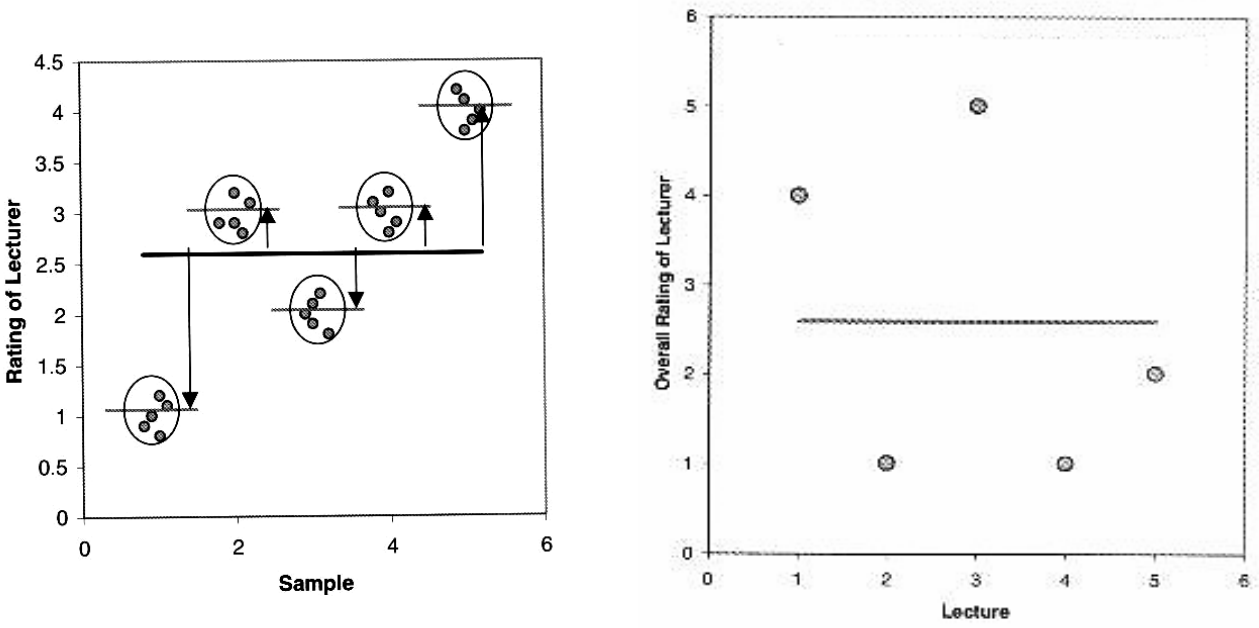


Fig. 10.3. **The difference between the S.D. (left) and S.E.M. (right)**

10.6. Which error measure to choose?

The choice depends, first of all, on the subject researched. If the scatter is caused by biological variability, it is important to show the variation. So it is more appropriate to report the S.D. rather than the S.E.M. Even better, we

can show in a graph all data points, or perhaps report the largest and smallest value.

If we are using an in vitro system with theoretically very little biological variability, the scatter can only result from experimental imprecision (no biological meaning). It is more sensible then to report the S.E.M. since the S.D. is less useful here. The S.E.M. gives the readers a sense of how well we have determined the mean.

Choosing between S.D. and S.E.M. also depends on what we want to show. If we just want to present our data on a descriptive purpose, then we go for the SD. If we want the reader to be able to infer an idea of significance, then you should go for the SEM or the confidence interval (CI). We will go a bit more in details later.

10.7. The confidence interval

The confidence interval quantifies the uncertainty in measurement. The mean we calculate from our sample of data points depends on which values we happened to sample. Therefore, the mean we calculate is unlikely to equal the true population mean. The size of the likely discrepancy depends on the variability of the values and the sample size. If we combine those together, we can calculate a 95 % confidence interval, which is a range of values. If the population is normal (or nearly so), there is a 95 % chance that the confidence interval contains the true population mean (*pop.mean*). For example, 95 % of observations in a normal distribution lie within *pop.mean* $\pm$ 1.96*S.E.

One other way to look at error bars is shown in Fig. 10.4 and Table 10.1.

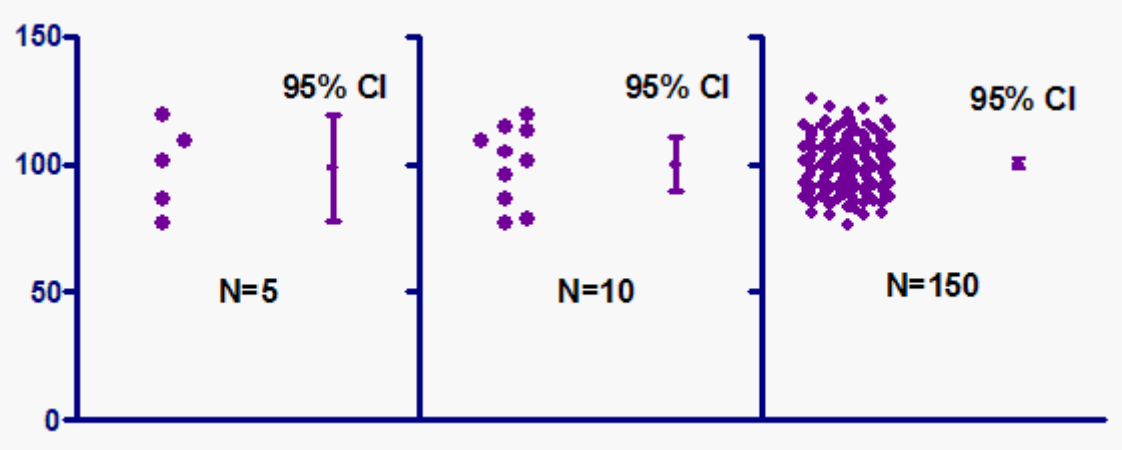


Fig. 10.4. **The bar-representation of data scattering around the mean**

Table 10.1

The differences between the S.D. and the S.E.M.

Error bars	Type	Description
Standard deviation (S.D.)	Descriptive	Typical or average difference between the data points and their mean
Standard error (S.E.M.)	Inferential	A measure of how variable the mean will be if you repeat the whole study many times
Confidence interval, usually 95 %	Inferential	A range of values contained in the 95 % CI around the mean

If we want to compare experimental results, it could be more appropriate to show inferential error bars such as S.E. or CI rather than S.D. If we want to describe our sample, for instance its normality, then the S.D. would be the one to choose.

However, if *n* is very small (for example *n* = 3), rather than showing error bars and statistics, it is better to simply plot the individual data points as it is sown in Fig. 10.5 and 10.6.

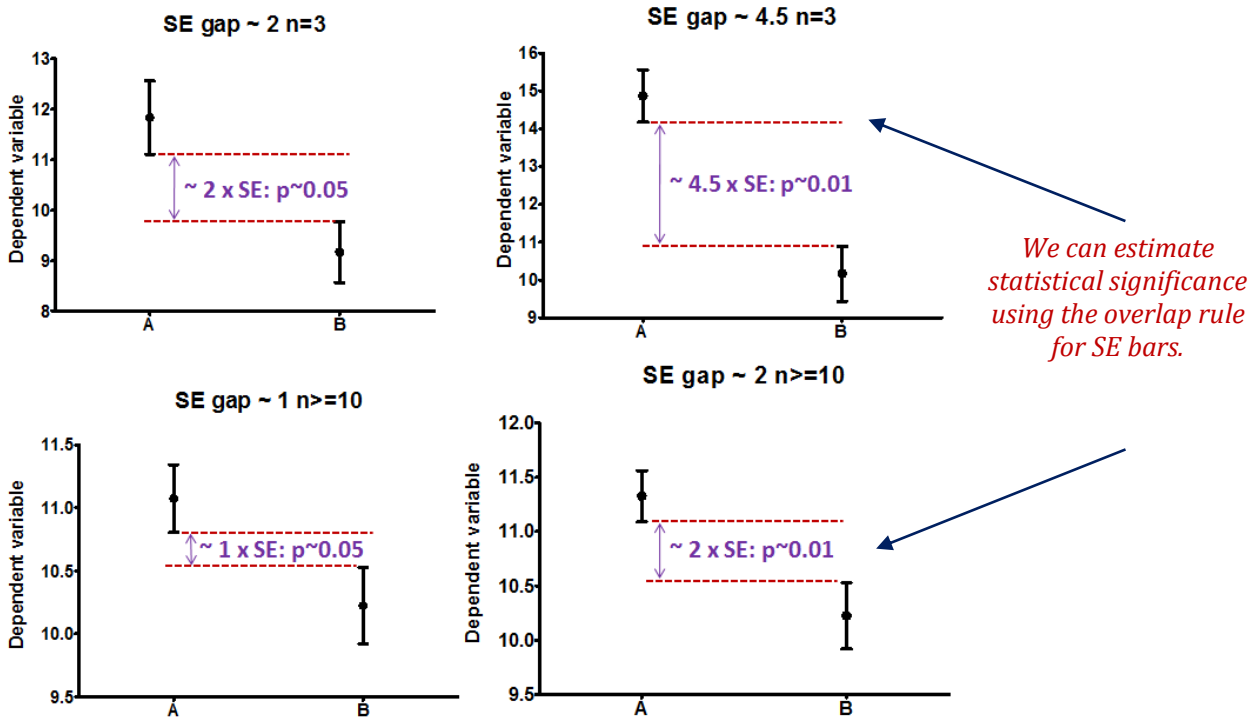


Fig. 10.5. **The SE-bars**

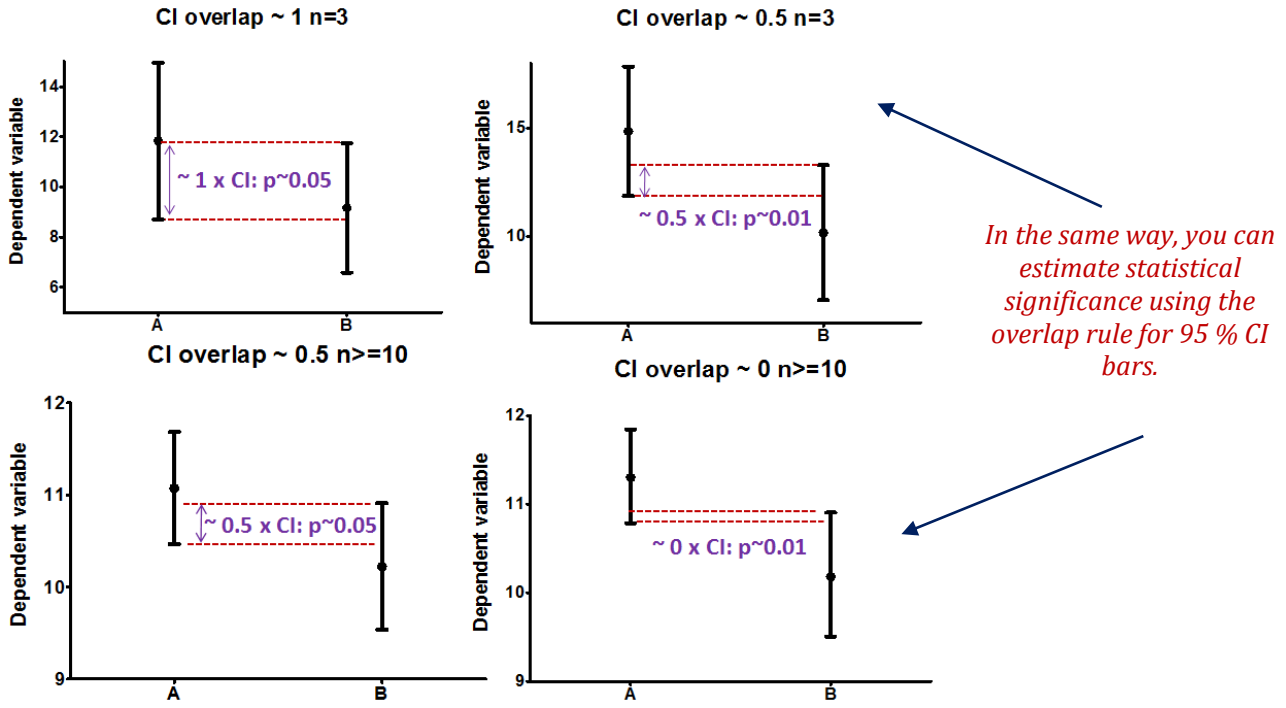


Fig. 10.6. **The CI-bars**

The assumptions of parametric data can be represented graphically (Fig. 10.7).

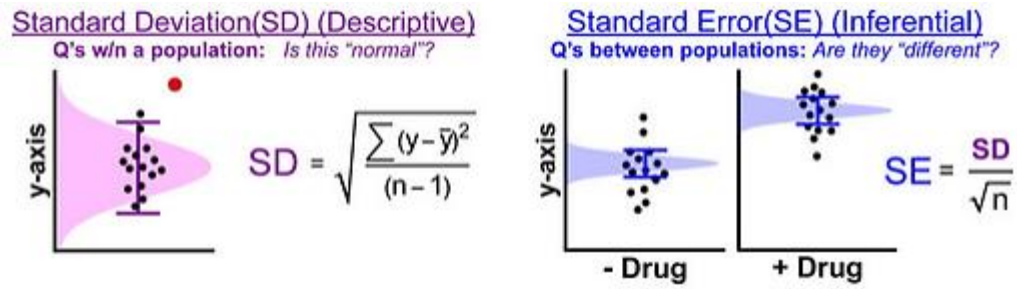


Fig. 10.7. **Graphical representation of data scattering**

When we are dealing with quantitative data, the first thing we should look at is how they are distributed, how they look like. The distribution of our data will tell us if there is something wrong in the way we collected them or enter them and it will also tell us what kind of test we can apply to make them say something.

