Medical Statistics

2021-11-17

Contents

About this course

In this course an introduction to basic statistical methods useful for biomedical data analysis will be given. During the course we will alternate between lectures and practicals, allowing for plenty of interaction and illustration with examples of practical interest.

The course requires little prior statistics knowledge, but assumes participants are able to work with R, R packages and RMarkdown for the practicals. Participants with no or little experience in R are strongly advised to follow an introductory R course prior to following this course, such as the one we offer.

The course program includes: exploratory data analysis, basic statistical tests, methods for count data tables, linear and logistic regression, as well as basic methods for survival data analysis. For each method, power analysis and sample size determination will be handled.

Teachers

- Renee Menezes, Biostatistics Center, NKI (**coordinator**)
- Renaud Tissier, Biostatistics Center, NKI
- Leyla Azarang, Biostatistics Center, NKI

Goals & Topics

After the course you will be able to:

- Perform data exploration analysis
- Perform comparison between two different groups using statistical tests
- Perform linear and logistic regression
- Perform survival data analysis
- Use the R software to perform these tasks

We will cover the following topics:

- Probabilistic distributions
- Hypothesis testing

- Linear regression
- Logistic regression
- Survival data analysis
- Power analysis

Prerequisites

The course assumes no prior programming knowledge. Elementary statistics knowledge is necessary.

Participants must **bring own laptops** capable of running RStudio.

Before the course **please prepare your laptop**:

1. install R, an open-source, free environment for statistical computing and graphics. You can find instructions for downloading and installing it from one of the CRAN mirrors, for example from the Univ. of Gent or from the Imperial College. A full list of mirrors can be found here.

2. install RStudio. Go to the RStudio download page, select a version of RStudio appropriate for your laptop, download it and then install. Please check whether you can start RStudio.

3. install RMarkdown, a very nice and easy tool to produce reports using RStudio. It is made available as an R package for Rstudio. One easy way to install it is as follows:

i) open RStudio

ii) click on the "File" menu on the top left, and choose "New file">"R Markdown". If RMarkdown is not yet installed on your machine, this will prompt you to install it and any packages required. Just follow the instructions that appear on the screen.

4. Several packages are required to perform the course. These packages can be installed before the course to gain some time. The required packages are:

- factoextra
- gplots
- pwr
- statmod

Materials

After the course material .zip file is downloaded, the course material can be assessed:

- as HTML pages by opening `index.html` in any browser

- by clicking on the `RcourseNKI.Rproj` file, which will open the entire course as an R project
- via the course source git repository

The materials contain a `data` directory with the **data files** used in the presentations/tasks. The directory can be also accessed at https://github.com/rxmenezes/RcourseNKI/tree/master/data

Programme

Third NKI edition, Novermber 22nd, 23rd, 24th, 25th and 26th 2021

This course will be given online via Zoom. All course days are in the period 9:00-16:00, with the last hour reserved for general Q&A.

Chapter 1

Introduction and exploratory data analysis

1.1 Introduction to probabilistic distributions

1.1.1 Motivation

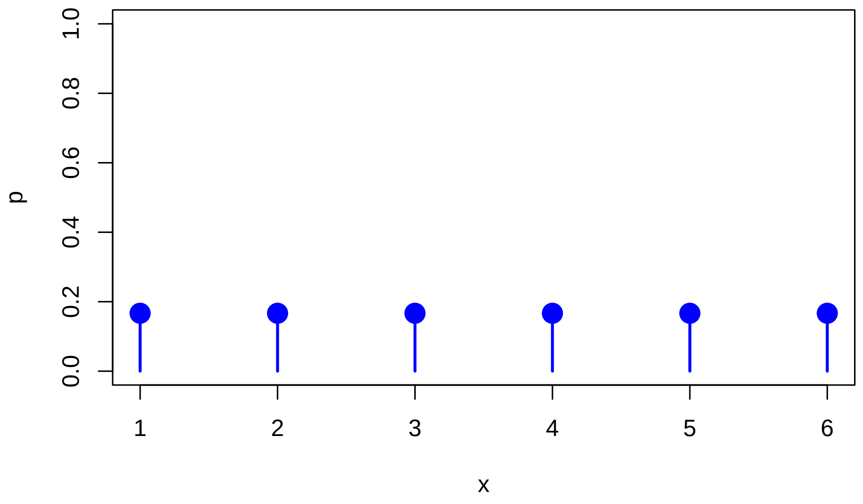
In this section we will see some of the most commonly used distributions used in statistics. It is important to learn to recognize these distributions and some of their properties to better use statistical models and tests in your analysis.

1.1.2 Discrete random variables

Discrete random variables are those taking only integer values. Common distributions for those variables are the uniform distribution, the Bernoulli distribution and the binomial distribution.

The **discrete uniform distribution** assigns the exact same probability to a finite set of values. A common example is throwing a fair dice. After throwing the dice, each number 1, 2, 3, .., 6 has the same probability to be obtained, equal to 1/6.

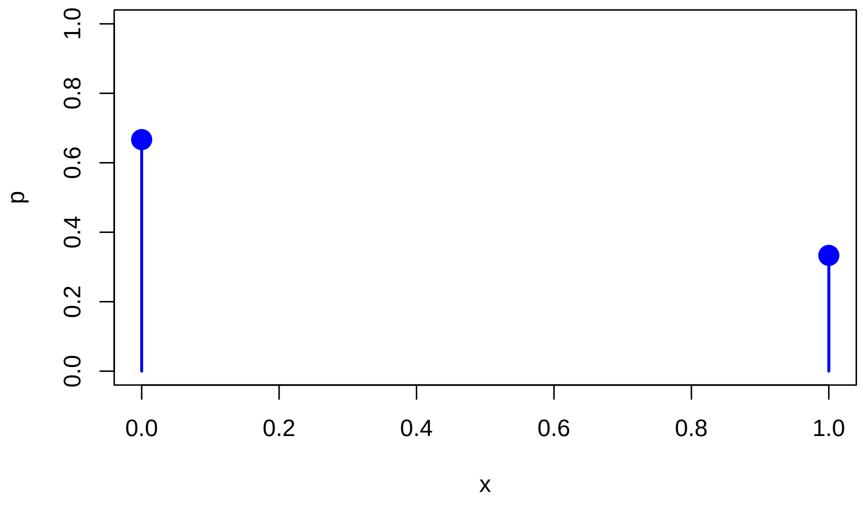
```
#plotting probability discrete uniform distribution

x <- c(1:6)
y <- rep(1/6,6)

plot(x,y,type="h",xlim=c(1,6),ylim=c(0,1),lwd=2,col="blue",ylab="p")
points(x,y,pch=16,cex=2,col="blue")
```



Consider another example, where you flip a coin: heads is registered as 1, and tails as 0. The **Bernoulli distribution** can be used to study this variable: it assigns probability p to 1 (heads), and *1-p* = q to 0 (tails).

You could also have a variable that can yield many possible values, which is transformed to yield only 1. For example, let us throw a fair dice and check if the number is equal to or lower than 2. The probability of a success (obtaining a value below or equal 2) corresponds to the dice returning 1 or 2, so is equal to 1/3. Therefore, the probability of throwing a number larger than 2 is 2/3 (corresponding to values 3, 4, 5, 6). The analysis of the association between covariates and the probability of success is often done via logistic regression.

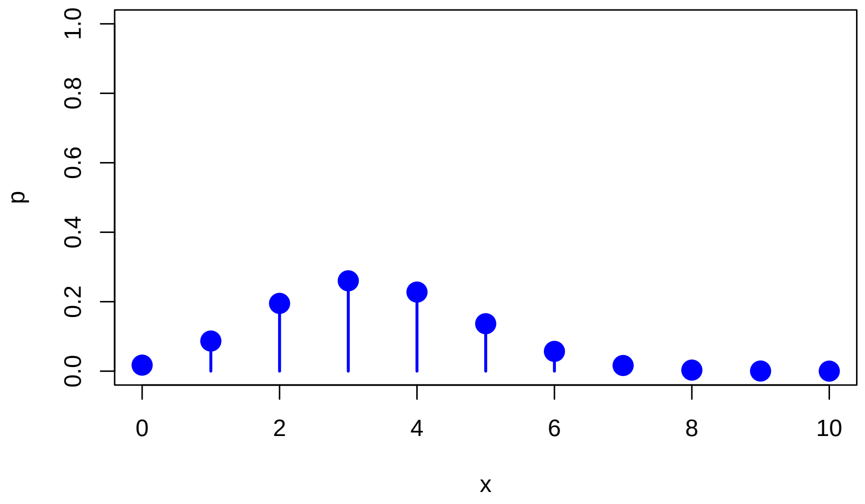
```
#plotting probability Bernoulli distribution

x <- c(0:1)
y <- c(2/3,1/3)

plot(x,y,type="h",xlim=c(0,1),ylim=c(0,1),lwd=2,col="blue",ylab="p")
points(x,y,pch=16,cex=2,col="blue")
```



Now imagine that you throw a fair coin not once, but 10 times. Then the number of times that the result was heads can be modelled by the **binomial distribution**. It describes the probabilities of results of several independent throws, each of which with the same probability of success (heads).

If we go back to the previous dice example, we can compute the probability to obtain 4 successes (dice gives 1 or 2) in 10 trials. This can be done with the function `dbinom`. The probability is 0.2276076.

```r
#plotting probability binomial distribution

x <- c(0:10)
y <- dbinom( x, 10, 1/3)

plot(x,y,type="h",xlim=c(0,10),ylim=c(0,1),lwd=2,col="blue",ylab="p")
points(x,y,pch=16,cex=2,col="blue")
```

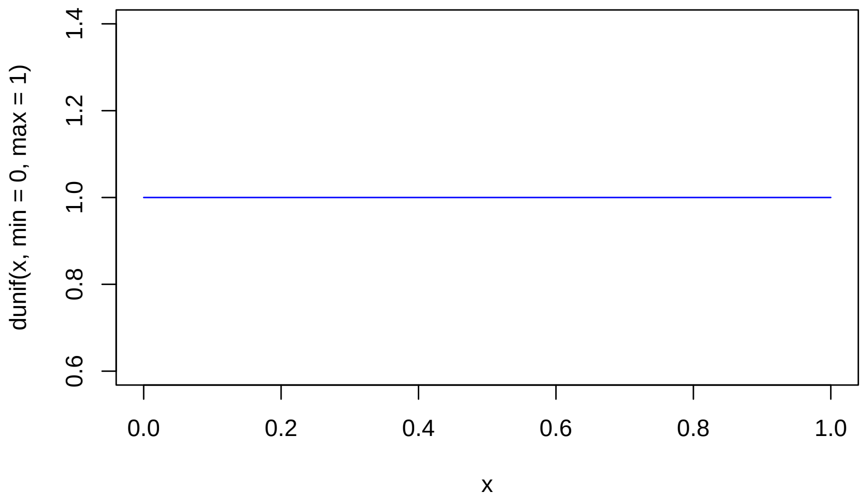


1.1.3 Continuous random variable

A continuous random variable can have an infinite number of possible values. Therefore, the probability to obtain any one value in particular is equal to 0. For these variables, we do not consider the probability of individual values, but instead we consider the probability associated with intervals, say all values between 0 and 1. We call the function describing the probabilities the *density probability function*. We will now look at some important continuous distributions.

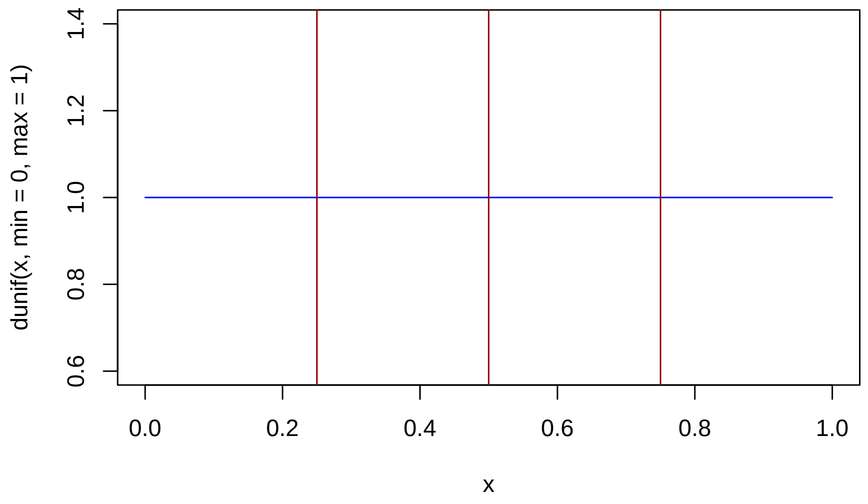
The **continuous uniform distribution** has a similar definition to the discrete uniform distribution. The probability density function yields the same value across the range of possible values. To illustrate this, let us plot the density of a continuous uniform distribution between 0 and 1.

```
#Plot the density
curve(dunif(x, min= 0, max= 1), col="blue", from = 0, to = 1 )
```



Quartiles are values that leave probability of 1/4, 2/4 (or 1/2) and 3/4 below them. Let us add the quartiles of the uniform distribution to the plot.

```
#Plot the density and quartiles
curve(dunif(x, min= 0, max= 1), col="blue", from = 0, to = 1 )
abline(v = qunif(c(0.25, 0.5, 0.75), min= 0, max= 1), col= 'darkred')
```



The quartiles split the range of values into 4 parts, each with 1/4 probability. So the probability to observe values between 2 subsequent quartiles is 1/4, or 25%. This means that the probability to observe a value between the first and the third quartiles is equal to 1/4+1/4 = 1/2, or 50%.