The *t*-test, analysis of variance and correlation tests, belongs to the family of parametric tests and to be able to use them, our data must comply with the following four assumptions.

1. The data have to be normally distributed (normal shape, bell shape, Gaussian shape).

An example of normally distributed data is presented in Fig. 10.8.

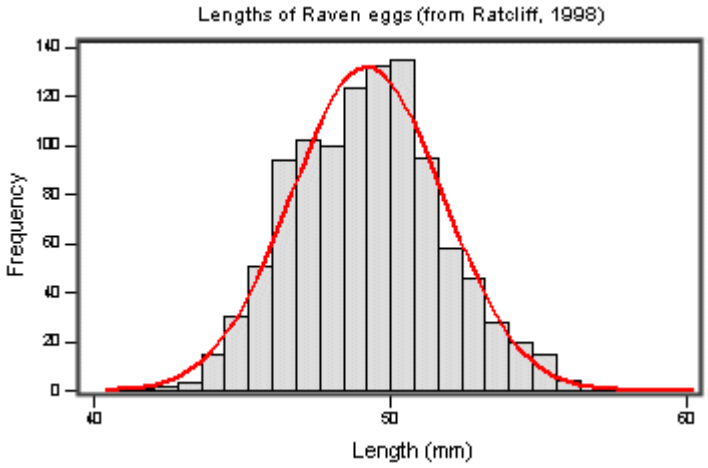


Fig. 10.8. **The theoretical *pdf* and the histogram of the normally distributed data**

The real data are not always distributed normally. There are two main types of departure from normality: skewness (the lack of symmetry of a distribution) and kurtosis (the measure of the degree of "peakedness" in the distribution). The two distributions in Fig. 10.10 have the same variance, approximately the same skew, but differ markedly in kurtosis. as shown in Fig. 10.9 and 10.10.

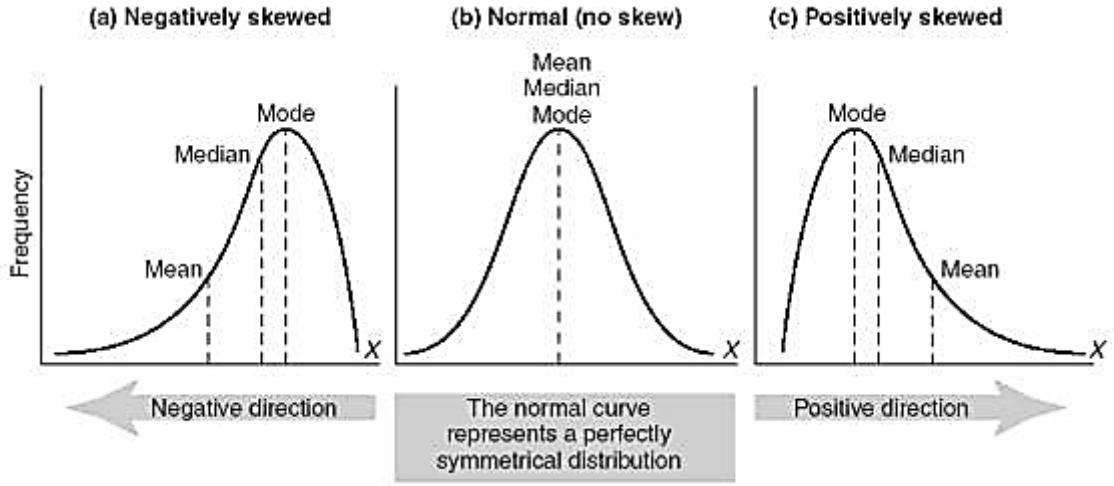


Fig. 10.9. **Asymmetry in the data distribution**

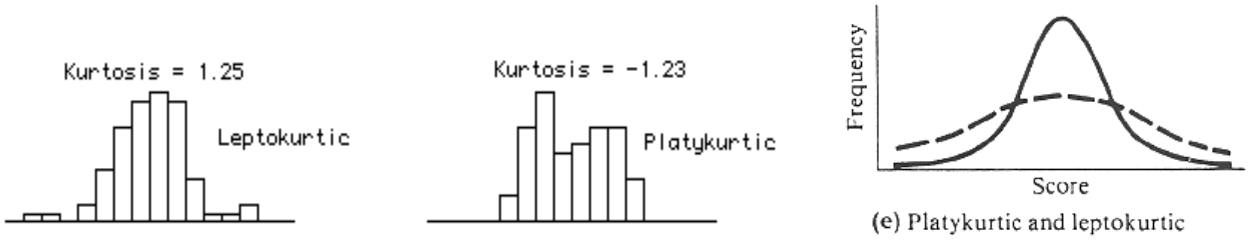


Fig. 10.10. **"Peakedness" in the data distribution**

2. Homogeneity in variance: the variance should not change systematically throughout the data.

3. Interval data: the distance between the points of the scale should be equal at all parts along the scale.

4. Independence: data from different subjects are independent so that values corresponding to one subject do not influence the values corresponding to another subject. Basically, it means one measure per subject. There are specific designs for repeated measure experiments.

How can we check that our data are parametrically normal? Let's try to do it through an example.

Example. We want to know if male coyotes are bigger than female coyotes. Of course, before doing anything else, we design our experiment and we are told that to compare two samples we need to apply a *t*-test (we will explain this test later). So basically we are going to catch coyotes and hopefully we will manage to catch males and females. Now, the tricky question here is how many coyotes do we need?

11. The power analysis with a *t*-test

Let's say, we don't have data from a pilot study but we have found some information in the literature. In a study run in similar conditions as in the one we intend to run, male coyotes (n = 20) were found to measure on average: 92cm $\pm$ 7cm (SD). We expect a 5 % difference between genders with a similar variability in the female sample.

The R-function in this case will be:

> power.t.test(n = , d = , sig.level = , power = , type = c("two.sample", "one.sample", "paired"))

$$d = \frac{M_1 - M_2}{\sqrt{\frac{s_1^2 + s_2^2}{2}}}$$

with d being the Cohen's distance, M_1 and M_2, the corresponding means and s_1 and s_2 standard deviations. In R it will be:

```
numerator <- abs(mean1-mean2)
denominator<- sqrt(((s1*s1)+(s2*s2))/2)
```

mean1<-92
mean2<-87.4 (5% less than 92cm)
s1<-7
s2<-7
d<- numerator/ denominator
d.

You should further get:

[1] 0.6571429.

So now:

> power.t.test(d = d, sig.level = 0.05, power = 0.8)

The default *Type* is two.*sample2,* so there is no need to specify it. Then we obtain:

<- n = 37.33624
<- delta = 0.6571429
<- sd = 1
<- sig.level = 0.05
<- power = 0.8
<- alternative = two.sided.

Note: *n* is the number in each group. We obtain the needed sample size of *n* = 76 (2*38).

Once the data are collected, we need to check it for normality. Though normality tests are good, the best way to get a really good idea of what is going on is to plot our data.

When it comes to normality, there are three ways to plot our data: the box plot, the scatter plot and the histogram. This has been done in R in Fig. 11.1 – 11.5. To get the data in R:

coyote<-read.csv("coyote.csv", header = TRUE)
View(coyote) or head(coyote).

Now let's start with the *stripchart*() function:

stripchart(coyote$length~coyote$gender, vertical = TRUE, method = "jitter", las = 1, ylab = "Lengths", pch = 16, col = c ("darkorange", "purple"), cex = 1.5, at = c(1.2,1.8)).

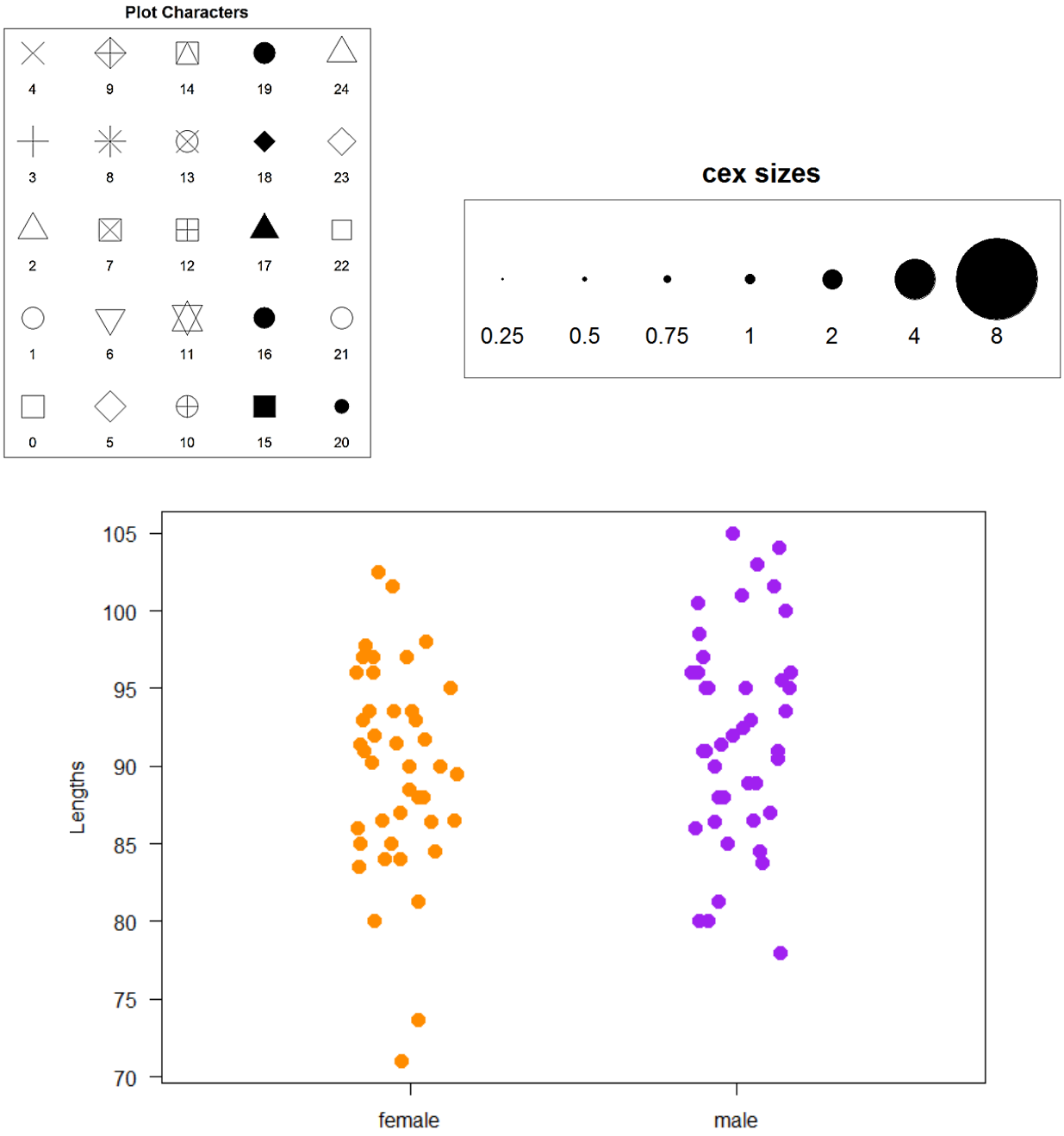


Fig. 11.1. **Scatter plots**

Now you may want to improve this graph by adding the group means (Fig. 11.2).

Another way to explore the data is the boxplot (Fig. 11.3):

```
length.means <- tapply(coyote$length,coyote$gender,mean)
loc.strip<- c(1.2,1.8)
segments(loc.strip-0.15, length.means, loc.strip+0.15, length.means, col = "black", lwd = 3)
boxplot(coyote$length~coyote$gender, col = c("orange", "purple")).
```

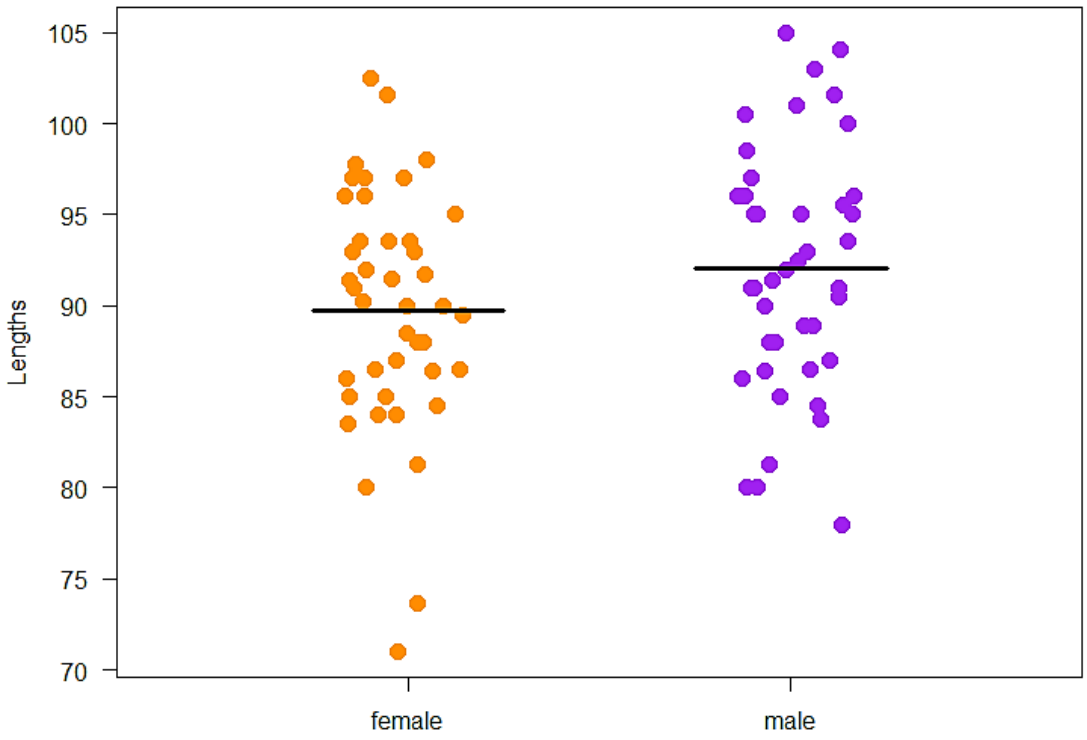


Fig. 11.2. **Adding the groups means**

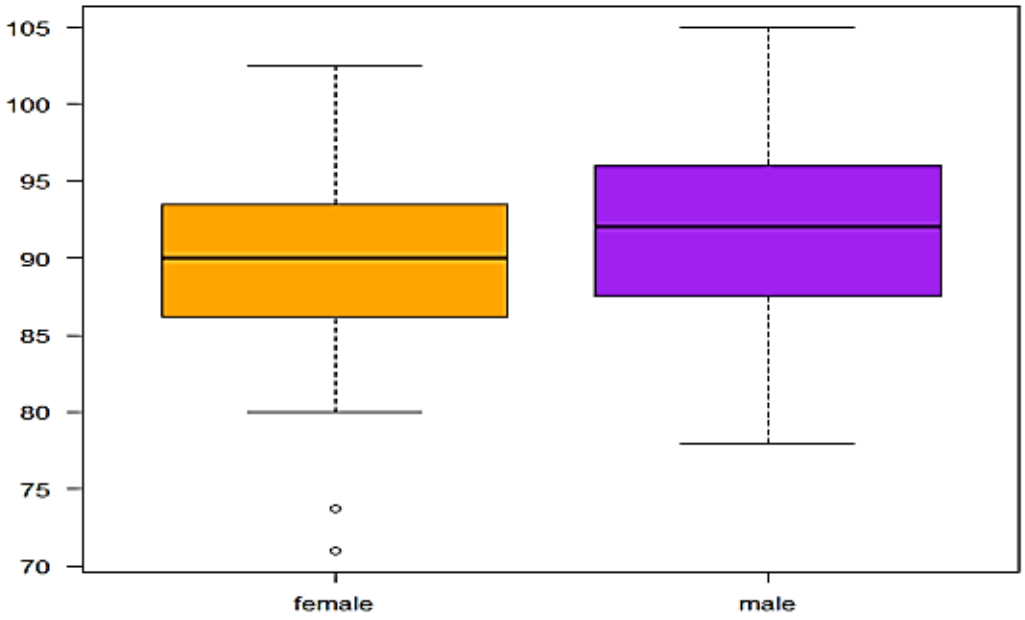


Fig. 11.3. **Box plots**

The anatomy of a boxplot is explained in the graph below. It is very important that you know how a box plot is built. It is rather simple and it will allow us to get a pretty good idea about the distribution of your data at a glance.

If the distribution is normal-ish then the box plot should be symmetrical-ish and if both (like in our case) are of the same size-ish, then we can be confident that the variances are about the same.

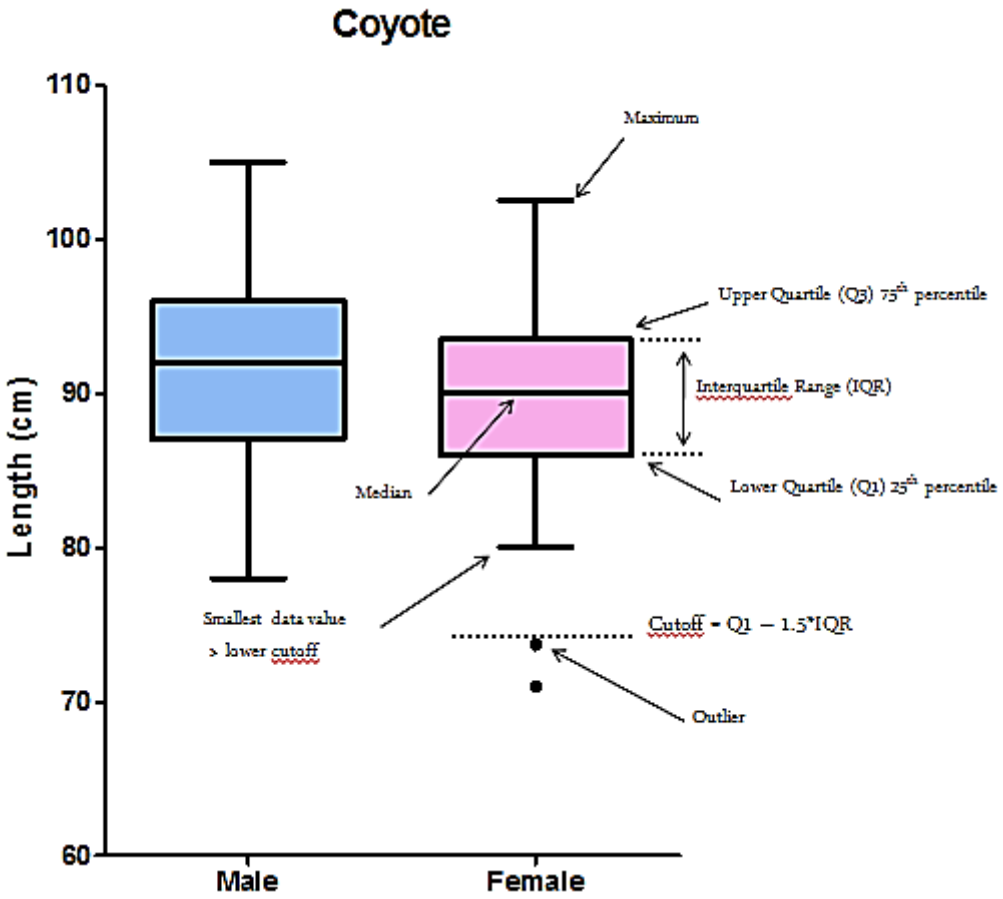


Fig. 11.4. **Confidence intervals on box plots**

Regarding the outliers, there is no really right or wrong attitude. If there is a technical issue or an experimental problem, you should remove it of course but if there is nothing obvious, it is up to you. I would always recommend keeping outliers if we can; we can run the analysis with and without it for instance and see what effect it has on the *p*-value. If the outcome is still consistent with our hypothesis, then we should keep it. If not, then it is between you and your conscience.

In Fig. 11.5, we can see the relationship between the box plot and the histogram.

Beanplots can be more informative than a boxplot in terms of "hidden" distribution especially with big datasets as we can see in Fig 11.6, but they do not identify outliers.

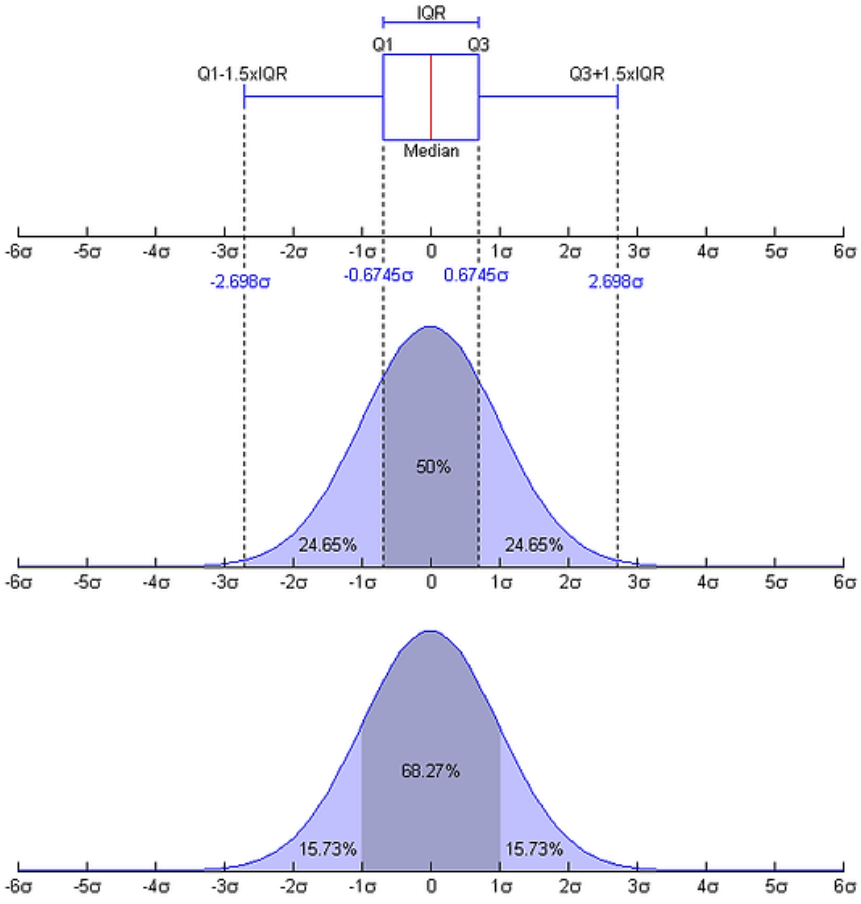


Fig. 11.5. **The relationship between the box plot and the histogram**

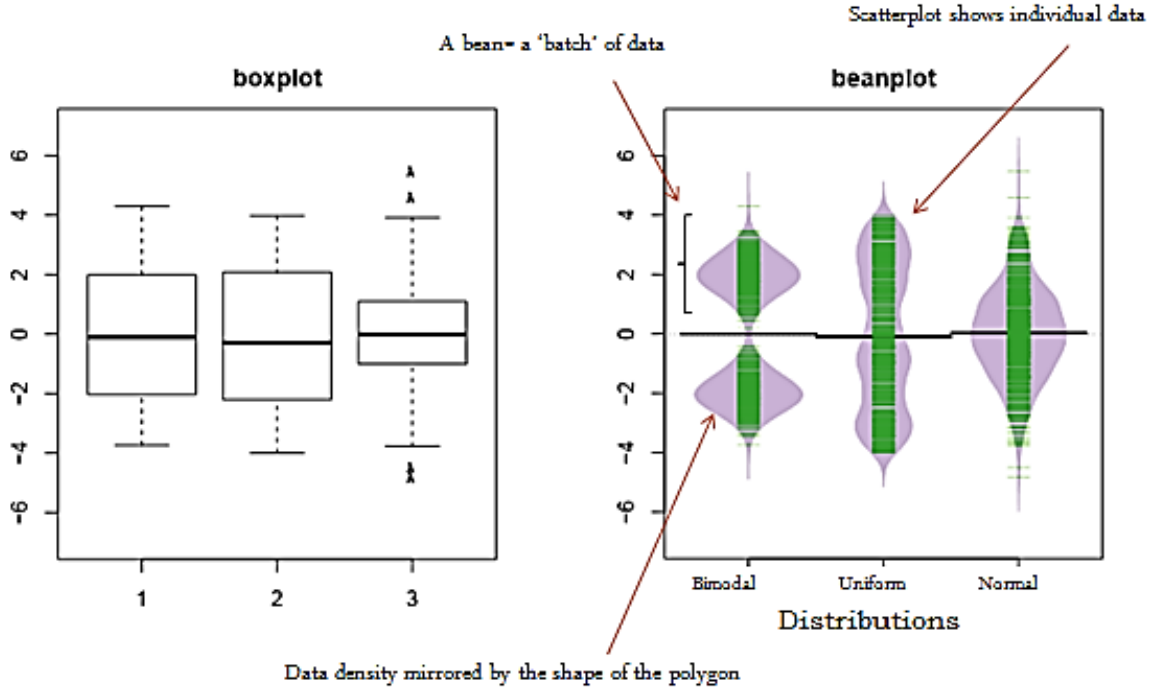


Fig. 11.6. **Beanplots**

Really, the beanplots look quite informative. More detailed graphics is given in Fig. 11.7.

```
> beanplot(coyote$length~coyote$gender, las=1,
ylab="Length (cm)")   ## beanplot package ##
```