Quartiles can also be computed for a sample: they are values that leave 1/4, 1/2 and 3/4 of the observations below them. For example, consider the following sample:

```
1:12
```

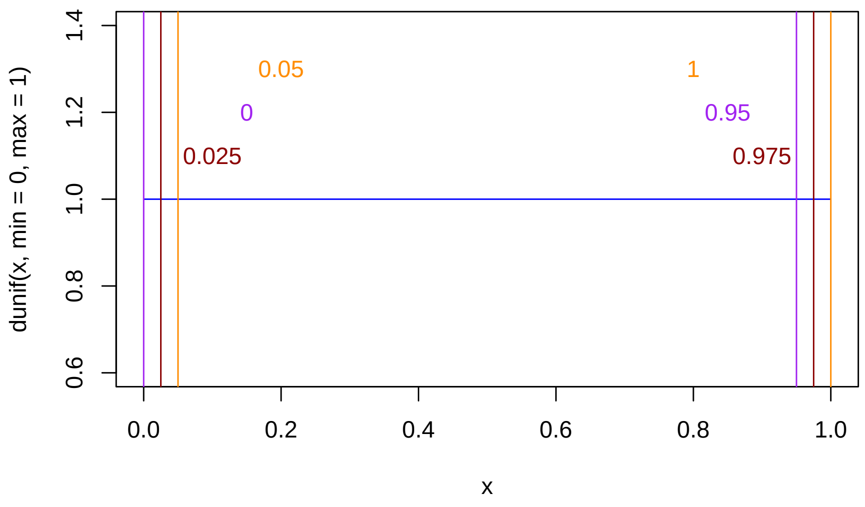
```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Then we can take the first quartile as 3, since it has observations 1, 2, 3 up to and below it, 25% of all observations. Of course, a value larger than 3, but smaller than the next value (4) can also be used.

In general, for a continuous variable, the probability to observe a value between 2 given values is equal to the area under the probability density function between these two points. The generalization of the quartiles for any probability is called **quantile**: the 5%-quantile is the value that leaves 5% (or 0.05) of probability below it. Similarly, the 95%-quantile is the value that leaves 95% (or 0.95) of probability below it. Below we illustrate these with the probability density function for the uniform between 0 and 1.

```
#Plot of the density and important quantiles
#of a continuous uniform distribution
curve(dunif(x, min= 0, max= 1), col="blue", from = 0, to = 1 )
```

```
mcol <- 'darkred'
abline(v = qunif(c(0.025, 0.975), min= 0, max= 1), col= mcol)
text(0.1, 1.1, col = mcol, labels = "0.025")
text(0.9, 1.1, col = mcol, labels = "0.975")

mcol <- 'purple'
abline(v = qunif(c(0, 0.95), min= 0, max= 1), col= mcol)
text(0.15, 1.2, col = mcol, labels = "0")
text(0.85, 1.2, col = mcol, labels = "0.95")

mcol <- 'darkorange'
abline(v = qunif(c(0.05, 1), min= 0, max= 1), col= mcol)
text(0.2, 1.3, col = mcol, labels = "0.05")
text(0.8, 1.3, col = mcol, labels = "1")
```



Quantiles are used in statistical testing to build the confidence intervals and compute p-values. Similar to quartiles, we can also obtain quantiles for samples of values. For example, for the sample

```
1:10
```

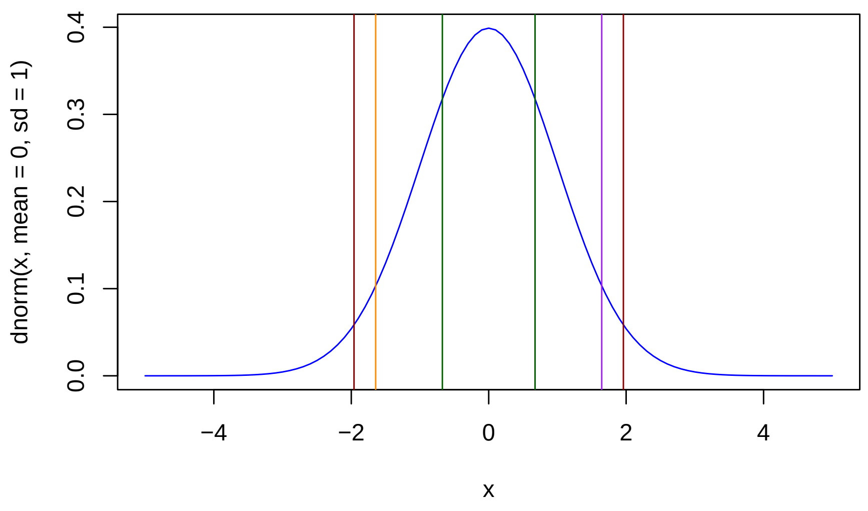
```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

the 10% quantile is 1 (or any number larger than 1 but smaller than 2), and the 90% quantile is 9.

The most common distribution encountered in the nature is the normal or

Gaussian distribution. This distribution is symmetric, and is defined by its expectation μ and its variance σ^2, i.e. $N(\mu, \sigma^2)$. Its values range from $-\infty$ to ∞. Note that it is important to make the difference between the expectation of the distribution and the sample mean derived from this distribution. Indeed, these two will be equal only if all possible samples are included to compute sample mean.

```
#Plot of the density and quantiles of a normal distribution
curve(dnorm(x, mean = 0, sd = 1), col = 'blue', from = -5, to = 5)
abline(v = qnorm(c(0.25, 0.75), mean = 0, sd = 1), col= 'darkgreen')
abline(v = qnorm(c(0.025, 0.975), mean = 0, sd = 1), col= 'darkred')
abline(v = qnorm(c(0, 0.95), mean = 0, sd = 1), col= 'purple')
abline(v = qnorm(c(0.05, 1), mean = 0, sd = 1), col= 'darkorange')
```

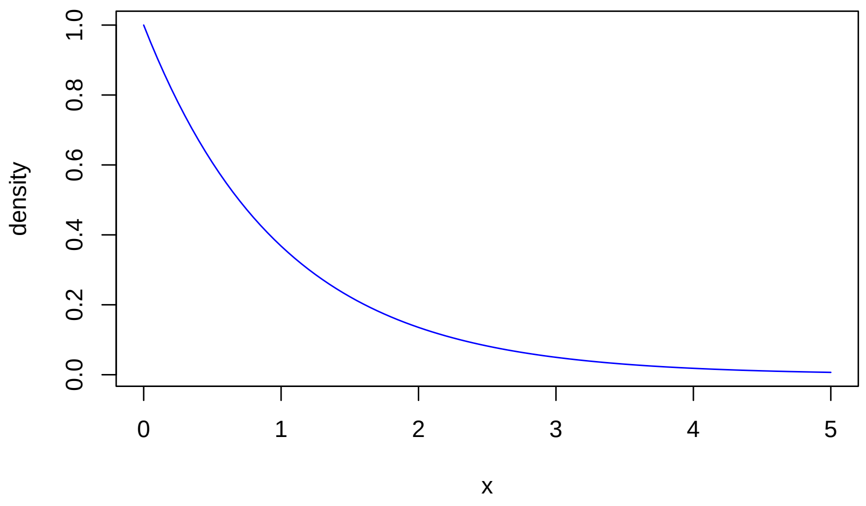


The normal distribution is often referred to as the *Gaussian* distribution, due to the work of C. F. Gauss in this area. The normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$ is referred to as the standard normal distribution.

This distribution is very important in statistics due to one of its properties, the **central limit theorem**: it states that if you have a population with expectation μ and standard deviation σ and take sufficiently large random samples from the population with replacement, then the sample means will follow approximately a normal distribution. This will hold true regardless of whether the source population is normal or skewed, provided the sample size is sufficiently large (usually $n > 30$) and the variance of the original distribution is finite.

Let's illustrate the central limit theorem with the gamma distribution that will be seen in another chapter of this course. First consider the probability density function of the gamma distribution:

```
curve(dgamma(x, shape = 1, rate = 1), from = 0, to = 5, col= 'blue'
      , ylab= 'density')
```

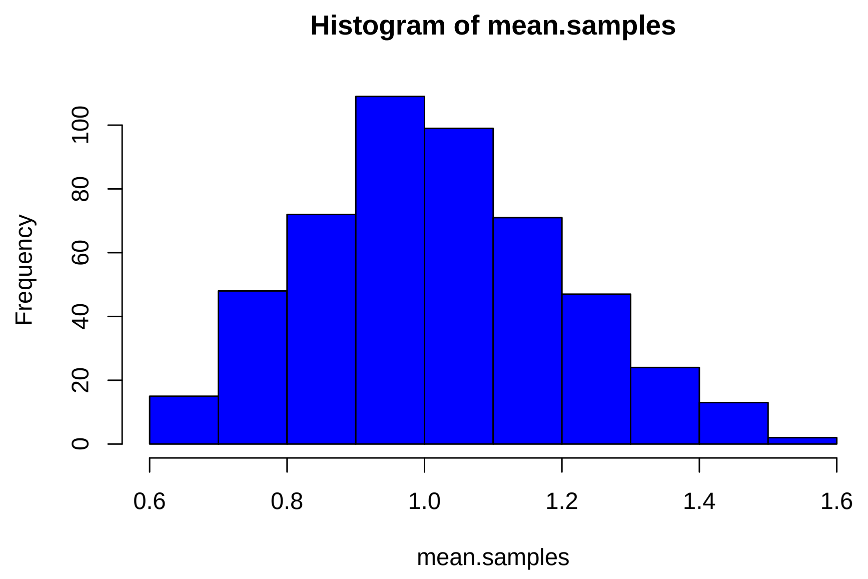


Now we simulate 500 samples, each of 30 values, from the same gamma distribution. We then compute the mean for each sample.

```
mean.samples <- NULL
for (k in 1:500){
  data.sample <- rgamma( 30 , shape = 1, rate = 1)
  mean.samples <- c(mean.samples, mean(data.sample))
}
```

The histogram of the means computed is:

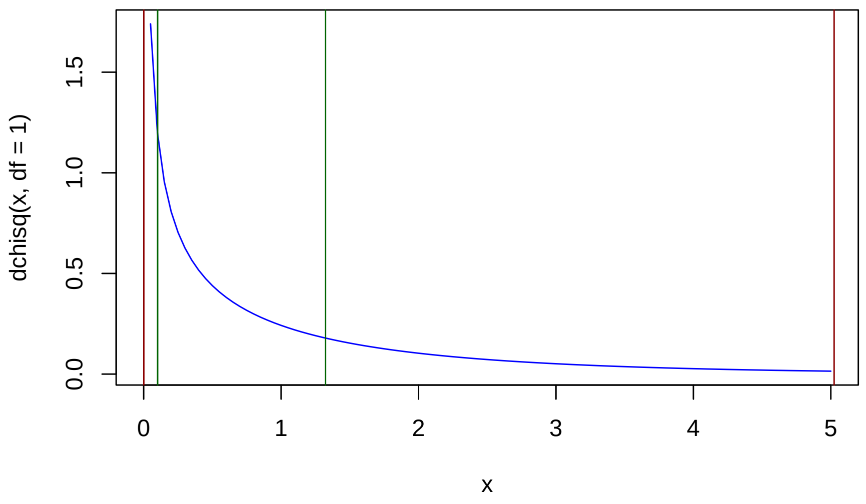
```
hist(mean.samples, col='blue')
```



Essentially the arithmetic mean smoothes out small random variations between samples, which can go in either direction. As a result, its distribution will not be skewed, as the original observations, but symmetric and with little variation around its own mean. This is then the normal distribution.