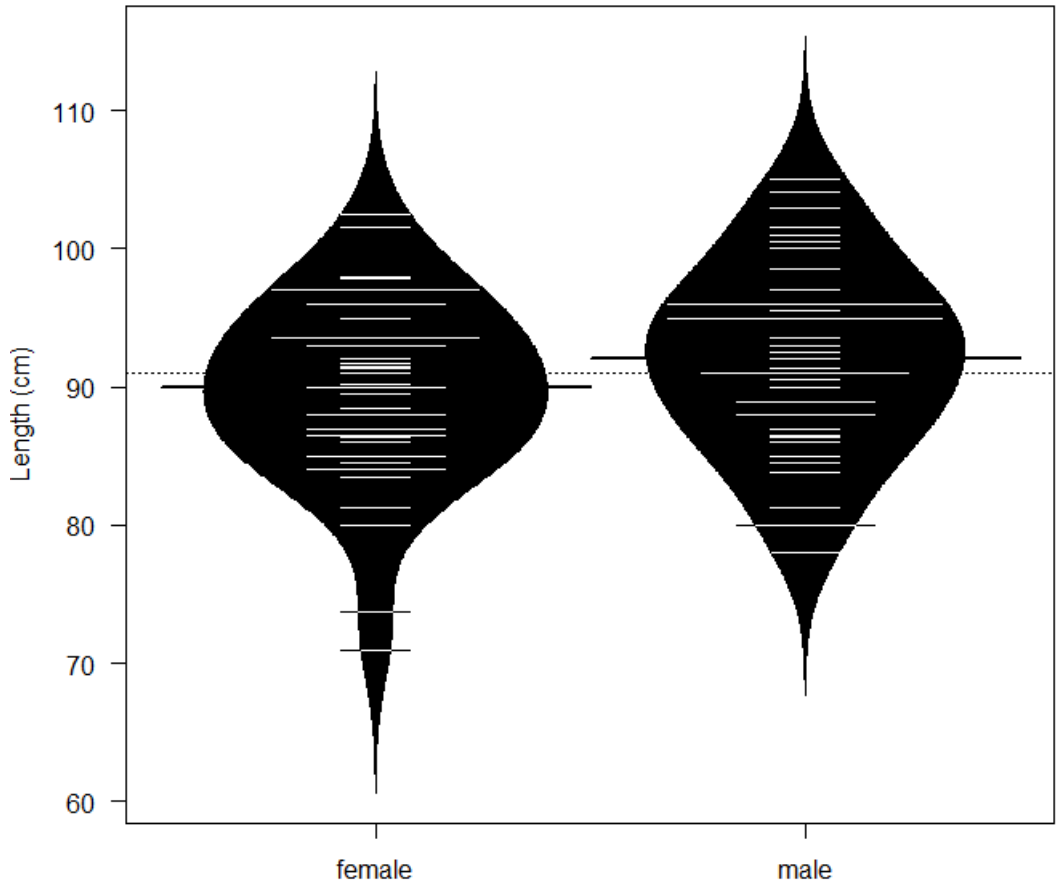


Fig. 11.7. **Beanplots in details**

11.1. Histograms in R

The histograms in R can be built using the *hist*() function (Fig. 11.8):

```
> par(mfrow=c(1,2))
>hist(coyote[coyote$gender == "male",]$length, main = "Male",
> xlab = "Length", col = "lightblue")
>hist(coyote[coyote$gender == "female",]$length, main = "Female",
> xlab = "Length", col = "pink")
```

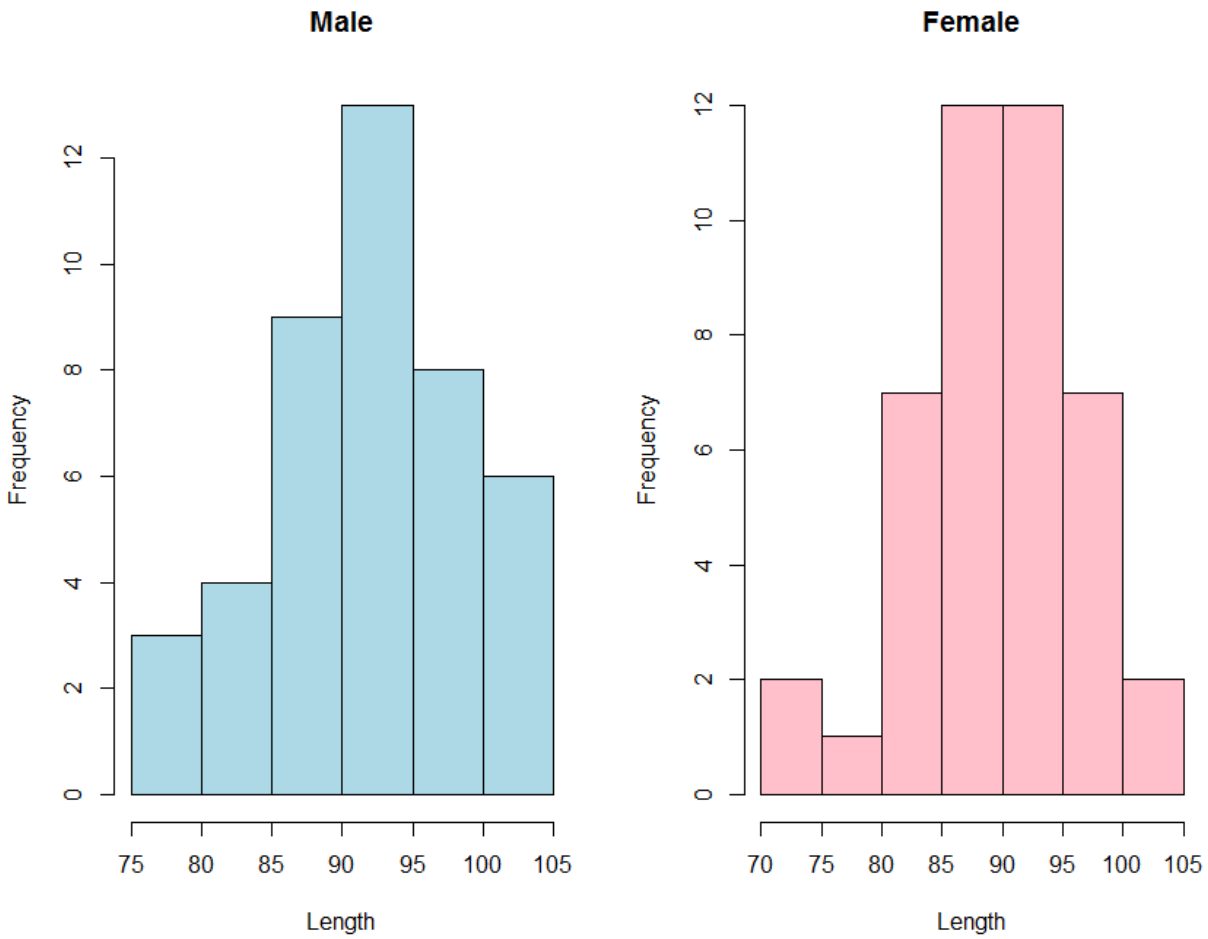


Fig. 11.8. **Histograms in R**

So from the graphs we have plotted, we can say that the first and second assumptions are likely to be met: the data seem normal enough (symmetry of the graphs) and the variability seems comparable between the groups (spread of the data). Frequently preference goes to the box plot as it tells you in one go anything you need to know: where you are with the first two assumptions and it shows you the outliers.

Still we may come across cases where it is not that obvious so you can ask R to test for normality (Shapiro – Wilk test or D'Agostino and Pearson tests) and homogeneity of variance (Levene test). Here we are going to use two new functions: *stat.desc*(), which gives a statistical summary of the data, and the test for normality (Shapiro – Wilk test). Also we can use the *tapply*() function, which allow us to do it for males and females separately in one go:

```
> tapply(coyote$length,coyote$gender, stat.desc, basic = F, desc = F, norm = T)
    ## pastecs package ##.
```

There is no significant departure from normality for females (p = 0.316) or males (p = 0.819).

That was the first assumption. Now we can check the second assumption (homogeneity of variances) using the Levene test. The second assumption:

```
> leveneTest(coyote$length, coyote$gender, center = mean) ## car package ##.
```

So good again but not surprising news: the variances of the two genders do not differ significantly (p = 0.698).

Don't be too quick to switch to nonparametric tests. While they do not assume Gaussian distributions, these tests do assume that the shape of the data distribution is the same in each group. So if your groups have very different standard deviations and so are not appropriate for a parametric test, they should not be analyzed for its non-parametric equivalent either. However parametric tests like ANOVA and *t*-tests are rather robust, especially when the samples are not too small so you can get away with small departure from normality and small differences in variances. Often the best approach is to transform the data using logarithms or reciprocals with restoring equal variance.

Finally we may want to represent the data as a classical bar chart. To achieve that, we can type the lines below:

```
bar.length<-barplot(length.means, col = c("pink", "lightblue"), ylim = c(50,100), beside = TRUE, xlim = c(0,1), width = 0.3, ylab = "Mean length", las = 1, xpd = FALSE, las = 1)

        ## plotrix package ##
        length.se<-tapply(coyote$length,coyote$gender,std.error)
```

Now, to plot the error bars, we are going to use *arrow()*. We need to specify the coordinates (*x, y*). *barplot* () returns the values of the center of the bars as we can see in Fig. 11.9:

```
>arrows(x0 = bar.length, y0 = length.means-length.se, x1 = bar.length, y1 = length.means+length.se, length = 0.3, angle = 90, code = 3)
```

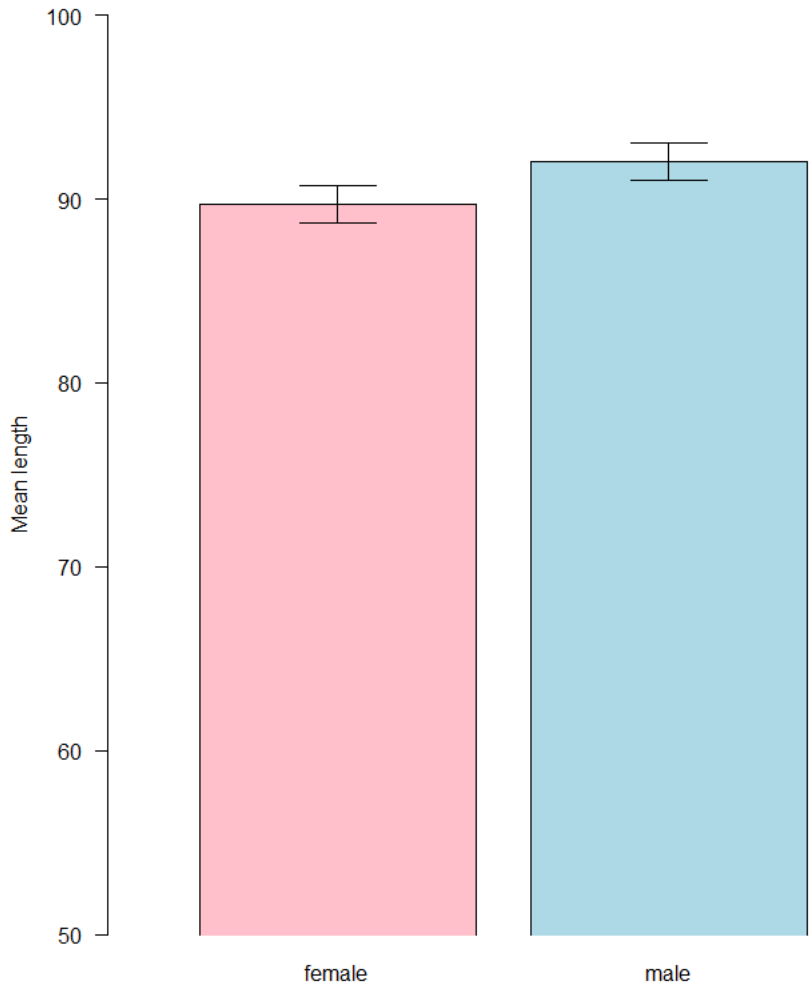


Fig. 11.9. **Error bars**

12. The comparison of more than two means: analysis of variance

When we want to compare more than two means (e.g. more than two groups), we cannot run several *t*-tests because it increases the "familywise error rate", which is the error rate across tests conducted on the same experimental data.

Example: if we want to compare three groups (1, 2 and 3) and we carry out 3 *t*-tests (groups 1–2, 1–3 and 2–3), each with an arbitrary 5 % level of significance, the probability of not making the Type I error is 95 % (= 1 - 0.05). The three tests being independent, we can multiply the probabilities, so the overall probability of no Type I errors is: 0.95 * 0.95 * 0.95 = 0.857, which means that the probability of making at least one Type I error (to say that there is a difference whereas there is not) is 1 - 0.857 = 0.143 or 14.3 %. So

the probability has increased from 5 % to 14.3 %. If we compare 5 groups instead of 3, the familywise error rate is 22.6 % (= 1 - $(0.95)^5$).

To overcome the problem of multiple comparisons, we need to run an analysis of variance (ANOVA), which is an extension of the two groups' comparison of a *t*-test but with a slightly different logic. If we want to compare 5 means, for example, we can compare each mean with another, which gives you 10 possible 2-group comparisons, which is quite complicated. So, the logic of the *t*-test cannot be directly transferred to the analysis of variance. Instead the ANOVA compares variances: if the variance amongst the 5 means is greater than the random error variance (due to individual variability for instance), then the means must be more spread out than we would have explained by chance.

The statistic for ANOVA is the F ratio:

$$F = \frac{\text{variance among sample means}}{\text{variance within samples (random)}},$$

also:

$$F = \frac{\text{variation explained by the model (systematic)}}{\text{variation explained by unsystematic factors}}.$$

If the variance amongst sample means is greater than the error variance, then $F > 1$. In an ANOVA, we test whether F is significantly higher than 1 or not.

Imagine we have a dataset of 78 data points, we advance a hypothesis that these points in fact belong to 5 different groups (this is our hypothetical model). So we arrange the data into 5 groups and run an ANOVA (Table 12.1).

Table 12.1

ANOVA

Source of variation	Sum of Squares	df	Mean Square	F	*p*-value
Between Groups	2.665	4	0.6663	8.423	<0.0001
Within Groups	5.775	73	0.0791		
Total	8.44	77			

Let's go through the figures in Table 12.1. First, the bottom row of the table: total = $_i(x_i - \mathrm{x})^2$.

In our case, total SS = 8.44. If we were to plot our data to represent the total SS, we would produce the graph shown in Fig. 12.1. So the total SS is the squared sum of all the differences between each data points and the grand mean. This is a quantification of the overall variability in our data. The next step is to partition this variability: how big is variability between the groups (explained by the model) and how big is the variability within the groups (random/individual/remaining variability)?

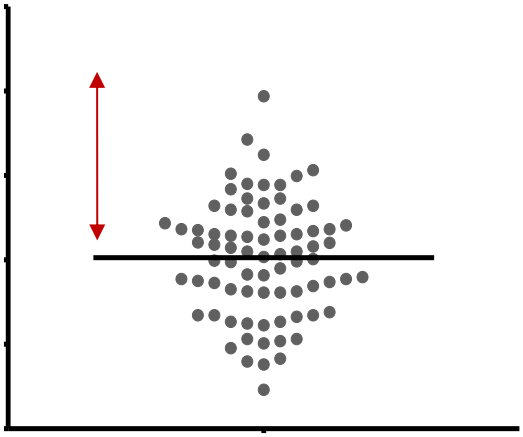


Fig. 12.1. **A scatter plot**

According to our hypothesis our data can be split into 5 groups because, for instance, the data come from 5 cell types, like in the graph in Fig. 12.2.

So we work out the mean for each cell type and we work out the squared differences between each of the means and the grand mean ($\sum n_i$ (Mean_i – Grand mean)^2). In our example (the second row of the table): between groups SS = 2.665 and, since we have 5 groups, there are 5 – 1 = 4 *df* and the mean SS = 2.665/4 = 0.6663.

If you remember the formula of the variance (= SS / *N* - 1, with df = *N* - 1), you can see that this value quantifies the variability between the groups' means: it is the between-groups variance.

There is one row left in Table 12.1, the within-groups variability. It is the variability within each of the five groups, so it corresponds to the difference between each data point and its respective group mean: within the groups the sum of squares = $\sum (x_i - \mathrm{mean}_i)^2$ which in our case is equal to 5.775.

This value can also be obtained by doing 8.44 – 2.665 = 5.775, which is logical since it is the amount of variability left from the total variability after the

variability explained by the model has been removed: in our example, the 5 groups' sizes are 12, 12, 17, 17 and 17 so *df* = 5 ∗ (*n* – 1) = 73 (Fig. 12.2).

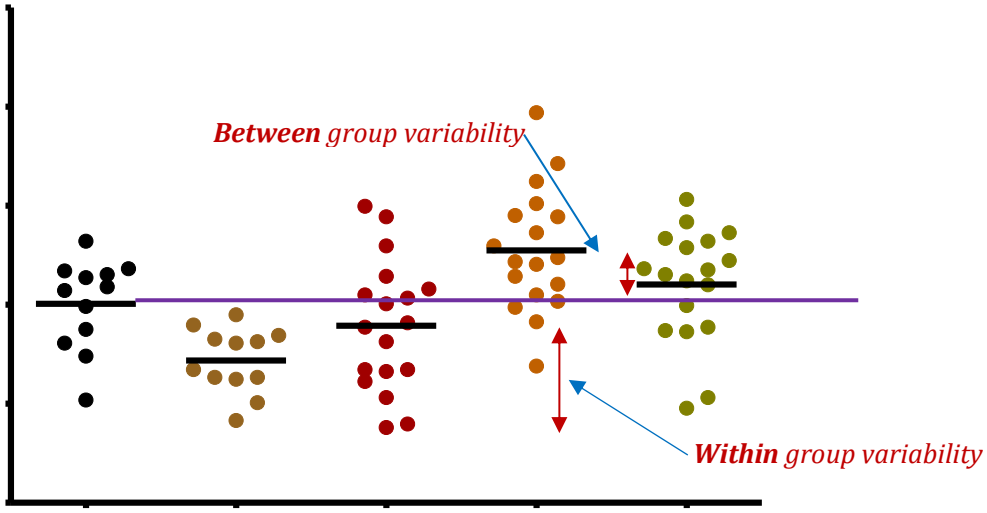


Fig. 12.2. **Five scatter groups**

So, the mean variability within the groups: SS = 5.775/73 = 0.0791. This quantifies the remaining variability, the one not explained by the model, the individual variability between each value and the mean of the group to which it belongs according to the hypothesis. From this value one can obtain what is often referred to as the pooled SD (= SQRT(MS (Residual or Within Group)). When obtained in a pilot study, this value is used in the power analysis.

At this point, we can see that the amount of variability explained by our model (0.6663) is far higher than the remaining one (0.0791).

We can work out the *F*-ratio: F = 0.6663 / 0.0791 = 8.423.

The level of significance of the test is calculated by taking into account the *F*-ratio and the number of *df* (degree of freedom) for the numerator and the denominator.

In our example, $p < 0.0001$, so the test is highly significant and we are more than 99 % confident when we say that there is a difference between the groups' means. This is an overall difference and even if we have an indication from the graph, we cannot say which mean is significantly different from which.

This is because the ANOVA is an "omnibus" test: it tells us that there is (or not) an overall difference between our means but not exactly which means are significantly different from which. This is why we apply a post-hoc test. Post-hoc tests could be compared to *t*-tests but with a more stringent

approach, a lower significance threshold to correct for familywise error rate. We will go through post-hoc tests in more details later.

Example. We want to find out if there is a significant difference in terms of protein expression between 5 cell types.

First we import the dataset:

```
> protein<-read.csv("protein.expression.csv", header = T)
```

Then for ease of graphical representation we restructure it:

```
protein.stack<-melt(protein) ## reshape2 package ##
colnames(protein.stack)<-c("line","expression").
```

Than we get rid of the missing values:

```
protein.stack.clean <- protein.stack[!is.na(protein.stack$expression),]
```

Now we can plot the data, either as a scatterplot (Fig. 12.3)

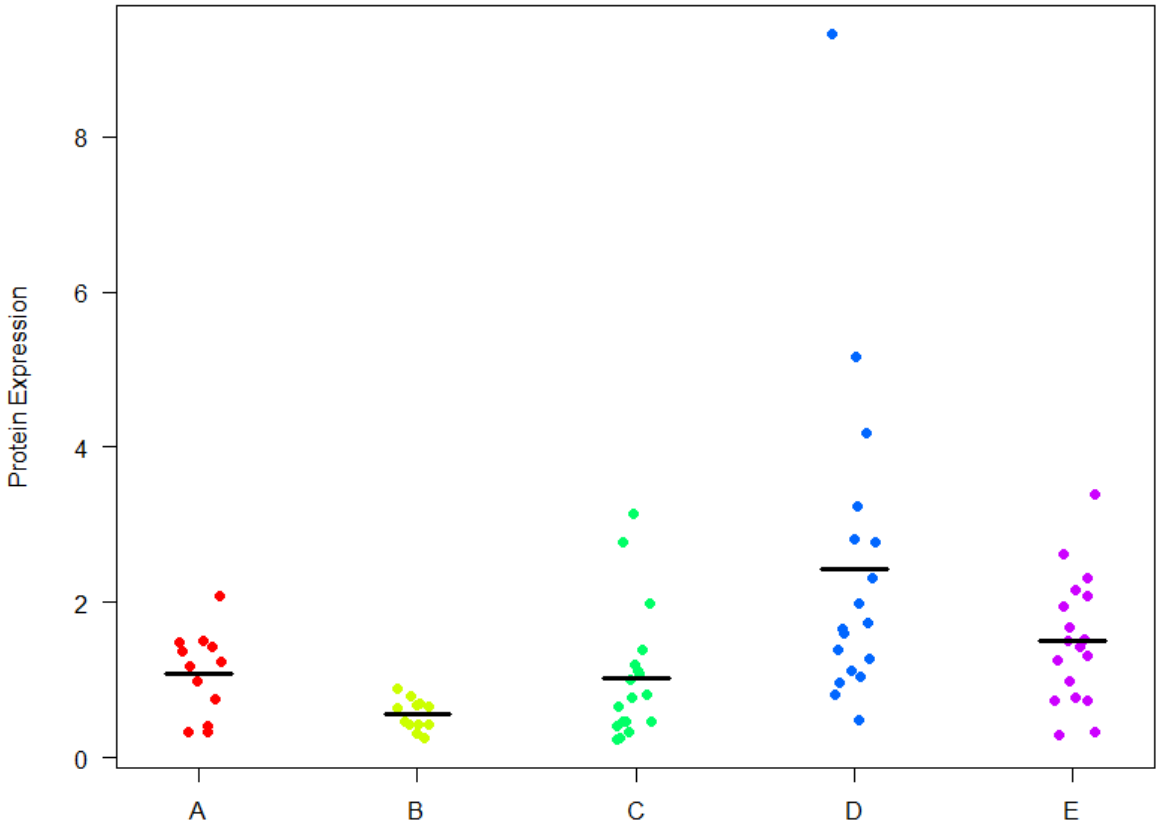


Fig. 12.3. **Scatterplots for groups**

```
>stripchart(protein.stack.clean$expression~protein.stack.clean$line,vertical=
TRUE,method="jitter",las=1,ylab="Protein Expression", pch=16, col=rainbow(5))
```

```
> expression.means<-
+ tapply(protein.stack.clean$expression,protein.stack.clean$line,mean)

> loc.strip<-1:5
> segments(loc.strip-0.15,expression.means,loc.strip+0.15,
expression.means,
+ col="black", lwd=3)
```

or a boxplot (Fig. 12.4).

```
>boxplot(protein.stack.clean$expression~protein.stack.clean$line,col=rainbow(5)).
```

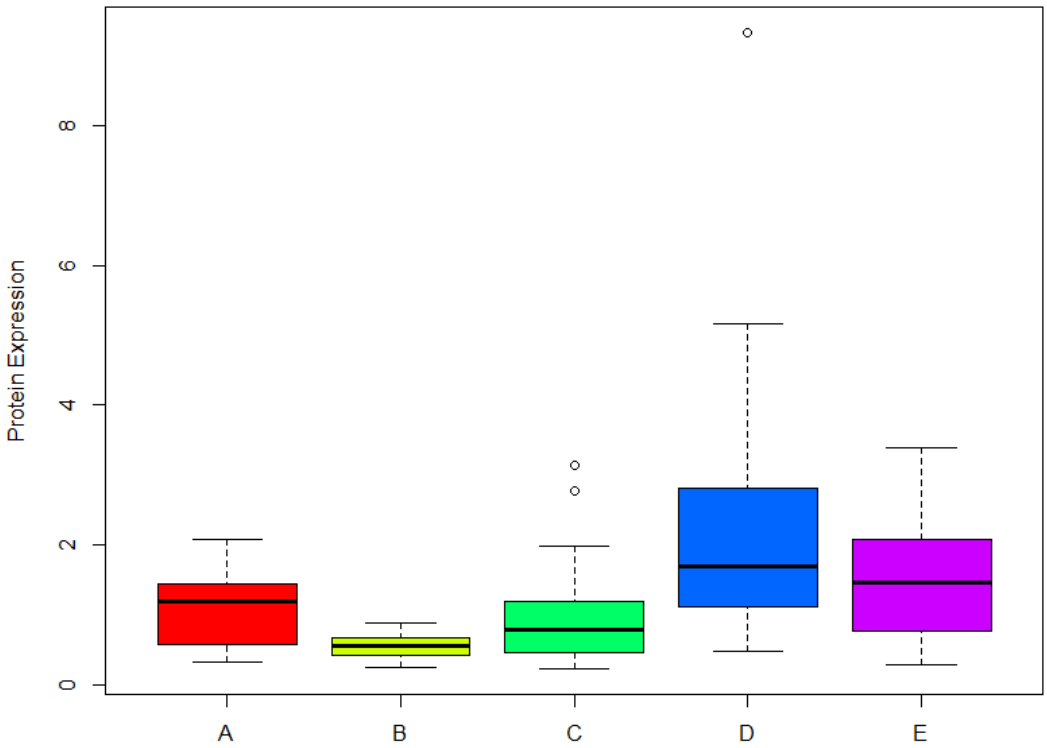


Fig. 12.4. **Boxplots for groups**

First we need to see whether the data meet the assumptions for a parametric approach. Well, it does not look good: 2 out of 5 groups (C and D) show a significant departure from normality (we cannot use the D'Agostino test as R requires $n \geq 20$). As for the homogeneity of variance, even before testing it, a look at the scatter plots and boxplots tells us that there is no way the second assumption is met. The data from groups C and D are quite skewed and a look at the raw data shows more than a 10-fold jump between values of the same group (e.g. in group A, value line 4 is 0.17 and value line 10 is 2.09). So,

```
>tapply(protein.stack.clean$expression,protein.stack.clean$line,stat.desc,des
c = F, basic = F, norm = T)) ## pastecs package ##.
```

A good idea would be to log-transform the data so that the spread is more balanced and to check again on the assumptions. The variability seems to be scale related: the higher the mean, the bigger the variability. This is a typical case for log-transformation.