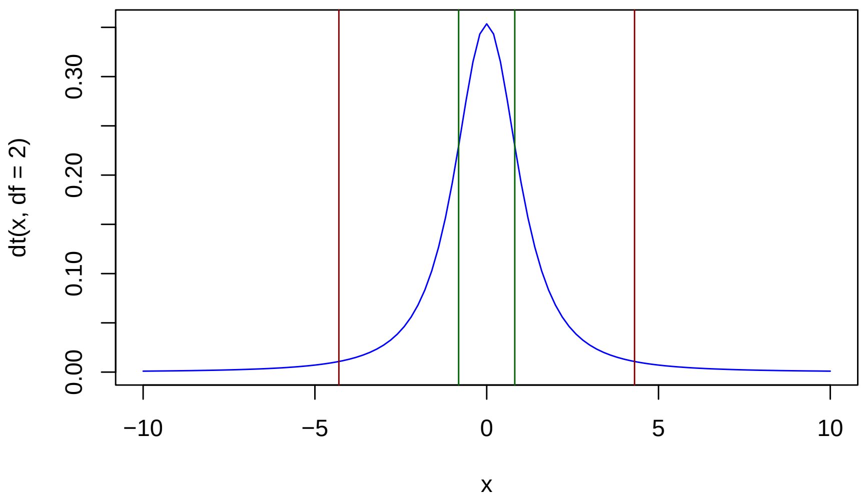
The chi-square distribution, represented by χ^2 is defined at the sum of k squared independent random variables, each following a normal distribution with mean 0 and variance equal to 1. The chi-square only has one parameter, k, called *degrees of freedom*. This distribution is often used for statistical testing. Its probability density function is shown below.

```
#Plot of the density and quantiles of a chi square distribution
#with 1 degree of freedom
curve(dchisq(x, df = 1), col = 'blue', from = 0, to = 5)
abline(v = qchisq(c(0.25, 0.75), df = 1), col= 'darkgreen')
abline(v = qchisq(c(0.025, 0.975), df = 1), col= 'darkred')
```



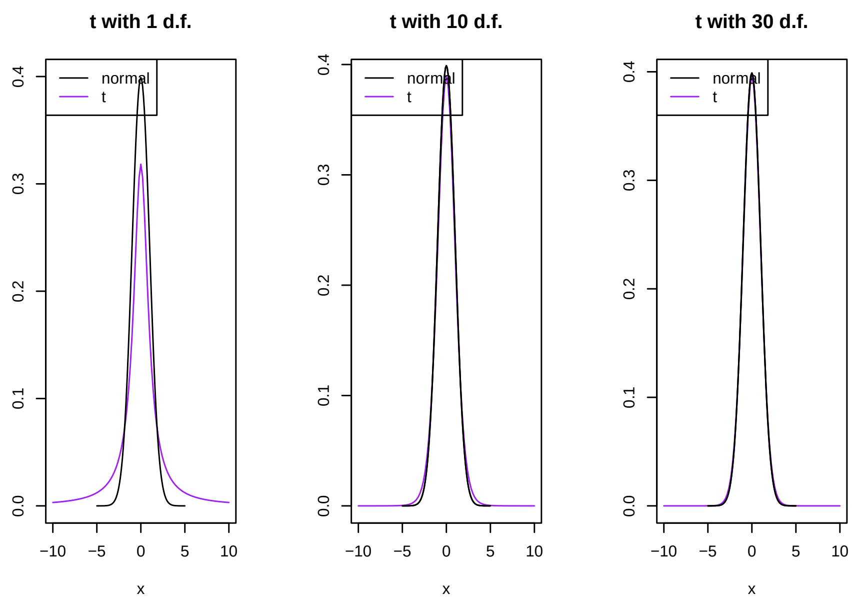
Finally, the Student's t distribution is very important as it is used in the well-known Student's-t test to compare means between two samples. This distribution is defined by its number of degrees of freedom k. This distribution, as the normal distribution, is symmetric, as we can see from its probability density function below.

```r
#Plot of the density and quantiles of a chi square distribution
#with 2 degree of freedom
curve(dt(x, df = 2), col = 'blue', from = -10, to = 10)
abline(v = qt(c(0.25, 0.75), df = 2), col= 'darkgreen')
abline(v = qt(c(0.025, 0.975), df = 2), col= 'darkred')
```



This distribution has the same *bell* shape as the normal distribution. However, this shape is wider for the Student's-t, becoming narrower as the number of degrees of freedom increases.

```
par(mfrow = c(1, 3))
x <- seq(from = -5, to = 5, by = .1)
curve(dt(x, df = 1), col = 'purple', from = -10, to = 10,
      main = "t with 1 d.f.",
      ylim = c(0, 0.4), ylab = "")
lines(x, dnorm(x, mean = 0, sd = 1), col = 'black')
legend("topleft", legend = c("normal", "t"), lty = "solid"
       ,
       col = c("black", "purple"))

curve(dt(x, df = 10), col = 'purple', from = -10, to = 10,
      main = "t with 10 d.f.", ylab = "")
lines(x, dnorm(x, mean = 0, sd = 1), col = 'black')
lines(x, dnorm(x, mean = 0, sd = 1), col = 'black')
legend("topleft", legend = c("normal", "t"), lty = "solid", col = c("black", "purple")

curve(dt(x, df = 30), col = 'purple', from = -10, to = 10,
      main = "t with 30 d.f.", ylab = "")
lines(x, dnorm(x, mean = 0, sd = 1), col = 'black')
lines(x, dnorm(x, mean = 0, sd = 1), col = 'black')
legend("topleft", legend = c("normal", "t"), lty = "solid",
        col = c("black", "purple"))
```



Quick task(s):

Solve the task(s), and check your solution(s) here.

1.2 Data representation

1.2.1 Motivating example

A researcher receives the dataset `decathlon2` from the `factoextra` package. This dataset describe the performances of several athletes in two different sport events. Before starting any analysis, the researcher wants to explore the data, for example by visualising the data.

Below we load the data.

```
library(factoextra)

data(decathlon2)
# Display a description of the dataset
str(decathlon2)
```

```
## 'data.frame':    27 obs. of  13 variables:
##  $ X100m       : num  11 10.8 11 11.3 11.1 ...
##  $ Long.jump   : num  7.58 7.4 7.23 7.09 7.3 7.31 6.81 7.56 6.97 7.27 ...
##  $ Shot.put    : num  14.8 14.3 14.2 15.2 13.5 ...
##  $ High.jump   : num  2.07 1.86 1.92 2.1 2.01 2.13 1.95 1.86 1.95 1.98 ...
##  $ X400m       : num  49.8 49.4 48.9 50.4 48.6 ...
##  $ X110m.hurdle: num  14.7 14.1 15 15.3 14.2 ...
##  $ Discus      : num  43.8 50.7 40.9 46.3 45.7 ...
##  $ Pole.vault  : num  5.02 4.92 5.32 4.72 4.42 4.42 4.92 4.82 4.72 4.62 ...
##  $ Javeline    : num  63.2 60.1 62.8 63.4 55.4 ...
##  $ X1500m      : num  292 302 280 276 268 ...
##  $ Rank        : int  1 2 4 5 7 8 9 10 11 12 ...
##  $ Points      : int  8217 8122 8067 8036 8004 7995 7802 7733 7708 7651 ...
##  $ Competition : Factor w/ 2 levels "Decastar","OlympicG": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Display the first lines of the table
head(decathlon2)
```

```
##           X100m Long.jump Shot.put High.jump X400m X110m.hurdle Discus
## SEBRLE    11.04      7.58    14.83      2.07 49.81        14.69  43.75
## CLAY      10.76      7.40    14.26      1.86 49.37        14.05  50.72
## BERNARD   11.02      7.23    14.25      1.92 48.93        14.99  40.87
## YURKOV    11.34      7.09    15.19      2.10 50.42        15.31  46.26
## ZSIVOCZKY 11.13      7.30    13.48      2.01 48.62        14.17  45.67
## McMULLEN  10.83      7.31    13.76      2.13 49.91        14.38  44.41
##           Pole.vault Javeline X1500m Rank Points Competition
## SEBRLE          5.02    63.19  291.7    1   8217    Decastar
## CLAY            4.92    60.15  301.5    2   8122    Decastar
## BERNARD         5.32    62.77  280.1    4   8067    Decastar
## YURKOV          4.72    63.44  276.4    5   8036    Decastar
## ZSIVOCZKY       4.42    55.37  268.0    7   8004    Decastar
## McMULLEN        4.42    56.37  285.1    8   7995    Decastar
```

You will use this dataset in exercises.

1.2.2 Working example

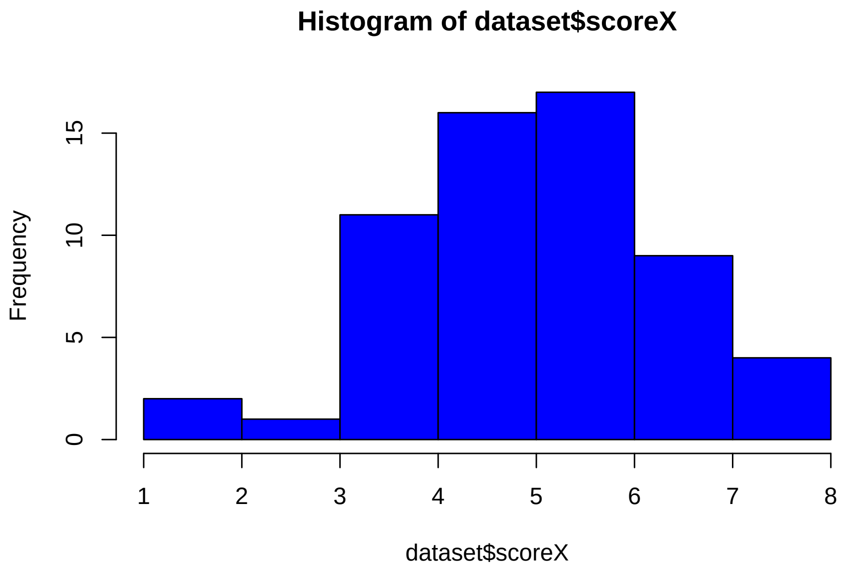
As a small, simple example, we consider the following simulated dataset: 2 continuous variables, `scoreX` and `scoreY`, and a categorical variable representing the membership of group A or B.

```r
set.seed(352)
sample.size<-60

scoreX <- rnorm(sample.size, mean = 5, sd = 1.2)
scoreY <- rgamma(sample.size, shape = 2, scale = 1)
group <- rep(c('A', 'B'), each = sample.size/2)
```

```r
dataset <- data.frame(scoreX = scoreX, scoreY = scoreY,
                      group = group, stringsAsFactors = T)
```

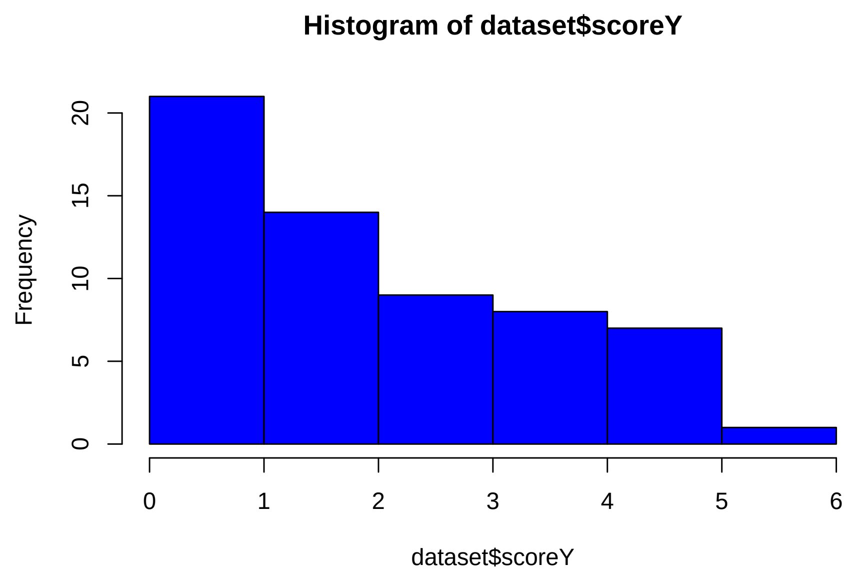
Before doing any statistical analysis, it is important to look at the distributions of the variables available. Indeed, looking at the shape of the distributions can help choosing adequate analysis methods. Let's look at the distribution of the variable `scoreX` by plotting an histogram of the variable. This can be done using the function hist in R:

```r
#plot the histogram of a variable
hist(dataset$scoreX, col = 'blue')
```



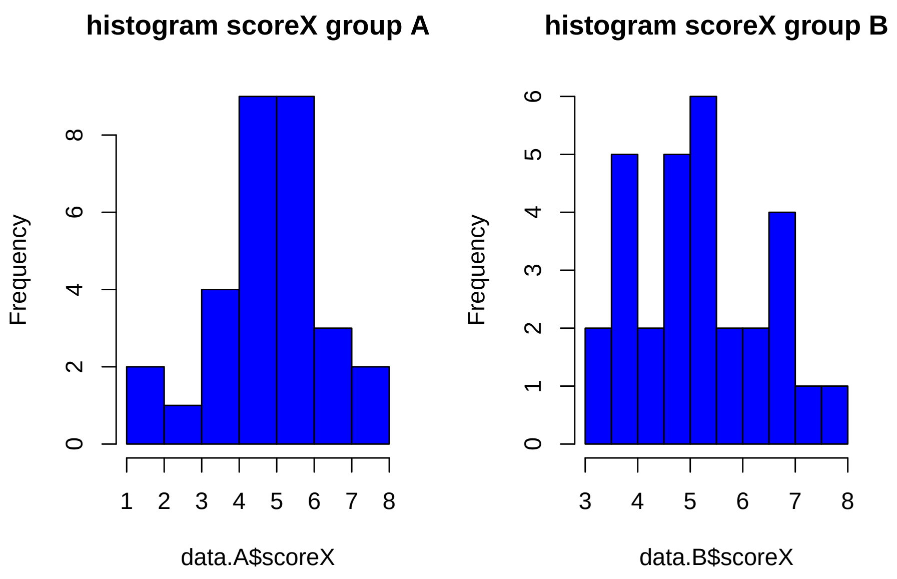
The histogram displays bars representing counts of the number of observations falling into specific bins (intervals of values). This gives us an idea about the shape of the distribution of a variable (in the example above, of `scoreX`). As we simulated values for =`scoreX` using a normal distribution, we expect the histogram to reflect this - be symmetric and to not have too many extremen values. Compare the histogram above to the one for the variable `scoreY`:

```r
#plot the histogram of a variable
hist(dataset$scoreY, col='blue')
```



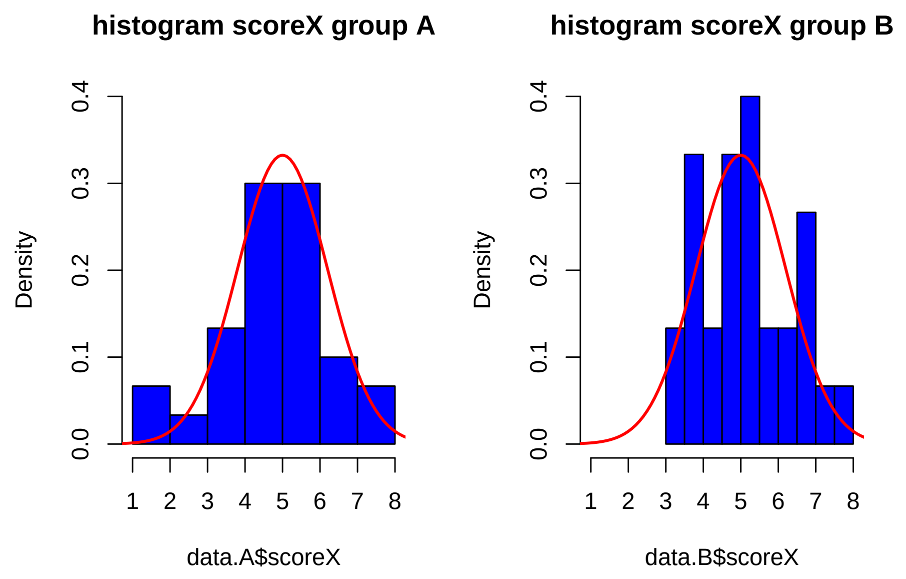
We can also display histograms per group to compare the shape of the distributions between two groups. To do so we need to split the data in two:

```
#Creation dataset group A
data.A <- dataset[dataset$group == 'A',]

#Creation dataset group B
data.B <- dataset[dataset$group == 'B',]

#plot the histograms of scoreX for both groups
par(mfrow = c(1,2))
hist(data.A$scoreX, col = 'blue', main = 'histogram scoreX group A' )
hist(data.B$scoreX, col = 'blue', main = 'histogram scoreX group B' )
```



Here, the histograms of subsets of `scoreX` look quite different. However, the values from both groups are drawn using the same distribution. The differences observed are only due to randomness. We can, for example, add to these histograms the density plot of the distribution used to simulate the `scoreX` variable:

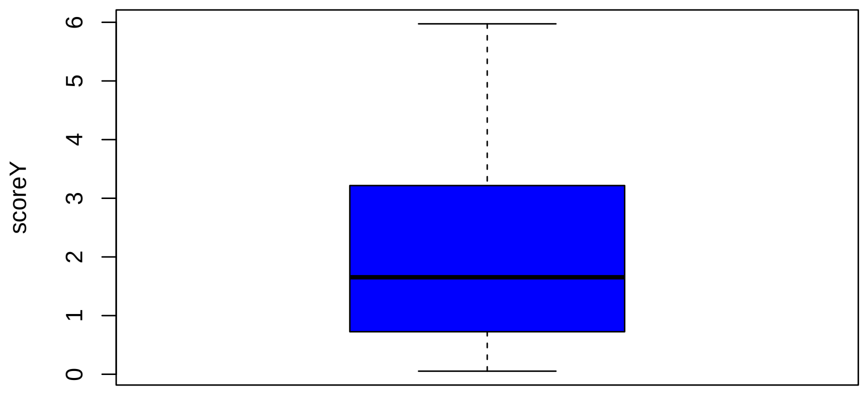
```
#plot the histograms of scoreX for both groups with added density
par(mfrow = c(1,2))


hist(data.A$scoreX, col = 'blue', prob = TRUE,
     main = 'histogram scoreX group A', ylim = c(0, 0.4), xlim= c(1, 8))
x <- seq(from = -5, to = 15, by = .1)
lines(x, dnorm(x, mean = 5, sd = 1.2), col = 'red', lwd = 2)
hist(data.B$scoreX, col = 'blue', prob = TRUE,
     main = 'histogram scoreX group B', ylim = c(0, 0.4), xlim= c(1, 8) )
lines(x, dnorm(x, mean = 5, sd = 1.2), col = 'red', lwd = 2)
```



As we can see, it is not possible to determine from these plots that the distributions of `scoreX` in both groups are different. This is why statistical testing is needed: to check if apparent differences could be due to chance or not.

Another important and useful way to represent the data are boxplots. Boxplots are graphical representations of summary measures of a distribution represented, as the name indicates, in the shape of a box. We will draw a boxplot of the variable `scoreY`:

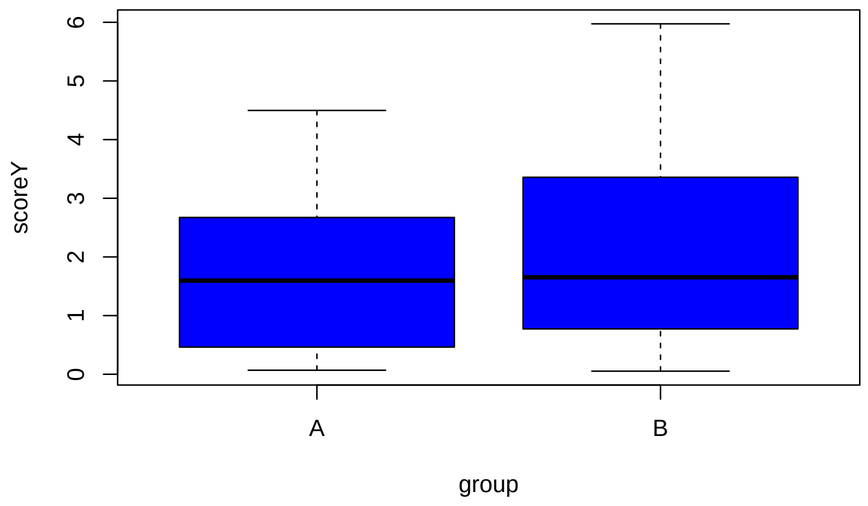
```
#Boxplot of a variable
boxplot(dataset$scoreY, col = 'blue', ylab = 'scoreY')
```



The box represents the space between the first and the third quartiles of the variable. The thick horizontal line represents the median of the variable, i.e. the second quartile leaving 50% of values below it. Finally, the top and bottom lines represent a space equal to 1.5 the boxsize from the nearest edge of the box. Any values above or below these lines are represented as points and are considered as extreme values, and possible outliers.

To make boxplots of the same variable for different groups is really easy, by using a formula as shown below:

```r
#Boxplot of a variable for different gruops
boxplot(dataset$scoreY ~ dataset$group, col = 'blue', ylab = 'scoreY'
        , xlab = 'group')
```



As we can see in these boxplots, the median value for both groups is similar but the group B box is wider, indicating more variations in this group. Again, statistical methods are needed to prove if this difference is real or not.

Quick task(s):

Solve the task(s), and check your solution(s) here.

1.3 Mean-Variance

1.3.1 Working example

We now want to compute summary measures for the variables in a dataset. Consider the same dataset, `dataset`, that we simulated in the previous part.

By looking at the histograms and boxplots, we noticed possible differences between the variables' values in the different groups. We now want to derive summary measures to see if they corroborate what the eyes could see. The most common summary measures of continuous variables are the mean and the standard deviation.

The sample mean represents the central value of the combined values of all samples for a specific variable. The mean is computed as the sum of all the values of a variable divided by the total number of values measured:

$\overline{x} = \frac{\sum_i^N (x_i)}{N}$

To compute mean values of `scoreX` and `scoreY` for the groups `A` and `B`.

```
#Creation dataset group A containing only scoreX and scoreY
data.A <- dataset[dataset$group == 'A', c('scoreX', 'scoreY')]

#Creation dataset group B containing only scoreX and scoreY
data.B <- dataset[dataset$group == 'B', c('scoreX', 'scoreY')]

#Computation of the mean for each variable of the dataset for group A
mean.variables.groupA <- apply(data.A, 2, mean, na.rm=T)
mean.variables.groupA
```

```
##   scoreX   scoreY
## 4.880274 1.750616
```

```
data.A
```

```
##      scoreX     scoreY
## 1  5.679002 0.53389766
## 2  1.810394 0.39127041
## 3  3.794993 1.01559188
## 4  5.691155 0.06849886
## 5  3.655173 1.42978089
## 6  2.996220 2.53244125
## 7  5.607361 1.49633795
## 8  7.283853 3.78528618
## 9  5.930352 2.94015306
## 10 7.244870 1.69968558
## 11 4.160336 4.49709044
## 12 1.985166 0.23456286
## 13 4.951650 1.69578956
## 14 3.784455 0.96296143
## 15 6.414988 4.29071898
## 16 5.336681 2.67268555
## 17 4.434049 2.34449025
## 18 5.087762 2.44975321
## 19 5.402171 0.44416274
## 20 4.198960 0.22505876
## 21 5.123555 1.90313042
## 22 3.583251 4.09168428
## 23 4.355833 0.46274765
## 24 4.544629 1.77943488
## 25 6.641351 1.30219871
## 26 6.991088 3.20300464
## 27 4.901021 0.84321431
## 28 4.769678 0.34874244
## 29 5.788688 2.71418112
## 30 4.259518 0.15992700
```

```
#Computation of the mean for each variable of the dataset for group B
mean.variables.groupB <- apply(data.B, 2, mean, na.rm=T)
mean.variables.groupB
```

```
##   scoreX   scoreY
## 5.148834 2.197438
```

We can see slightly different values for both variables in both groups. However, the mean gives us only information about the central values for both variables. It is important to introduce a measure of variation in order to determine how values are distributed *around* the mean. To do so, we use the standard deviation.

The standard deviation represents the amount of variation of values around their mean and can be computed as:

$\sigma = \sqrt(\frac{\sum_i^N (x_i - \overline{x})^2}{N})$

We will now compute the standard deviation for both variables in each group :

```
#Computation of the standard deviation for both variables
#of the dataset in group A
sd.variables.groupA <- apply(data.A, 2, sd, na.rm=T)
sd.variables.groupA
```

```
##   scoreX   scoreY
## 1.370947 1.331300
```

```
#Computation of the standard deviation for both variables
#of the dataset in group B
sd.variables.groupB <- apply(data.B, 2, sd, na.rm=T)
sd.variables.groupB
```

```
##  scoreX  scoreY
## 1.14792 1.61810
```

A large standard deviation indicates that one or several observed values are very different from the mean, while a small standard deviation shows that most values are very close to the mean.

These two measures are very useful to describe a distribution and are very important in statistical testing, as we will see later in the course. However, mean and standard deviation are not perfect to describe all distributions.

Indeed, one weakness of the mean is the lack of robustness to extreme values. The mean can be strongly influenced by the presence of a proportion of extremely large or small values. It is, therefore, not the best summary measure for skewed distributions. In such case, quantiles are preferred.

The median is the quantile of 50%, so it is the value that splits observations into two equal parts. By definition, it is more robust to extreme values: its value is not influenced directly by the extreme values. For example, the median of

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

is 5.5, and is the same as the median of

```
c(1:9, 100)
```

```
##  [1]   1   2   3   4   5   6   7   8   9 100
```

which is 5.5. So, replacing the largest value by an even larger value does not affect the median.

However, the mean is clearly affected: for the first set of values the mean is 5.5, and for the second it is 14.5.

We can compute easily the median in R for multiple columns at once:

```
#computation of the median for both variables in group A
median.variables.groupA <- apply(data.A, 2, median, na.rm = T)
median.variables.groupA
```

```
##   scoreX   scoreY
## 4.926336 1.596064
```

```
#computation of the median for both variables in group B
median.variables.groupB <- apply(data.B, 2, median, na.rm = T)
median.variables.groupB
```

```
##   scoreX   scoreY
## 5.060254 1.653759
```

For both variables, the medians per group are close to each other. This is particularly the case for `scoreY`, which was identified as skewed when plotting the histograms. Due to larger values in group B arising at random, their mean is larger than that for group A. However, the set of values in group A and group B were obtained from the same distribution, so these differences are due to random noise. The median is much less sensitive to such noise.

It can also be of interest to look at the other quantiles of the set of values. Let's take the variable `ScoreY` as an example:

```
#computation of quantile of the variable scoreY for group A
quantile(data.A$scoreY)
```

```
##         0%        25%        50%        75%       100% 
## 0.06849886 0.48053515 1.59606376 2.63762448 4.49709044
```

```
#computation of quantile of the variable scoreY for group B
quantile(data.B$scoreY)
```

```
##         0%        25%        50%        75%       100% 
## 0.05282527 0.77458628 1.65375910 3.34040589 5.97353896
```

By default the function `quantile` provides the minimum, the maximum and the quartiles of a set of values. In this example we can see that the difference between quantiles in these groups is small until the median. The third quartile and the maximum values explain the difference obtained in the mean for the variable `scoreY` between group A and group B.

We can also measure the amount of variation using robust meaures. One such measure is the median absolute variation (MAD). It is obtained by calculating the median value of the distance to the median, as described by the formula below:

$MAD = \text{median}(|X - \widetilde{X}|)$

```
#computation of the median absolute deviation for group A
mad.variables.groupA <- apply(data.A, 2, mad, na.rm =T)
mad.variables.groupA
```

```
##   scoreX   scoreY 
## 1.134796 1.626960
```

```
#computation of the median absolute deviation for group A
mad.variables.groupB <- apply(data.B, 2, mad, na.rm =T)
mad.variables.groupB
```

```
##   scoreX   scoreY 
## 1.282703 1.723980
```

To see that this is a more robust measure, let us compute it for:

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

yielding 3.7065, which is the same as the MAD of

```
c(1:9, 100)
```

```
##  [1]   1   2   3   4   5   6   7   8   9 100
```

given by 3.7065.

Quick task(s):

Solve the task(s), and check your solution(s) here.

1.4 Correlation and heatmap

1.4.1 Motivation

A next step when analysing data is to ask whether observed variables are related or independent. How can we quantify association? The measure of relatedness between variables is called **correlation**. Several types of correlation exist and they do not measure exactly the same thing. This is what we are going to explore in this section.