Speaking of log-transformation, the function *beanplot*() has a built-in procedure to automatically determine whether a log transformation of the response axis is appropriate or not, to get rid of it, we need: log = "". In our case, since we are thinking log we might as well let the function choose (Fig. 12.5):

> beanplot(protein.stack.clean$expression~protein.stack.clean$line, ylab = "Protein + Expression") ## beanplot package ##.

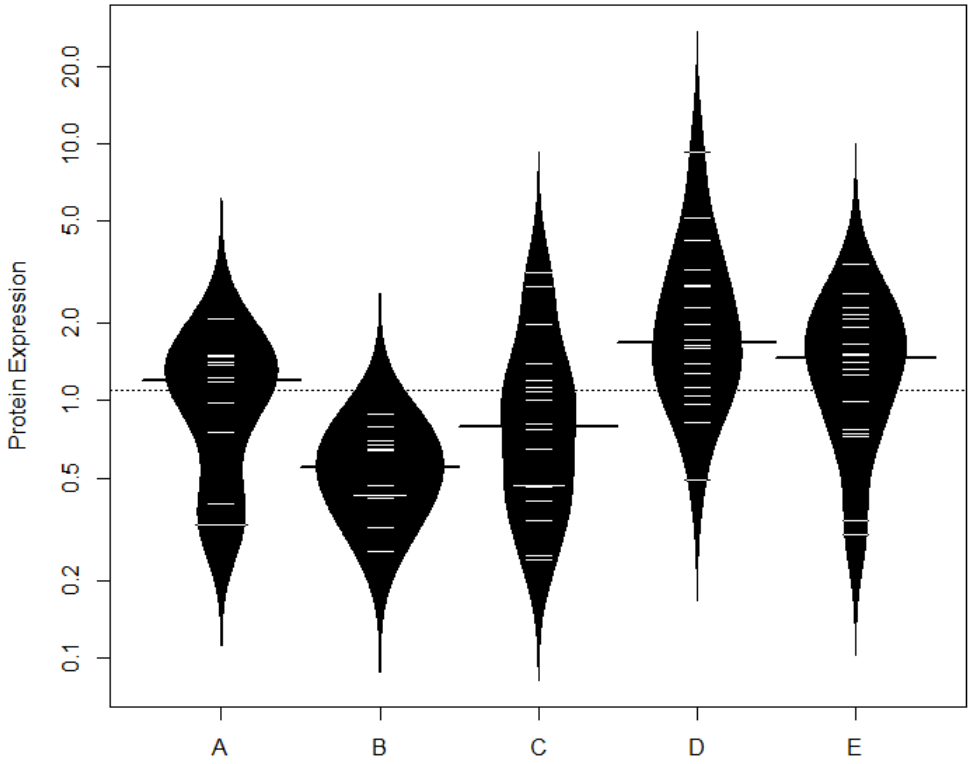


Figure 12.5. **Beanplots for groups**

We can actually check beforehand that a log transformation will stabilize your data by changing your linear *y*-axis to a log *y*-axis. To do so, with the stripchart for instance (Fig. 12.6), we go:

>stripchart(protein.stack.clean$expression~protein.stack.clean$line,vertical = = TRUE,
+ method="jitter",las=1,ylab="Protein Expression", pch=16, col=rainbow(5),
+ **log = "y")**
expression.means<-

> tapply(protein.stack.clean$expression,protein.stack.clean$line,mean)
> loc.strip<-1:5
>segments(loc.strip-0.15,expression.means,loc.strip+0.15,expression.means,
+ col = "black", lwd = 3)

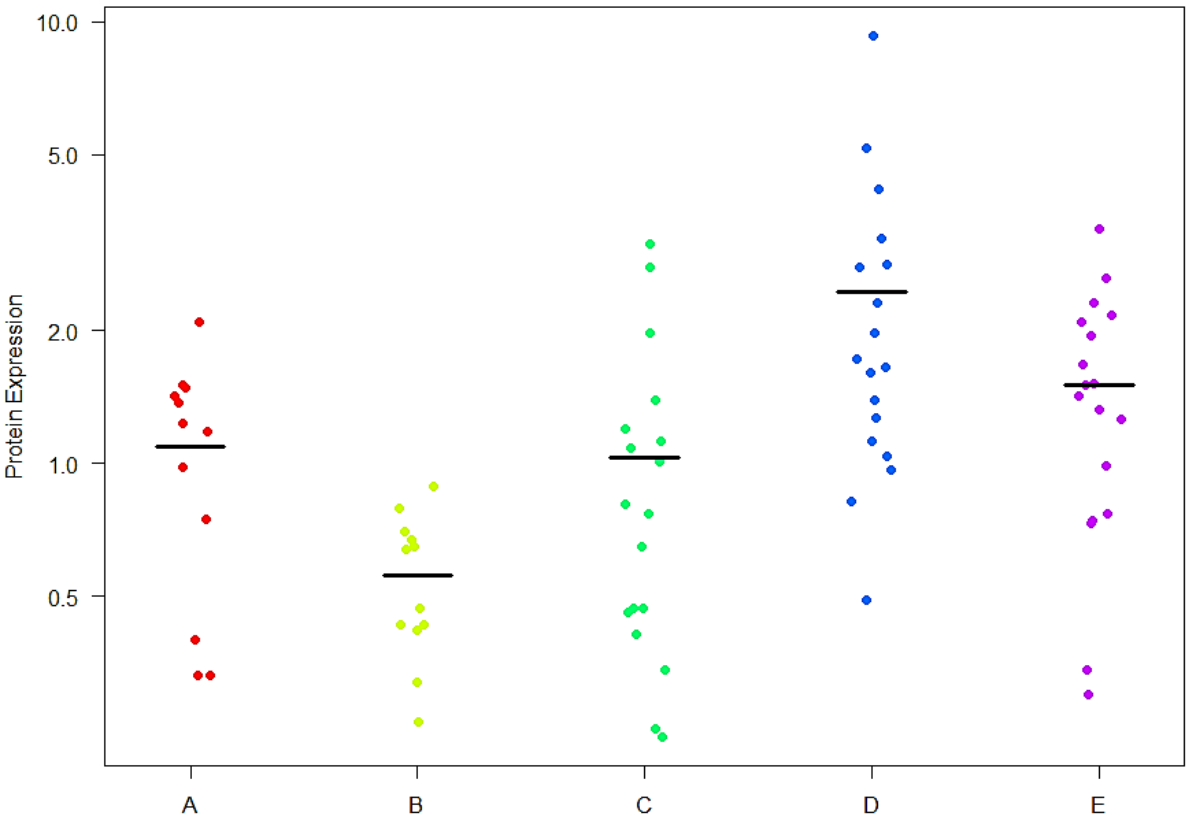


Fig. 12.6. **Stripcharts for groups**

It looks much better, so let's go for the actual log-transformation:

protein.stack.clean$log10.expression<-log10(protein.stack.clean$expression)

>tapply(protein.stack.clean$log10.expression,protein.stack.clean$line,stat.de
+sc,basic = F, norm = T, desc = F)

OK, the situation is getting better: the first assumption is met-ish and from what we see when we plot the transformed data (boxplots) the homogeneity of variance has improved a great deal (Fig. 12.7):

>boxplot(protein.stack.clean$log10.expression~protein.stack.clean$line,col = rainbow(5)).

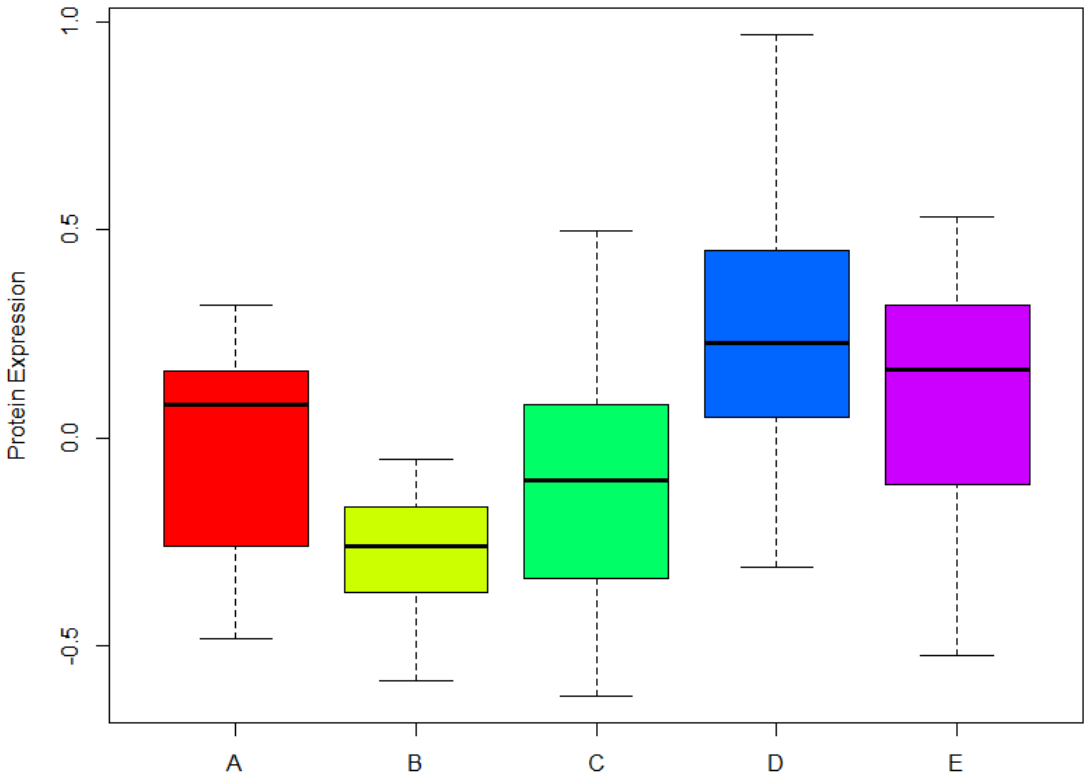


Fig. 12.7. **Boxplots for groups after data transformation**

One last thing before we run the ANOVA: we need to check for the second assumption: the homogeneity of variance. To do so, we do what we did before running the *t*-test: we run a Levene test:

```
> leveneTest(protein.stack.clean$log10.expression,protein.stack.clean$line,
+ center=mean) ## car package ##
```

Now that we have sorted out the data, we can run the ANOVA: to do so, you go:

```
> anova.log.protein<-aov(log10.expression~line,data = protein.stack.clean)
+ summary(anova.log.protein).
```

The overall *p*-value is significant (*p* = 1.78e-05) so the next thing to do is to choose a post-hoc test. There are 2 widely used: the Bonferroni test which is quite conservative so we should only choose it when we are comparing no more than 5 groups and the Tukey-test, which is more liberal. First let's try the Bonferroni test. It is built into R:

```
> pairwise.t.test(protein.stack.clean$log10.expression,
+ protein.stack.clean$line, p.adj = "bonf").
```

Then Tukey:

TukeyHSD(anova.log.protein,"line")

Again, from Table 12.1 we can find out which pairwise comparison reaches significance and which does not (Fig. 12.8):

```
>bar.expression<-barplot(expression.means, beside = TRUE, ylab = "Mean
+ expression", ylim = c(0,3), las = 1)
> expression.se <- tapply(protein.stack.clean$expression,
+ protein.stack.clean$line,std.error)
>arrows(x0 = bar.expression, y0 = expression.means-expression.se, x1 =
bar.expression,
> y1 = expression.means+expression.se, length = 0.2, angle = 90, code = 3)
```

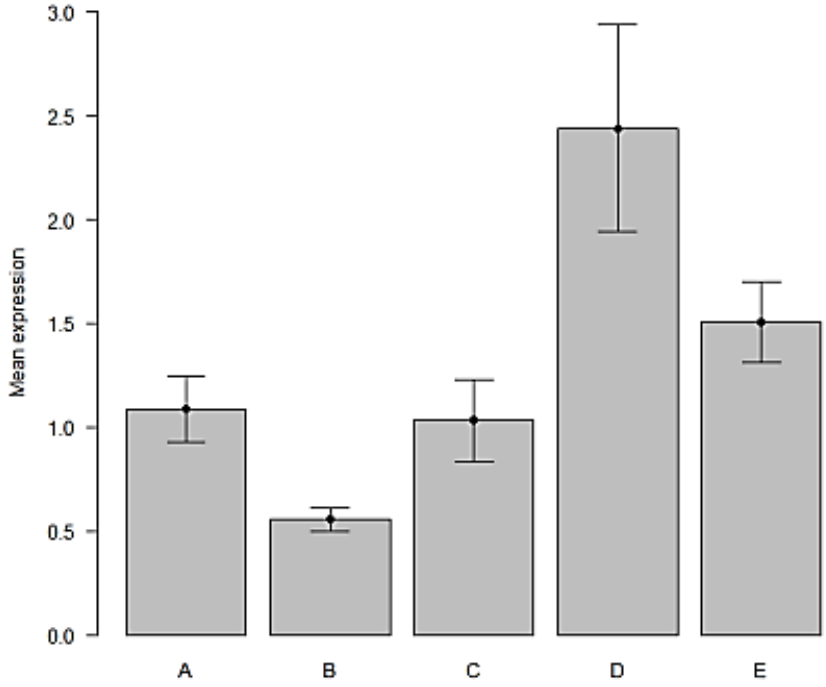


Fig. 12.8. **Pairwise comparison**

If we want to find out about the relationship between two continuous variables, we can run a correlation.