1.4.2 Working example

In the previous section, two unrelated variables were simulated. In this example we will create a dataset by simulating 10 more or less correlated variables.

```
library(mvtnorm)
set.seed(352)
sample.size <- 60

corrs <- runif(45, -1, 1)
corrs.matrix <- matrix(0, 10, 10)
corrs.matrix[upper.tri(corrs.matrix, diag=FALSE)] <- corrs
corrs.matrix[lower.tri(corrs.matrix, diag=FALSE)]  <- t(corrs.matrix)[lower.tri(corrs.matrix)]
diag(corrs.matrix) <- 1
standard.deviations <- rep(1,10)
covs.matrix<- as.matrix(Matrix::nearPD(diag(standard.deviations) %*% corrs.matrix %*% diag(standa

dataset<-as.data.frame(rmvnorm( sample.size, mean=rep(0,10), covs.matrix))
colnames(dataset) <- paste('Variable', c(1:10), sep='')
```

By definition, the correlation is a measure of association between two variables. It ranges from -1 to 1, is symmetric and scale-invariant. Being symmetric in this case means that the correlation between x and y is the same as the one between y and x. Scale-invariant means that multiplying a set of values of a variable by a positive number (except 0) will not impact the correlation between this variable with any other variable.

Let us look at the following variables:

```
x <- rnorm(5)
x
```

```
## [1]  1.28821371 -1.01602894 -0.27022292  0.02957032 -0.35486645
```

```
y <- rnorm(5)
y
```

```
## [1]  0.6278424  0.3978634 -0.8106308 -0.1570483  0.1107076
```

Their correlation is computed here to illustrate symmetry and scale-invariance:

```
cor(x, y)
```

```
## [1] 0.3129188
```

```
cor(y, x)
```

```
## [1] 0.3129188
```

```
cor(x, 10*y)
```

```
## [1] 0.3129188
```

The values -1 and 1 represent perfect (anti) correlation, while 0 represents absence of correlation. Having a negative correlation between two variables means that large values of one variable are associated with small values of the other. These properties are similar for all types of correlation.

Three main correlations are used. The most common one is the **Pearson** correlation, also called linear correlation, which can be computed via the formula:

$$r = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sum_i (x_i - \overline{x})^2 \sum_i (y_i - \overline{y})^2}$$

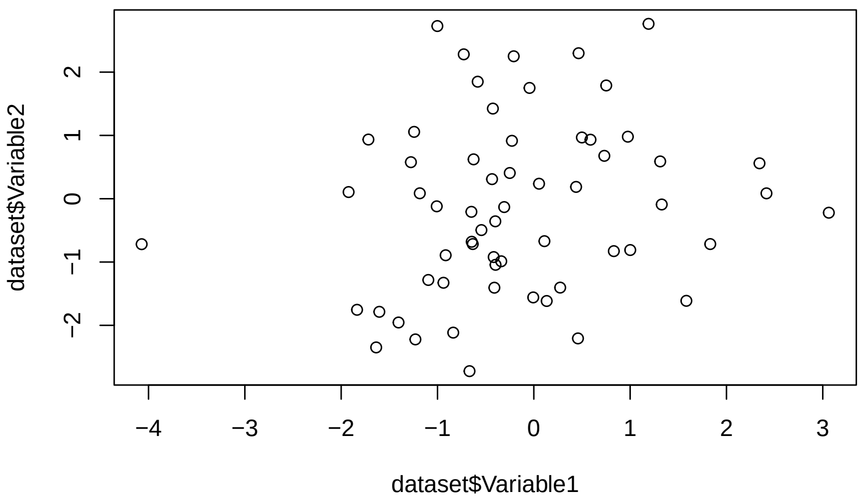
The Pearson correlation is computed by default by the R function `cor`:

```
#computation of the pearson correlation between Variable1 and Variable2
cor(dataset$Variable1,dataset$Variable2, method = 'pearson', use = 'complete.obs')
```

```
## [1] 0.1912684
```

The absolute value of the correlation 0.1912684 tells us that variation of `Variable1` does not necessarily mean variation for `Variable2`. Indeed, a scatterplot of the two variables does not suggest much association:

```
plot(dataset$Variable1, dataset$Variable2)
```



The function `cor` can also be used on the entire data matrix, yielding all pairwise correlations between variables:

```
#computation of the Pearson correlation matrix between all #variables of the dataset
correlation.pearson <- cor(dataset, method = 'pearson', use = 'complete.obs')
correlation.pearson
```

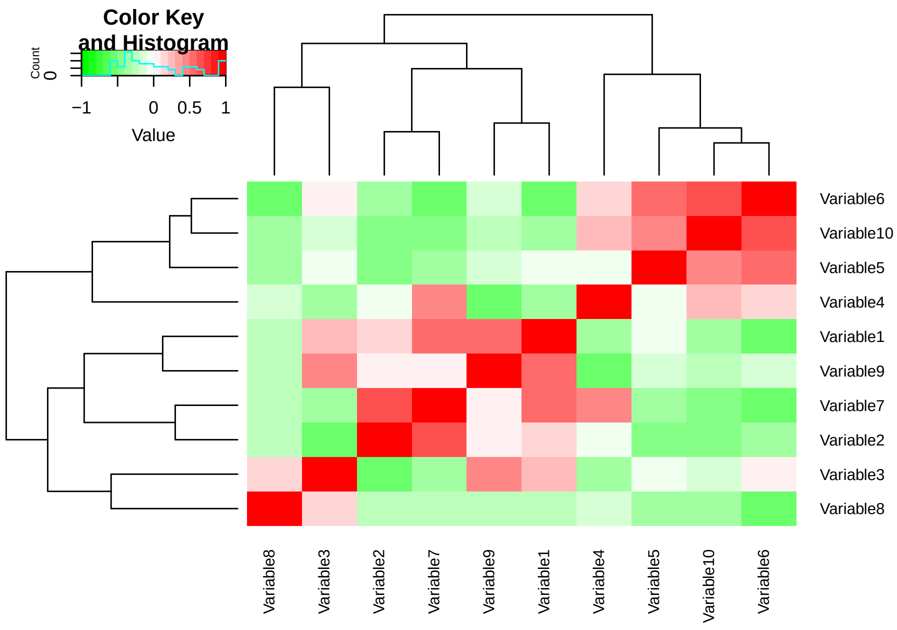
```
##              Variable1   Variable2   Variable3   Variable4   Variable5
## Variable1   1.00000000  0.19126844  0.21752224 -0.33142355 -0.03093511
## Variable2   0.19126844  1.00000000 -0.51765078 -0.01337544 -0.49872569
## Variable3   0.21752224 -0.51765078  1.00000000 -0.30235531 -0.07174296
## Variable4  -0.33142355 -0.01337544 -0.30235531  1.00000000 -0.03809821
## Variable5  -0.03093511 -0.49872569 -0.07174296 -0.03809821  1.00000000
## Variable6  -0.53181299 -0.35595914  0.04109194  0.14018466  0.58563977
## Variable7   0.50523610  0.63096198 -0.32476449  0.43999575 -0.33488644
## Variable8  -0.24407265 -0.28430370  0.11700066 -0.14327152 -0.33167843
## Variable9   0.54567571  0.09542881  0.45464382 -0.58902176 -0.17758377
## Variable10 -0.32526502 -0.47960646 -0.13381373  0.25007363  0.45375321
##              Variable6   Variable7  Variable8   Variable9 Variable10
## Variable1  -0.53181299  0.50523610 -0.2440727  0.54567571 -0.3252650
## Variable2  -0.35595914  0.63096198 -0.2843037  0.09542881 -0.4796065
## Variable3   0.04109194 -0.32476449  0.1170007  0.45464382 -0.1338137
## Variable4   0.14018466  0.43999575 -0.1432715 -0.58902176  0.2500736
## Variable5   0.58563977 -0.33488644 -0.3316784 -0.17758377  0.4537532
## Variable6   1.00000000 -0.55376001 -0.5352480 -0.10179077  0.6954481
## Variable7  -0.55376001  1.00000000 -0.2478635  0.02078802 -0.4821684
## Variable8  -0.53524803 -0.24786347  1.0000000 -0.26898602 -0.3316227
## Variable9  -0.10179077  0.02078802 -0.2689860  1.00000000 -0.2663179
## Variable10  0.69544806 -0.48216841 -0.3316227 -0.26631788  1.0000000
```

A matrix of correlation as we just obtained can provide a lot of information about the variables. However, for illustration in a paper it is better to use a representation of the correlation matrix called `heatmap`:

```
#Heatmap plot
library(gplots)
```

```
##
## Attachement du package : 'gplots'

## L'objet suivant est masqué depuis 'package:stats':
##
##     lowess
```

```
heatmap.2(correlation.pearson, trace = 'none', cexRow = 0.8,
          cexCol = 0.8, col= colorRampPalette(c("green","white","red"))(20))
```



Here, the correlation is represented by a ‘heat’, blue or ‘cold heat’ represent negative correlations, red or ‘hot heat’ represent strong correlations.

The trees produced by the function `heatmap` on each side of the plot represent the similarity between the variables based on their correlation matrix. Each variable has a root, and all roots merge at the base of the tree. The height at which the roots of two variables merge represents the dissimilarity between them.

The higher they meet, the more dissimilar they are.

The Pearson correlation is particularly good at quantifying association given a linear relationship between two variables X and Y, i.e., in cases where $ X = aY + b $. However, more complex relationship can exist between variables. Alternative correlation measures can be used to capture other forms of association. However, the interpretation of this correlation can be challenging.

One common alternative is to use **Spearman**'s correlation coefficient. It is non-parametric and often denoted by ρ. The computation is similar to the Pearson correlation, but the values are replaced by their rank. For example, to compute the Spearman correlation coefficient between

```
x <- c(7, 2, 5)
```

and

```
y <- c(1, 5, 4)
```

we actually compute the Pearson correlation between the ranks of x:

```
rank(x)
```

```
## [1] 3 1 2
```

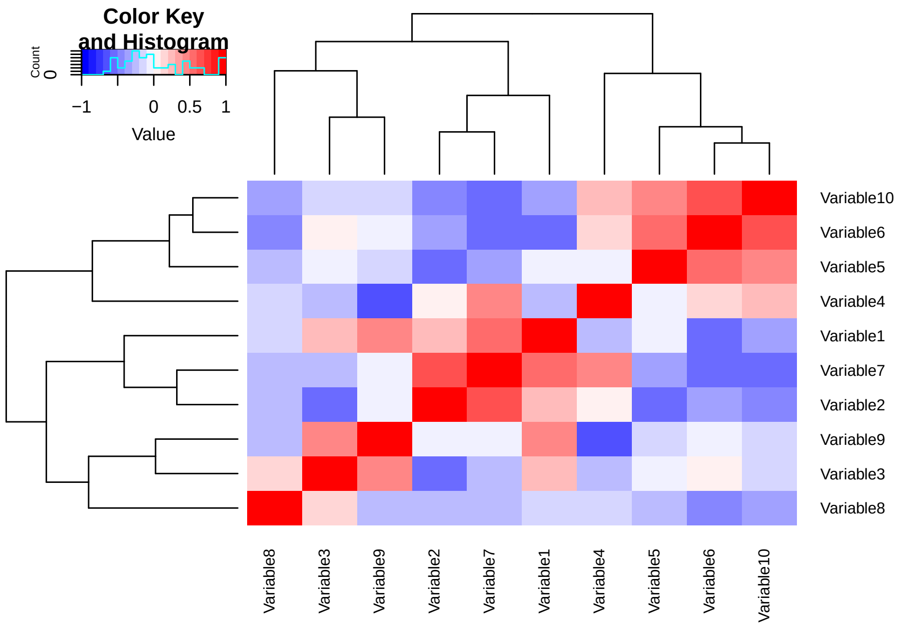
and the ranks of y:

```
rank(y)
```

```
## [1] 1 3 2
```

Let us now compute the Spearman's rank correlation matrix for `dataset` and plot the `heatmap`.

```
#computation of the correlation matrix between all variables of
correlation.spearman <- cor(dataset, method = 'spearman', use = 'complete.obs')
heatmap.2(correlation.spearman, trace = 'none', cexRow = 0.8,
          cexCol = 0.8, col= colorRampPalette(c("blue","white","red"))(20))
```



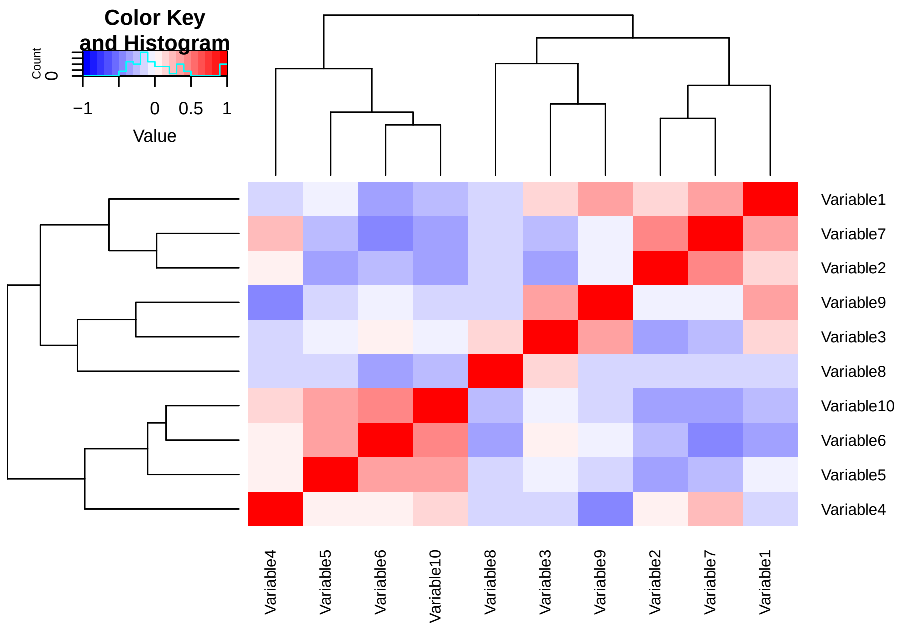
Here as I have only created linearly correlated variables, not much difference can be seen between both heatmaps. Note that a transformation of the data that does not change the ranks of observations, such as the logarithm or the square root, does not affect the Spearman correlation.

The third correlation coefficient, **Kendall**'s tau represented by τ, is also non-parametric. Let us consider 2 variables X and Y having 3 observations, (0, 1, 2) and (1, 4, 3) respectively. The 3 subjects have each 2 values, one for each variable, given by (0, 1) for the first, (1, 4) for the second, and (2, 3) for the third. Kendall's tau will check if each pair of subjects has concordant or discordant observations, and count the number of subjects of each type. In the above example, the second subject (1, 4) involves both entries larger than the first subject (0, 1), so they are called concordant. The third subject (2, 3) involves one entry larger and one smaller than the second subject (1, 4), so they are called discordant. Finally, the third subject (2, 3) is concordant compared with the first (0, 1).

The correlation is then computed as the number of concordant pairs of subjects minus the number of discordant pairs, divided by the total number of pairs. In the above example, there are 2 concordant pairs and one discordant, and the total number of pairs of subjects (combinations of subjects) is equal to 3, leading to a Kendall's τ of 1/3=0.333.

This coefficient can be more interpretable than the Spearman's correlation coefficient.

```
#computation of the correlation matrix between all variables of
correlation.kendall <- cor(dataset, method = 'kendall', use = 'complete.obs')
heatmap.2(correlation.kendall, trace = 'none', cexRow = 0.8,
          cexCol = 0.8, col= colorRampPalette(c("blue","white","red"))(20))
```



Quick task(s):

Solve the task(s), and check your solution(s) here.

Chapter 2

Statistical tests - Part 1

2.1 Comparing 2 groups

2.1.1 Motivation

We consider again the `decathlon2` dataset. After exploring the dataset and understanding its structure, we want to test the performances between the two different events, i.e. the Decastar and the OlympicGames.

```
library(factoextra)

data(decathlon2)
```

When comparing the distribution of a of values from a continuous variable of two groups we commonly compare the mean of the set of values obtained in both groups. Sample means could look different but are the samples really drawn from 2 different distributions?

2.1.2 Working Example

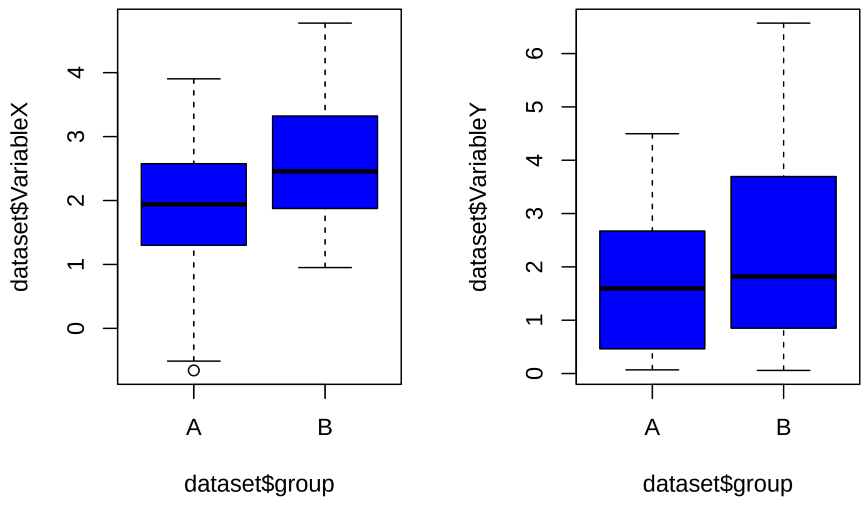
Let us consider a dataset containing two continuous variable that we will call `VariableX` and `VariableY` and a categorical variables indicating which groups samples are from.

```
set.seed(352)
sample.size<-30

variableX.A <- rnorm(sample.size, mean = 2, sd = 1)
variableX.B <- rnorm(sample.size, mean = 2.4, sd = 1.1)

variableY.A <- rgamma(sample.size, shape = 2, scale = 1)
variableY.B <- rgamma(sample.size, shape = 2, scale = 1.1)
```

```
group <- rep(c('A', 'B'), each = sample.size)

dataset <- data.frame(VariableX = c(variableX.A, variableX.B),
                      VariableY = c(variableY.A, variableY.B),
                      group = group, stringsAsFactors = T)
```

We start by plotting the boxplots for each group of both variables.

```
#Boxplots for variable X and Y
par(mfrow=c(1,2))
boxplot(dataset$VariableX ~ dataset$group, col = 'blue')
boxplot(dataset$VariableY ~ dataset$group, col = 'blue')
```



We can identify by 'eye' differences between the distribution of both group but how can we quantify the differences if there is any? To do so we will use statistical tests.

The most common test to compare two groups is the Student's t test, this test as the same name as the Student's t distribution as it utilizes the same distribution. Let's assume that two sets of values are drawn from distributions with expectancies μ_1 and μ_2, and variances σ_1^2 and σ_2^2, respectively. The Student's test statistic is computed as follows:

$t = \frac{\overline{x_1} - \overline{x_2}}{SEDM}$

where the *standard error of difference of means* is:

$SEDM = \sqrt{SEM_1^2 + SEM_2^2}$

The square root of the sum of the squared standard errors to mean of each group. The standard error to the mean can itself be computed as the standard deviation divided by the square root of the number of samples.

$SEM_1 = \frac{\sigma_1}{\sqrt{n}}$

When using a statistical test two hypotheses are formulated, the null hypothesis H_0 and the alternative hypothesis H_a, respectively. In the case of the Student's t-test, the hypotheses are:

$H_0 : \mu_1 = \mu_2$ and $H_a : \mu_1 \neq \mu_2$

Note that there can be 3 different alternative hypothesis depending on what we want to test. The test for alternative hypothesis presented above is called the *two sided* Student's t test as we are testing for different means but we are not making any assumptions on which direction the differences is. Two other alternative can be tested. Namely, $H_a = \mu_1 < \mu_2$, and $H_a = \mu_1 > \mu_2$. Under the null hypothesis the variable t follows a Student's distribution of degrees of freedom equal to $n1 + n2 - 2$ if the variances σ_1^2 and σ_2^2 are equal. If this is not the case the statistics t can still be approximated as a Student's t distribution with a number of degrees of freedom given by the Welch procedure that we won't detail here.

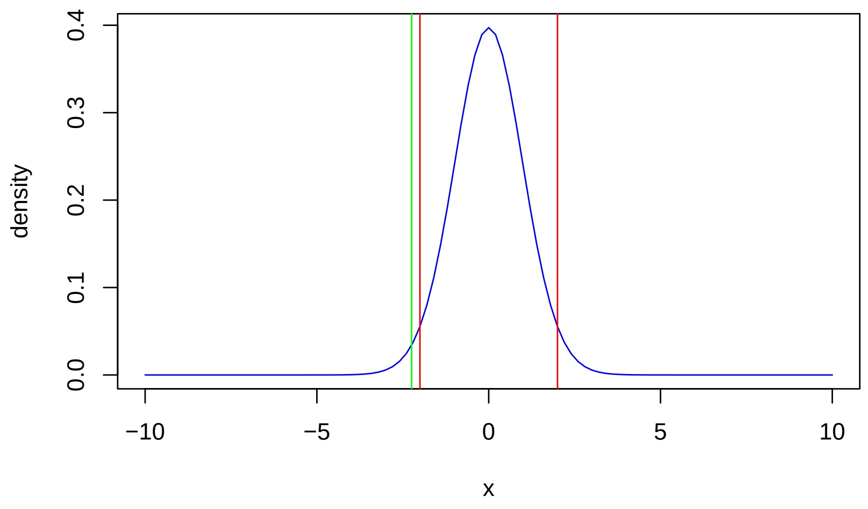
Let's apply the Student's t test to our dataset:

```r
#t-test for VariableX
test.variableX <- t.test(dataset$VariableX ~ dataset$group)
test.variableX
```

```
##
##  Welch Two Sample t-test
##
## data:  dataset$VariableX by dataset$group
## t = -2.2435, df = 57.612, p-value = 0.02872
## alternative hypothesis: true difference in means between group A and group B is not equal to
## 95 percent confidence interval:
##  -1.20392326 -0.06848317
## sample estimates:
## mean in group A mean in group B
##        1.900228        2.536431
```

Let's look at the results obtained. We can see that the estimated means are equal to 1.9002279 and 2.5364311 in group A and group B respectively. This lead to an estimated difference in mean of -0.6362032. The computed test statistic is equal to -2.2435031. The number of degrees of freedom estimated is equal to 57.6120941. We can then plot the density distribution of this Student's t random variable.

```
curve(dt(x, df = test.variableX$parameter), col = 'blue', from = -10,
      to = 10, ylab = 'density')
abline(v = qt(c(0.025, 0.975), df = test.variableX$parameter),
       col = 'red')
abline(v = test.variableX$statistic, col= 'green')
```



As we can see the value of the statistics is outside the range delimited by the 2.5% quantile and the 97.5% quantile. Therefore it is unlikely that the statistics is drawn from such random variable. It allows us to reject the null hypothesis and accept the alternative hypothesis with less than 5% chance risk of making a mistake. The p-value obtained from the test is the probability that the statistics is drawn from the hypothetical Student's t distribution under the null hypothesis from quantiles table for example.

An important output information from the output is also the 95% confidence interval. This interval quantifies the uncertainty of the estimate of the mean difference. The 95% confidence interval is a range of values that you can be 95% certain contains the true mean difference between the two populations. As we can see 0 is not in this range allowing us to, once again, reject the null hypothesis of equality of the means. The confidence interval can be computed by using the SEDM and the mean difference obtained in the test. For a level of confidence α of a two sided Student test is give by the formula:

$highBoundary = mean + t_{1-\frac{\alpha}{2}} * SEDM \; lowBoundary = mean - t_{1-\frac{\alpha}{2}} * SEDM$

We can see from this formula that two factors can influence the size of the confidence interval. Indeed, the SEDM is a function depending on the standard deviation and the number of samples in each group. On one hand, increasing the number of sample will improve the accuracy of the test and narrow the confidence interval. On the other hand, samples with large standard deviation will lead to larger confidence intervals. We can compute the confidence intervals of the test we made earlier:

```
low.boundary <- test.variableX$estimate[1] - test.variableX$estimate[2] -
  qt(0.975, df = test.variableX$parameter)*test.variableX$stderr
high.boundary <- test.variableX$estimate[1] - test.variableX$estimate[2] +
  qt(0.975, df = test.variableX$parameter)*test.variableX$stderr
c(low.boundary, high.boundary)
```

```
## mean in group A mean in group A
##     -1.20392326     -0.06848317
```

The Student's t test is a great tool to compare two distributions. However, it has its weaknesses. In case of skewed data, the standard deviation of both compared samples can be relatively different and impact the power of the test to detect relevant differences. In such cases, it is preferable to transform the data using for example a log transformation. For the Student's t test it is requested that the distributions of the samples values are following a normal distribution or to have at least a sample size of 30 in each group (central limit theorem). If this is not the case, it is better to use a non parametric test called Mann-Whitney-Wilcoxon test (or Mann-Whitney U test or Wilcoxon sum rank test).

The Mann-Whitney-Wilcoxon test is non parametric as it does not test a specific parameter to differentiate two populations. Instead we will test if the probability that an observation in group A is superior to an observation from group B is different from the probability that an observation from group B is superior from group A. The main assumption of the Mann-Whitney-Wilcoxon test is that the samples from both groups must be independent so it cannot be used for paired observations. For paired observations we would prefer the Wilcoxon signed-rank test that we won't detail here. If the observations in group A are drawn from a random variable X and the observation in group B are coming from a random variable Y the null and alternative hypotheses are:

$H_0 : P(X > Y) = P(Y > X)$ and $H_a : P(X > Y) \neq P(Y > X)$

To simplify the test relies on comparing the medians of both populations. The statistic of the test called U is computed as follows. All observations are pooled together and ranked based on their values. For example if we have a pooled set of 6 distinct values (2, 4, 4, 4, 1, 5), the ranked version of this set of obervations becomes (2, 4, 4, 4, 6, 1). Let's call R_1. We can then compute U as:

U $= R_1 - \frac{(n_1)(n_1+1)}{2}$

With this statistics U we can then compute the p-value using either a Wilcoxon Rank-Sum Table in case of small samples and approximating U as a normal distribution for large samples:

```
wilcox.test(dataset$VariableY ~ dataset$group, alternative = "two.sided")
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  dataset$VariableY by dataset$group
## W = 358, p-value = 0.1774
## alternative hypothesis: true location shift is not equal to 0
```

We can notice that the output of the Mann-Whitney-Wilcoxon test is smaller than for the Student's t test. Indeed as it is a non parametric test it does not provide direct effect size or confidence interval. However, relying only on the p-value is not correct and one would prefer to report also the difference between the median of both samples.