12.1. The correlation coefficient

A correlation is a measure of a linear relationship (which can be expressed as straight-line graphs) between variables. The simplest way to find

out whether two variables are associated is to look at whether they covary. To do so, we combine the variance of one variable with the variance of the other:

$$\text{cov } X,Y = \sum_{i=1}^{N} \frac{x_i - x \quad y_i - y}{N}.$$

A positive covariance indicates that as one variable deviates from the mean, the other one deviates in the same direction, in other words if one variable goes up, the other one goes up as well.

The problem with the covariance is that its value depends upon the scale of measurement used, so we would not be able to compare covariance between datasets unless both data are measures in the same units. To standardize the covariance, it is divided by the SD of the 2 variables. It gives us the most widely-used correlation coefficient: the Pearson product-moment correlation coefficient "r":

$$r = \sum_{i=1}^{N} \frac{x_i - x \quad y_i - y}{N S_x S_y}.$$

Of course, you don't need to remember that formula but it is important that you understand what the correlation coefficient does: it measures the magnitude and the direction of the relationship between two variables. It is designed to range in value between 0.0 and 1.0 (Fig. 12.9).

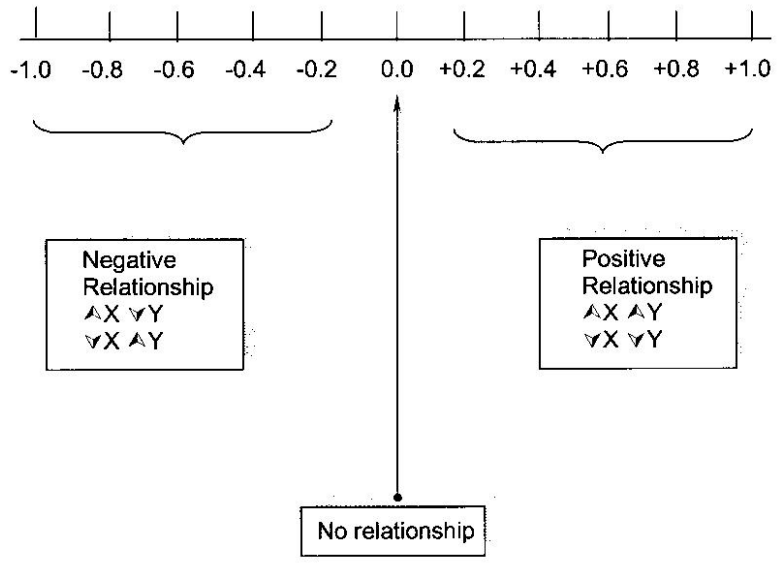


Fig. 12.9. **Positive and negative correlations**

The two variables do not have to be measured in the same units but they have to be proportional (meaning linearly related). Apart from *r*, there is

another important coefficient: the coefficient of determination R^2: it gives the proportion of variance in *Y* that can be explained by *X*, in percentage.

Finally, the assumptions for correlation (regression in general) are pretty much the ones we have seen before:

Linearity: The relationship between *X* and the mean of *Y* is linear.

Homoscedasticity: The variance of the residual is the same for any value of *X*.

Independence: Observations are independent of each other.

Normality: For any fixed value of *X*, *Y* is normally distributed.

When running a regression in general and a correlation in particular, we need to check for problematic points. They can be:

Outliers: an outlier is defined as an observation that has a large residual. In other words, the observed value for the point is very different from that predicted by the regression model.

Leverage points: A leverage point is defined as an observation that has a value of *X* that is far away from the mean of *X*.

Influential observations: An influential observation is defined as an observation that changes the slope of the line. Thus, influential points have a large influence on the fit of the model. One method to find influential points is to compare the fit of the model with and without each observation.

The bottom line is that first we look at the outliers, once we have identified them, we check the influence statistics and if one or more are "out of line", we can then safely remove the value.

Example. Graphical data mining. Input data in R:

```
> exam.anxiety<-read.table("Exam Anxiety.dat", sep = "\t",header = T).
```

The first thing we are going to do is to plot the data (Fig. 12.10). We will start with revising time vs anxiety levels.

```
>  plot(exam.anxiety$Anxiety~exam.anxiety$Revise,col=exam.anxiety$Gender,
+ pch = 16)
> legend("topright", title = "Gender", inset = .05, c ("Female","Male"), horiz =
TRUE, + pch = 16, col = 1:2).
```

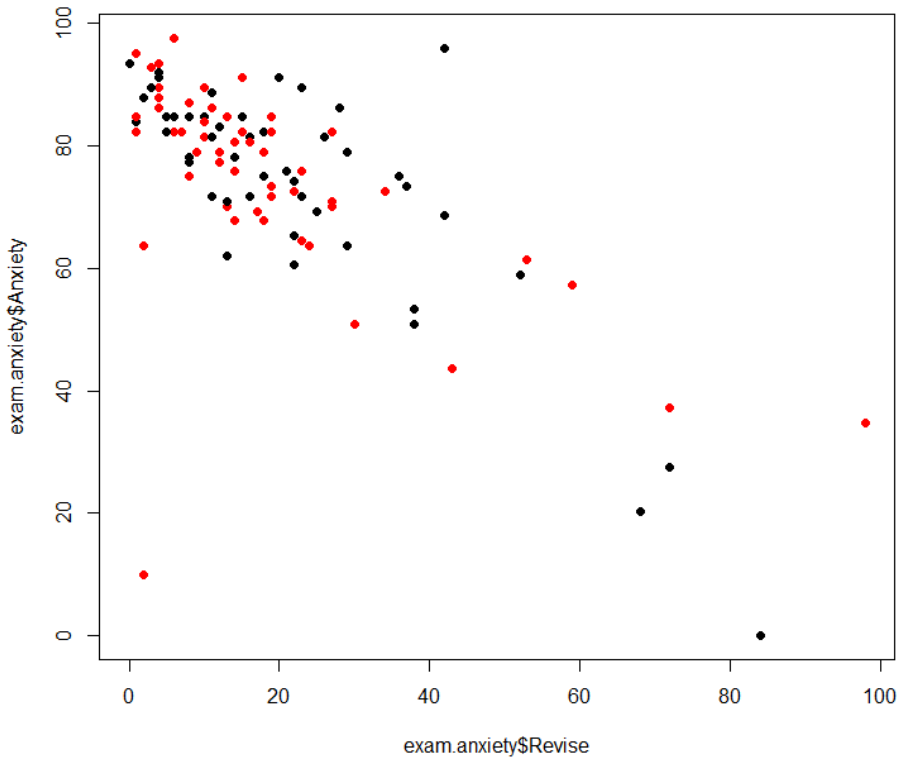


Fig. 12.10. **Plotting the data**

By looking at the graph, one can think that something is happening here. To get a better idea we can add lines-of-best fit (Fig. 12.11) but to do that we first need to fit the model as the lines-of-best fit's coefficients are one of the outputs of the regression:

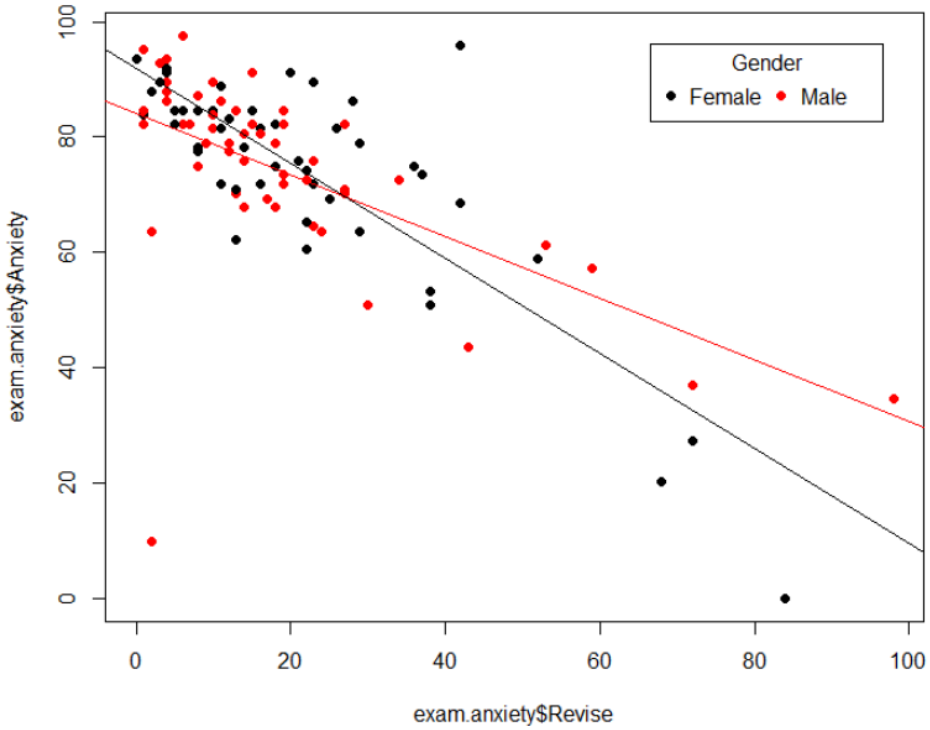


Fig. 12.11. **Data with the best fit lines**

```
fit.male<-lm(Anxiety~Revise, data = exam.anxiety[exam.anxiety$Gender == "Male",])
fit.female<-lm(Anxiety~Revise, data = exam.anxiety[exam.anxiety$Gender == "Female",])

> abline((fit.male), col = "red")
> abline((fit.female), col = "black")
```

Now, we want to quantify the strength of the relationship between our two variables of interest but first we need to check on the data.

12.2. Outliers and influential cases

We might have noticed that one point, possibly two, is really far from the others. So let's check out our data and keep an eye on our misbehaving cases and in particular the boy (point Code 78) who spent two hours revising, did not feel stressed about it (Anxiety score: 10) and managed a 100 % mark in his exam. Then, in Fig. 12.12 we see:

```
> par(mfrow = c(2,2) plot(fit.male)
```

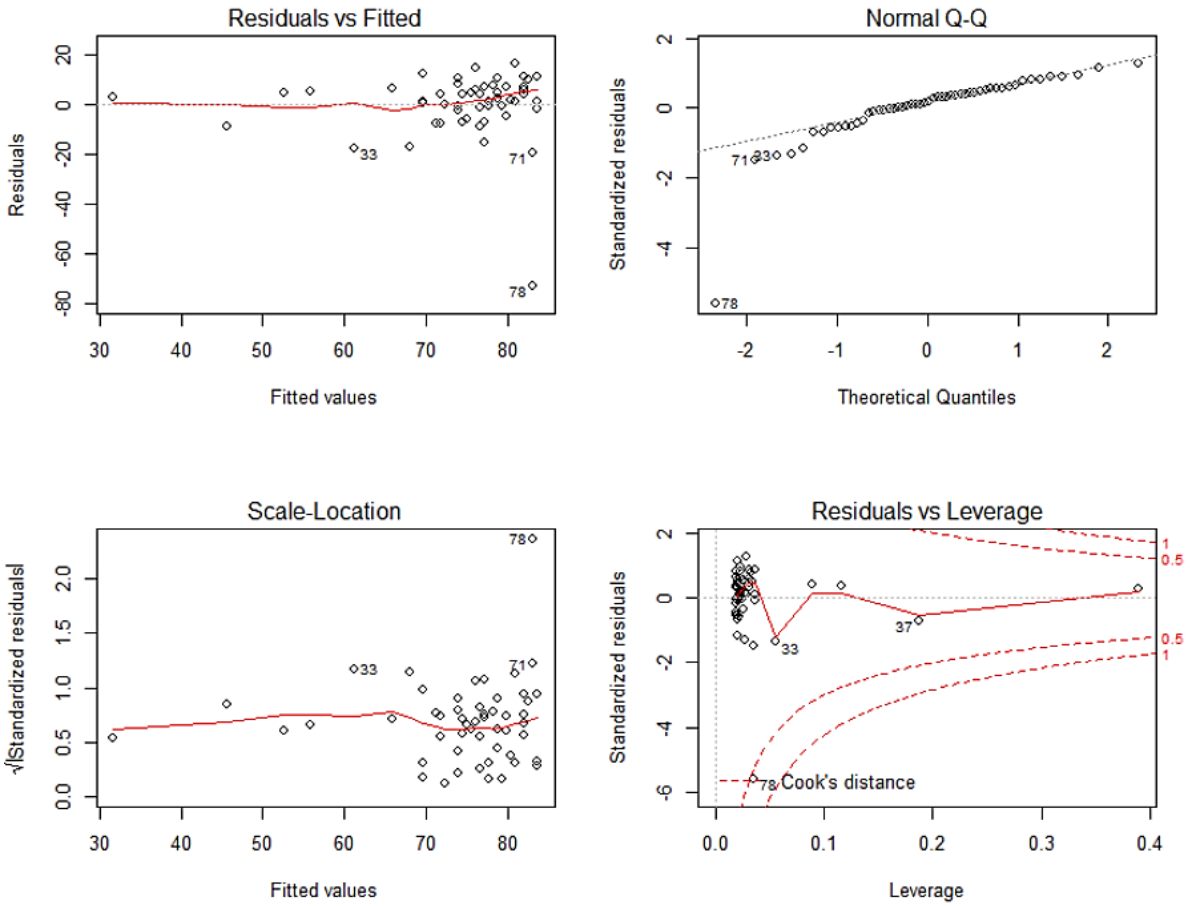


Figure 12.12. **Identification of outliers for the boys**

The first plot depicts residuals versus fitted values. Residuals are measured as follows: residual = observed *y* – model-predicted *y*.