Quick task(s):

Solve the task(s), and check your solution(s) here.

2.2 Power of a statistical test

2.2.1 Motivation

After looking at the results of the group comparison using the `decathlon2` dataset the searcher wants to build another study in order to prove that the impact of the material used to build the surface of the stadium can impact the performance of the racing events and the long jump event. While building the protocol of his study the searcher encounters the question : How many sportsmen should he recruit in order to obtain satisfactory testing results?

```
library(factoextra)

data(decathlon2)
```

2.2.2 Working example

To answer this question we need to introduce an important notion in statistical modeling. The power of a statistical test. Let's simulate two variable for 2 different group. To do so we will have a look at two tests:

```
set.seed(352)
#Creation dataset
variableX.A <- rnorm(10, mean = 0, sd =1)
```

```r
variableX.B <- rnorm(10, mean = 2, sd =1)
variableY.A <- rnorm(10, mean = 0, sd =1)
variableY.B <- rnorm(10, mean = 0.3, sd =1)
dataset <- data.frame(VariableX = c(variableX.A, variableX.B),
                      VariableY = c(variableY.A, variableY.B),
                      group = c(rep('A', 10), rep('B',10)))
```

We created both Variables to have a different means depending on the group. Now let's test these differences using the Student's t test.

```r
t.test(VariableX ~ group, data = dataset)
```

```
##
##  Welch Two Sample t-test
##
## data:  VariableX by group
## t = -2.9165, df = 15.489, p-value = 0.01036
## alternative hypothesis: true difference in means between group A and group B is not equal to 
## 95 percent confidence interval:
##  -2.8904084 -0.4533992
## sample estimates:
## mean in group A mean in group B
##      -0.0255522       1.6463516
```

```r
t.test(VariableY ~ group, data = dataset)
```

```
##
##  Welch Two Sample t-test
##
## data:  VariableY by group
## t = -0.55427, df = 17.078, p-value = 0.5866
## alternative hypothesis: true difference in means between group A and group B is not equal to 
## 95 percent confidence interval:
##  -1.220411  0.712452
## sample estimates:
## mean in group A mean in group B
##      0.07988438      0.33386405
```

We can see that even if we simulated both variables from different distributions depending on the group. We ended up with one test detecting significant differences between the groups and the other one not detecting differences. Typically, there is 4 different possible outcomes from a statistical test::

Reality Test results	Non-significant	Significant
H_0 True	$1-\beta$ (True Negative)	α (False Positive)
H_0 False	β (False Negative)	$1-\alpha$ (True Positive)

The α represents they type I error, i.e. the probability of finding a significant difference when there is none. This coefficient α is often fixed to 5%. The β coefficient represent the probability to obtain a non-significant test while the alternative hypothesis is true and is called type II error. This is the case when we test the VariableY that we simulated. The power of a statistical test correspond to the values 1-β, and is the probability to reject the null hypothesis appropriately.

The power of a statistical test depends on 4 different parameters, the size of the effect (in the example the effect size is small), the sample size, the variance of the sample, and the significance level of the test α.

If the effect size of the phenomenon we want to study cannot be easily changed, we can influence the three others to obtain a good power:

- Increasing the sample size
- Decreasing the variability of the samples
- Increasing the level *alpha* , i.e. increasing the chance of false positives

As the latter is not advisable in most studies, the easiest approach to influence the power of the test is to increase the sample size. Let's compute the power to reject the null hypothesis in our example. To do so we can for example use the function `pwr.t.test` or power from the R package *pwr*:

```r
library(pwr)
#Power computation VariableX
pwr.t.test(n = 10, d = 2/1, sig.level = 0.05, power = NULL,
           type = c("two.sample"), alternative = c("two.sided"))
```

```
##
##      Two-sample t test power calculation
##
##               n = 10
##               d = 2
##       sig.level = 0.05
##           power = 0.988179
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```r
#Power computation VariableY
pwr.t.test(n = 10, d = 0.5/1, sig.level = 0.05, power = NULL,
           type = c("two.sample"), alternative = c("two.sided"))
```

```
##
##      Two-sample t test power calculation
##
##               n = 10
##               d = 0.5
##       sig.level = 0.05
##           power = 0.1850957
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

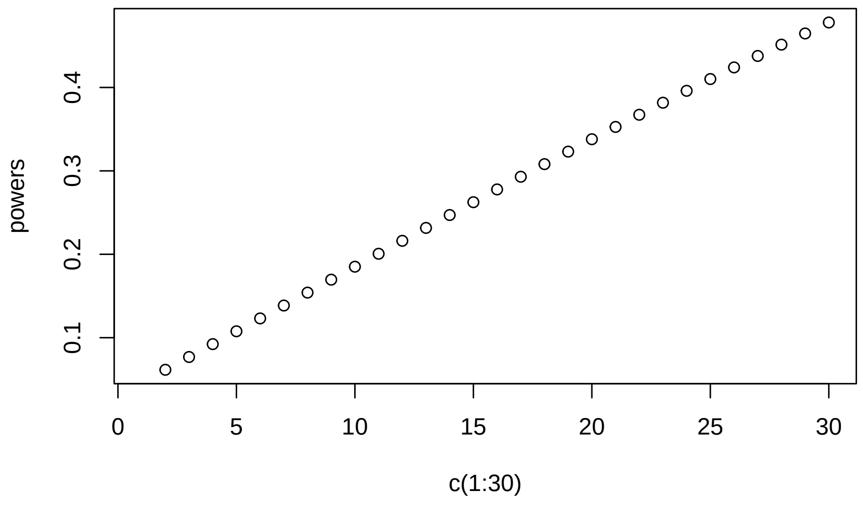
We can see that the power to reject the null hypothesis for VariableX is 98.8% but for VariableY it is only 18.5%. As you can see the relation between the power and the effect size is not linear as the effect size is 4 times larger (0.5 to 2) but the power to reject H_0 is more than 5 times larger for VariableX. For simplicity we will not present, here, the exact formula to compute the power as it involves more statistical knowledge than the aim of this course.

We can plot the evolution of the power as a function of the effect size:

```
#computation of the power for different sample size
powers <- pwr.t.test(n = c(1:30), d = 0.5/1, sig.level = 0.05, power = NULL,
                     type = c("two.sample"), alternative = c("two.sided"))$power
```

```
## Warning in qt(sig.level/tside, nu, lower = FALSE): Production de NaN
```

```
#Plot of the power as a function of the sample size

plot(c(1:30), powers)
```



We see that even with a sample size of 30 the power to detect the difference between the two groups is only 0.4778965. Instead we can compute what would be the sample size needed for a given a power. The common desired power requested in studies are 80% and 90%. We can compute the sample size needed for both power for the VariableY

```
#computation of sample size for a fixed power of 80%
test.80.percent <- pwr.t.test(n = NULL, d = 0.5/1, sig.level = 0.05, power = 0.8,
                              type = c("two.sample"),alternative = c("two.sided"))
test.80.percent
```

```
##
##      Two-sample t test power calculation
##
##               n = 63.76561
##               d = 0.5
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```
#computation of sample size for a fixed power Of 90%
test.90.percent <- pwr.t.test(n = NULL, d = 0.5/1, sig.level = 0.05, power = 0.9,
                              type = c("two.sample"), alternative = c("two.sided"))
test.90.percent
```

```
##
##      Two-sample t test power calculation
##
##               n = 85.03128
##               d = 0.5
##       sig.level = 0.05
##           power = 0.9
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

We can see that we would need 64 samples in each group for a power of 80% and 85 for a power of 90%.

We have seen that in order to determine the proper sample size needed in a study we need to know or to give an estimate of the effect sizes we want to test, the standard deviation existing in the sample and to fix both type I error and the power to detect a true positive. To do so the approach often used is to go back to the literature or to used pilot studies.

Note that for the Student's t test obtaining power and sample sizes is relatively simple due to the exact mathematical formula linking all 4 parameters. But for a lot of test (especially non parametric ones) there are no exact formulas and simulations are needed. Hopefully, there are a lot of function in R that are able to do the simulation and power computations.

Quick task(s):

Solve the task(s), and check your solution(s) here.

Chapter 3

Common regression models

3.1 Linear regression

3.1.1 Motivation

We have seen in the previous chapter how to compare the distribution between two groups. This gives us an understanding of the relationship between a continuous variable and a binary variable, i.e. a discrete variable with only two possible values. In order to test the association between a continuous variables we need to another approach. For example, does your ability in sprint can influence your pole vault or long jump performances? To answer this kind of question, the best approach is to use linear regression.

```
library(factoextra)

data(decathlon2)
```

3.1.2 Working Example

Let's use the simulated dataset we used in the previous chapter and add one variable `VariableZ` that we will build using the 2 other variables.

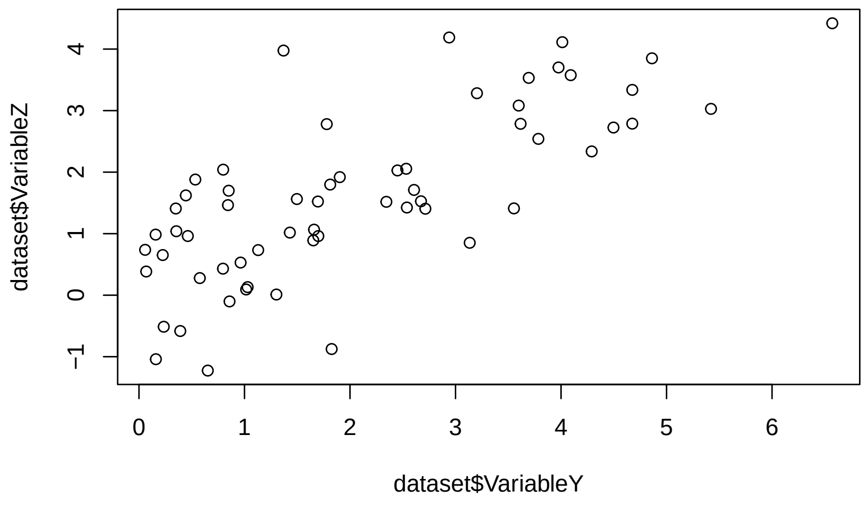
```
set.seed(352)
sample.size<-30

variableX.groupA <- rnorm(sample.size, mean = 2, sd = 1)
variableX.groupB <- rnorm(sample.size, mean = 2.4, sd = 1)
VariableX <- c(variableX.groupA, variableX.groupB)

variableY.groupA <- rgamma(sample.size, shape = 2, scale = 1)
variableY.groupB <- rgamma(sample.size, shape = 2, scale = 1.1)
```

```r
VariableY <- c(variableY.groupA, variableY.groupB)

VariableZ <- runif(1, min = 0, max = 1)*VariableX +
  runif(1, min = 0, max = 1)*VariableY +
  rnorm(2*sample.size, mean = 0, sd = 1)

group <- rep(c('A', 'B'), each = sample.size)

dataset <- data.frame(VariableX = VariableX, VariableY = VariableY,
                      VariableZ = VariableZ,
                      group = group, stringsAsFactors = T)
```

What is a linear regression analysis?

Let's make a scatterplot of `VariableZ` and `VariableY`.

```r
plot(dataset$VariableY, dataset$VariableZ)
```



We can clearly see that as we look at higher values of the `VariableY` the values of `VariableZ` are also increasing. Quantifying the effect of a variable x on the variation of a variable of interest y is the aim of a linear regression model. A linear regression model is written as:

$y_i = \alpha + \beta x_i + \epsilon_i$

where y is the variable of interest also called outcome and x is the variable we are testing for association with the outcome. ϵ_i are called residuals and represent the part of the outcome that could not be 'explained' by x. Each ϵ_i are supposed independent and drawn from a normal distribution $N(0, \sigma^2)$. The aim of a logistic regression analysis is to estimate the value of α, β and σ^2. To do so we use the method of least square, i.e., we find the value of α and β that minimizes the sum of squared residuals:

$SS_{res} = \sum_i (y_i - (\alpha + \beta x_i))^2$

This equation leads to exact formulas for the estimation of the parameters :

$\hat{\beta} = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sum_i (x_i - \overline{x})^2}$

$\hat{\alpha} = \overline{y} - \hat{\beta}\overline{x}$

$\sigma^2 = \frac{SS_{res}}{n-2}$

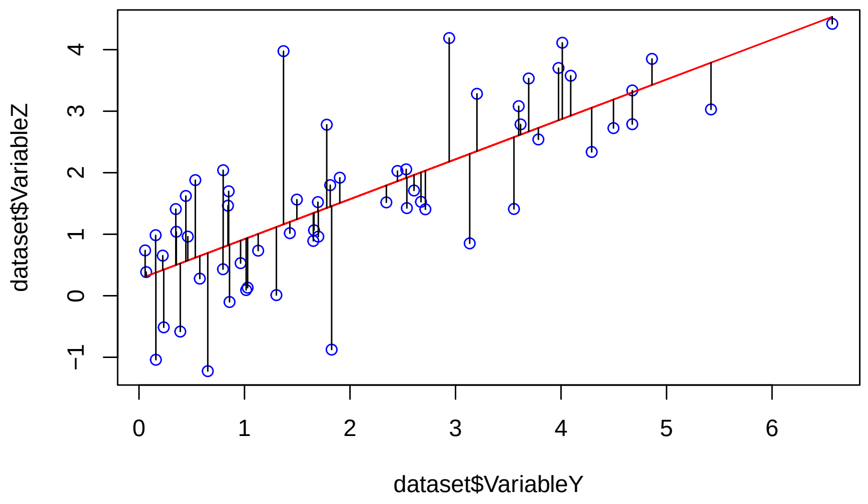
Let's do a linear regression of `VariableY` on `VariableZ` by using the function lm in R:

```
regression <- lm(VariableZ ~ VariableY, data = dataset)
regression
```

```
##
## Call:
## lm(formula = VariableZ ~ VariableY, data = dataset)
##
## Coefficients:
## (Intercept)    VariableY
##      0.2723       0.6487
```

The function lm returns 2 values, the intercept which correspond to α and the β coefficient corresponding to the value below the `VariableY`. The 4beta$ coefficent can be interpreted as if my variableX increases of 1 my variableZ will increase of $summary(regression)$coefficients[2,4]. We can represent the results onto the scatterplot:

```
plot(dataset$VariableY, dataset$VariableZ, col = 'blue')
lines(dataset$VariableY,fitted(regression), col = 'red')
segments(dataset$VariableY,fitted(regression), dataset$VariableY,dataset$VariableZ)
```



In this plot, the blue points are the observed values, the red line is the result of the regression analysis with α being the intercept of the line and β its slope. Finally the black vertical line represent the residuals, i.e. the variation that could not be explained by the linear regression model.