So the further the observed *y* from the one predicted by the model, the poorer the prediction. The plot of residuals versus predicted values is useful for checking the assumption of linearity and homoscedasticity. If the model does not meet the linear model assumption, we would expect to see residuals that are very large (a big positive value or a big negative value).

To assess the assumption of linearity we want to ensure that the residuals are not too far away from 0.

To assess if the homoscedasticity assumption is met, we look to make sure that there is no pattern in the residuals and that they are equally spread around the $y = 0$ line. In our case, R identifies 3 points with high residuals, one of which has a really high one: point 78.

The second plot (QQ-plot) evaluates the normality assumption. It compares the residuals to "ideal" normal observations. We want our observations lie well along the 45-degree line in the QQ-plot, which is the case here, except for point 78.

The third plot is a scale-location plot (square rooted standardized residual vs predicted value). This is useful for checking the assumption of homoscedasticity. In this particular plot we are checking to see if there is a pattern in the residuals. In our case, things look OK. Point 78 is however away from the others.

Finally, the fourth plot is of the "Cook's distance", which is a measure of the influence of each observation on the regression coefficients. The Cook's distance statistic is a measure, for each observation in turn, of the extent of change in model estimates when that particular observation is omitted. Any observation for which the Cook's distance is close to 1 or more (above 0.5), or that is substantially larger than other Cook's distances (highly influential data points), requires investigation. Once more, in our case, point 78 is of concern.

Outliers may or may not be influential points**.** As stated before, influential outliers are of the greatest concern. They should never be disregarded. Careful scrutiny of the original data may reveal an error in data entry that can be corrected. They can be excluded from the final fitted model but they must be noted in the final report or paper.

In our case, one points stands out in all 4 graphs: point 78, so we will look at the correlation with and without this value.

Now we study the data for "female" (Fig. 12.13). We build data with the aid of the function *plot*():

```
> plot(fit.female)
```

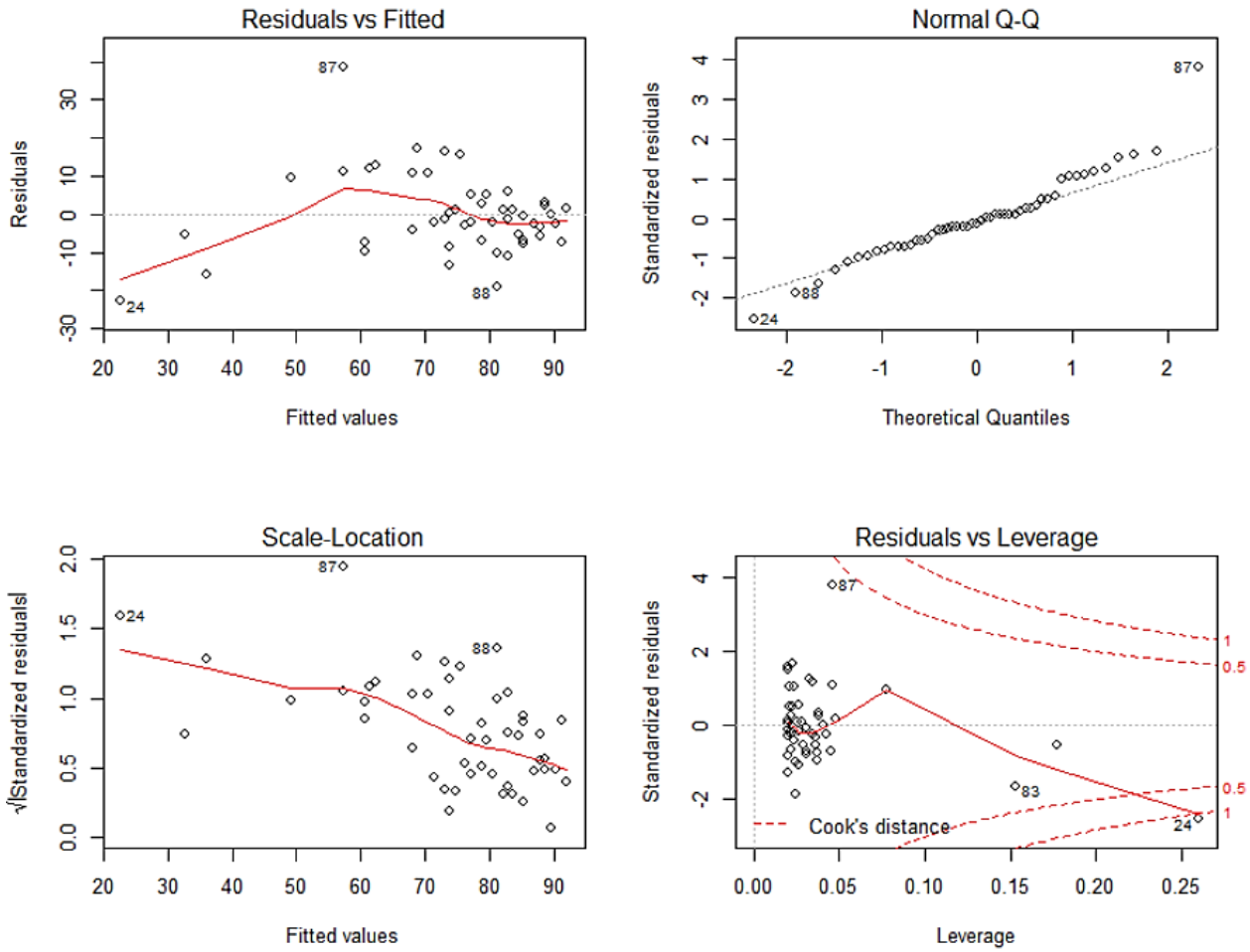


Fig. 12.13. **Identification of outliers for the girls**

For the girls, point 87 stands out though not as strikingly as point 78 for the boys. And it is below the threshold to be identified as an influential case (plos 4). We will however keep an eye on it (Fig 12.14). To get the output of the analysis:

> summary(fit.male)

From this output we get 4 important pieces of information. First the coefficients of the line of best fit: Intercept: 84.19 and slope: -0.53. So it goes: Anxiety = 84.19-0.53*Revise.

We can also see that the relationship between the two variables is highly significant: p < 2e-16. And finally R^2 = 0.3568: the model explains about 36 % of the variability observed in anxiety. We can get the coefficient of correlation by calculating the square root of R^2 or with the line below if we want to look at the relationships between all variables.

cor(exam.anxiety[exam.anxiety$Gender == "Male", c("Exam", "Anxiety", "Revise")])

For the females:

```
> summary(fit.female)
```

We get as a result: Anxiety = 91.94-0.82*Revise with p < 2e-16.

So a significant result again, with a higher intercept and a steeper slope as expected. And for the correlations:

```
> cor(exam.anxiety[exam.anxiety$Gender == "Female",
> c("Exam","Anxiety","Revise")])
```

Now what happens if we remove point 78 from the males dataset and rerun the analysis?

```
> fit.male2<-lm(Anxiety~Revise,
> data = exam.anxiety[exam.anxiety$Gender == "Male"&exam.anxiety$Code!
= 78,])
> summary(fit.male2)
```

We can notice that, without the influential outlier, the slope is steeper but most importantly R^2 jumps from 36 % to 65 % so a much better fit.

For the females:

```
> fit.female2<-lm(Anxiety~Revise,
> data = exam.anxiety[exam.anxiety$Gender == "Female"&exam.anxiety$Code!
= 87,])
> summary(fit.female2)
```

This model is better than the one with the outlier but the influence of point 87 is not as big. Keeping or removing the value is more debatable.

```
>plot(exam.anxiety$Anxiety~exam.anxiety$Revise, col =
exam.anxiety$Gender,pch = 16)

> legend("topright", title = "Gender", inset = .05, c("Female","Male"),
horiz = TRUE, pch = 16, col = 1:2)

> abline((fit.male), col = "red")
> abline((fit.female), col = "black")
> abline((fit.male2), col = "red", lty = 3)
> abline((fit.female2), col = "black", lty = 3)
```

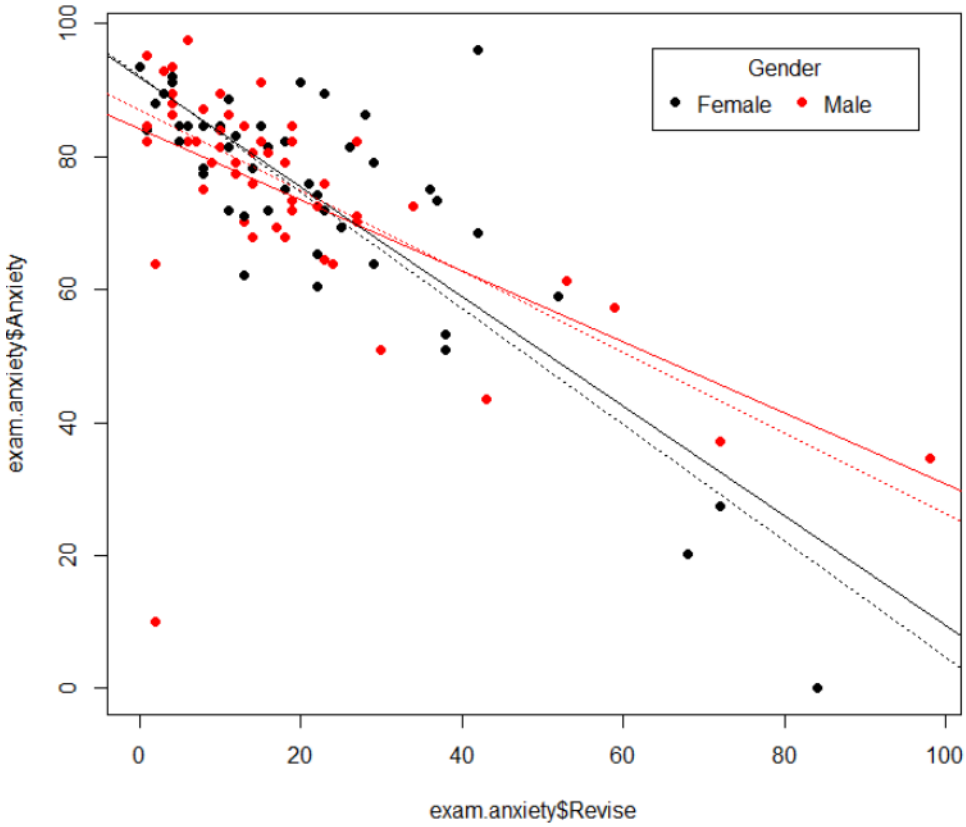


Fig. 12.14. **Trends for boys and girls after the removal of the outliers**

Conclusions

Almost all the basic features of R have been presented in these guidelines. You will get to know how to solve the statistic problems with the statistic software package R by following the R example step by step. Unfortunately, there are more useful features, which have not been studied and mentioned here because of the time limitation of the project. Also some R functions were not introduced in detail due to the lack of space. You can get detailed explanation of the functions with the help of the *help*() function. But these introduced R functions are enough for the beginner to start using R for the statistics analysis. You can get more information from the books and publications listed in the bibliography.

Bibliography

1. Dalgaard Peter. Introductory Statistics with R. / P. Dalgaard. – 3rd ed. – New York : McGraw-Hill, Inc., 1995. – 370 p.

2. Field A. Discovering statistics using R / A. Field, J. Miles, Z. Field. – 1st ed. – London : Sage, 2012. – 546 p.

3. Milton J. S. Introduction to Probability and Statistics / J. S. Milton, J. C. Arnold. – 4th ed. – New York : McGraw-Hill, Inc., 2001. – 798 p.

4. Montgomery D. C. Design and Analysis of Experiments / D. C. Montgomery. – 3rd ed. – New York : John Wiley & Sons, Inc., 1991. – 432 p.

5. Verzani John. Simple R – Using R for Introductory Statistics [Electronic resource] / J. Verzani. – Access mode : https://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf.

НАВЧАЛЬНЕ ВИДАННЯ

ТЕОРІЯ ЙМОВІРНОСТЕЙ ТА МАТЕМАТИЧНА СТАТИСТИКА

Методичні рекомендації до лабораторних робіт з використанням програмного середовища R

для студентів усіх спеціальностей першого (бакалаврського) рівня

(англ. мовою)

Укладачі: **Малярець** Людмила Михайлівна
Тижненко Олександр Григорович

Відповідальний за видання *Л. М. Малярець*

Редактор *З. В. Зобова*

Коректор *З. В. Зобова*

Розглянуто методологічні аспекти вивчення проблем ймовірностей та статистики за допомогою програмного забезпечення R шляхом розв'язання набору стандартних задач. Наведено декілька базових принципів використання R для розв'язання задач з теорії ймовірностей та статистики на реальних прикладах. Такий підхід дозволить студентам, які мають базові знання з математичного аналізу та математичних методів в економіці, швидко оволодіти методами розрахунків у R. Тому ці методичні рекомендації можуть бути використані студентами економічних спеціальностей як основа для засвоєння курсів імовірностей та статистики з використанням програмного середовища R.

Рекомендовано для студентів усіх спеціальностей першого (бакалаврського) рівня.

План 2018 р. Поз. № 36 ЕВ. Обсяг 114 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***