The interest of a linear regression is that we can test if the coefficient β is equal to 0 in order to test for association between the 2 variables. To do so in a univariate linear regression we define the statistics t as :

$t = \frac{\beta}{se(\beta)}$

where t (under the assumption that the outcome is normally distributed and under the null hypothesis $H_0 : \beta = 0$) follows a Student's t distribution with n-2 degress of freedom. With this information computing the p-value of the test is the same as for the Sutdent's t test.

For example we can look at the association between `VariableX` and `group` using both Student's t test and the linear regression:

```
#Student's t test between VariableX and group
t.test(VariableX ~ group, data = dataset)
```

```
##
##  Welch Two Sample t-test
##
## data:  VariableX by group
## t = -2.293, df = 56.263, p-value = 0.02561
## alternative hypothesis: true difference in means between group A and group B is not
## 95 percent confidence interval:
##  -1.16872012 -0.07888065
## sample estimates:
## mean in group A mean in group B
##        1.900228        2.524028
```

```
#linear regression between VariableX and group
linear.regression <- lm (VariableX ~ group, data = dataset)
summary(linear.regression)$coefficients
```

```
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 1.9002279  0.1923664 9.878170 4.879130e-14
## groupB      0.6238004  0.2720472 2.292986 2.549308e-02
```

We see that we obtain exactly the same results. As both approaches are equivalent the sample size and power computation for a linear regression model are the same than for the student's t test.

Note that there is another possible statistical test to apply is the likelihood ratio test. This test comes from a statiscal test called goodness of fit test which allows to test for a significant improvement of the model fit. By comparing the likelihoods, i.e. the probability of observing the set of values based on the statistical model fitted, of the models fitted under the null hypothesis (in our case with only an intercept) and the alternative hypothesis (the model that we fitted). This test will be seen more in detail later in the course. Asymptotically (with the number of samples sufficiently large), both likelihood ratio test and student's t test are equivalent.

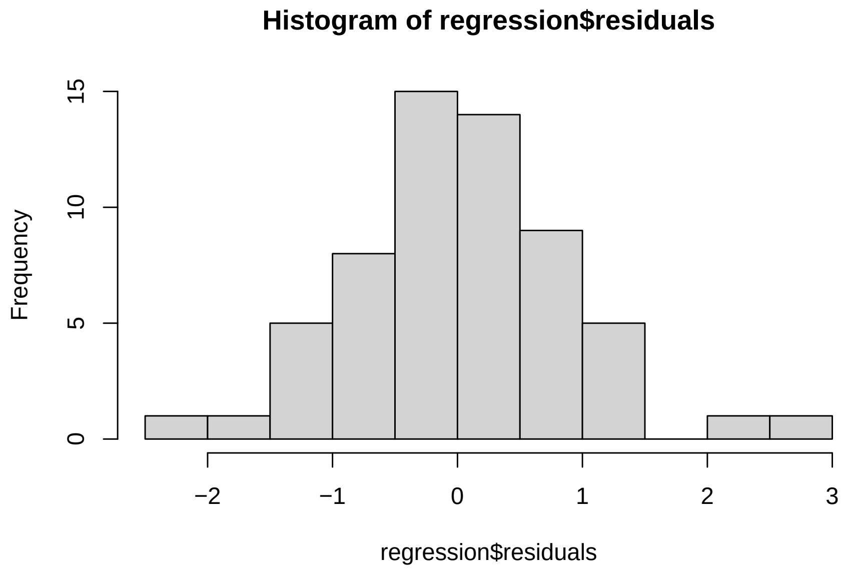
There are four assumptions associated witha linear regression model. First, the outcome has to be a normally ditributed continuous variable. The relationship between the variable x and $\overline{y}$ is linear. The observations have to be independent and the variance of the residuals have to be independent of X (homoscedasticity). To trust the results of the linear regression several steps have to be taken based on theses assumptions:

- Checking the distribution of the outcome if the distribution is too far from a gaussian distribution it can be good to apply a transformation to the data. Such as log transformation.

- Checking the distributions of the residuals as they are assumed to be normal.

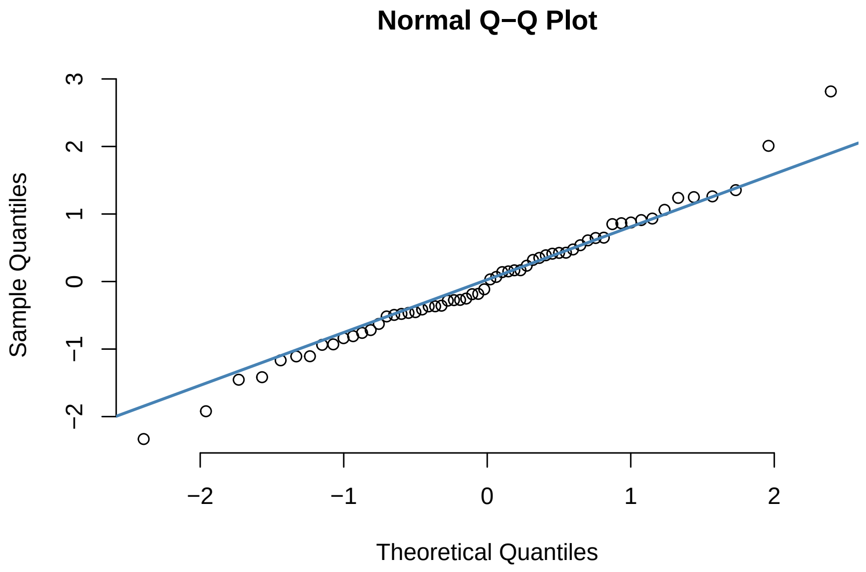
In our case we simulated the outcome `VariableZ` as the sum of two gaussian distribution so `VariableZ`. Instead let's plot the distribution of the residuals:

```
hist(regression$residuals, breaks = 10)
```



We can see clearly that the residuals distribution is really close from a normal distribution. This clearly indicates that the regression went well. Another plit is commonly used to check if a distribution is following a gaussian distribution. The Q-Q plot. This plot consist of comparing a set of values with the quantiles of a normal distribution:

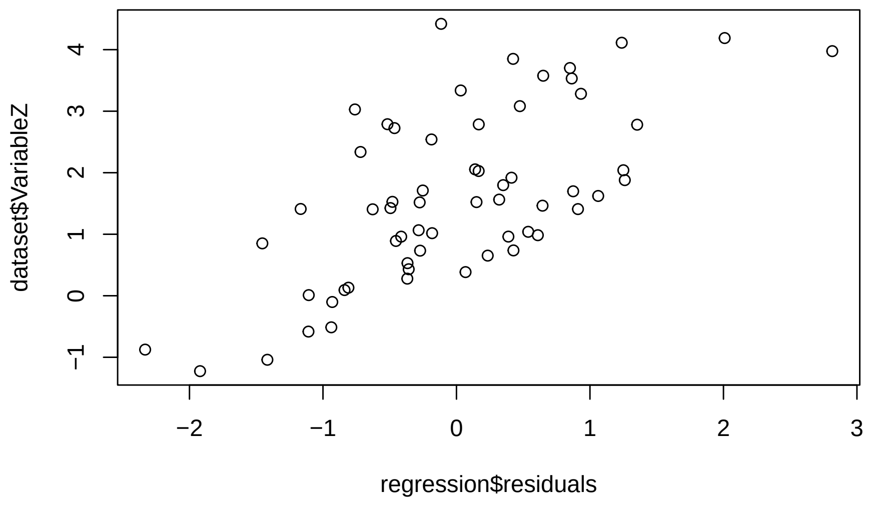
```r
qqnorm(regression$residuals, pch = 1, frame = FALSE)
qqline(regression$residuals, col = "steelblue", lwd = 2)
```



The closer the points are from the blue line the closer the set of values is close from a normal distribution.

Another interesting graph to plot is the scatter plot of the outcome and the residuals.

```
plot(regression$residuals, dataset$VariableZ)
```



We can clearly see a trend in the scatterplot. This trend is interesting, as it means that the residuals of the linear models are associated with the outcome Y. Meaning that the variation not explained by the linear model is not due to randomess and another linear association, independent to `VariableY` exists. It is easy to explain in our case as `VariableZ` was simulated as the weighted sum of `VariableX` and `VariableY`.

To obtain a better linear model it might of interest to use linear models with multiple variables such as:

$z_i = \alpha + \beta_1 y_i + \beta_2 x_i + \epsilon_i$

let's run the model using the `lm` function:

```
multiple.regression <- lm (VariableZ ~ VariableX + VariableY, data=dataset)
multiple.regression
```

```
##
## Call:
## lm(formula = VariableZ ~ VariableX + VariableY, data = dataset)
##
## Coefficients:
## (Intercept)    VariableX    VariableY
##     -0.2473       0.2658       0.6158
```

And we can check the significance of the βs obtained:

```
summary(multiple.regression)$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -0.2473115 0.28451895 -0.8692268 3.883686e-01
## VariableX    0.2658086 0.10814688  2.4578481 1.704112e-02
## VariableY    0.6158445 0.07401264  8.3208018 2.020188e-11
```

We can indeed see that both coefficents are correctly identified as significant. We can also note that the coefficent of `VariableY` is almost not impacted by the addition of `VariableX`. This is due to the fact that these variables were simulated independently and their correlation should be close to 0. After, computation we obtain a correlation of 0.1804148. With a larger correlation between both variables, the impact on the coefficient obtained on the simple linear regression would have been stronger.

Strong correlation between predictors in the same models can create the so-called multi-collinearity problem that leads to a severe increase of the p.value and therefore a reduction in significance.

Note that the power and sample size computations for linear regression with multiple predictors becomes rather challenging. Searcher often use rules of thumbs. A commonly encountered rule of thumbs is at least 10-15 samples for each predictors in the model. But this is only a rule of thumbs and as we know, the required sample size is depending on the variation of the outcome of interest as well as the effect size to test.

Quick task(s):

Solve the task(s), and check your solution(s) here.

3.2 Logistic regression

3.2.1 Motivation

In some studies the response variable only has two possible values. Some examples:

- individuals who have or have not cancer;
- cancer patients who, after treatment, relapse or not;
- from all patients receiving the same treatment, each develops resistance or not.

If analysing data with such a *binary* response variable, care must be taken, in particular when using regression models. Such a variable can be coded as $\{0, 1\}$

in R, for example. However, linear regression is not going to return predicted values as only 0 or 1. In fact, nothing prevents a linear regression of returning a negative value, or a value larger than 1.

3.2.2 Link function

In order to avoid this and model the data correctly, we need to use a regression model with a *link* function. For a regression model with a response Y and covariates X_1, X_2, this link function can be written as h, as in the model:

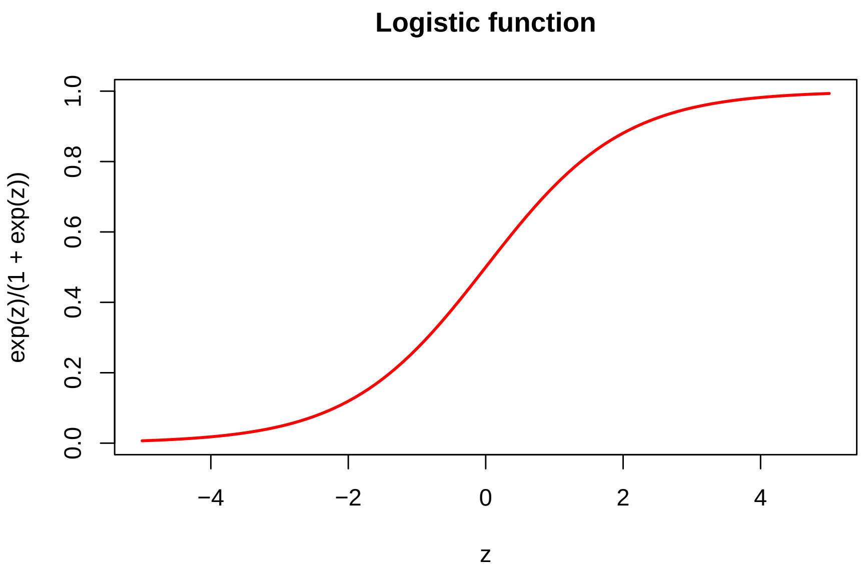
$$E(Y) = h^{-1}\left(\alpha + \beta_1 X_1 + \beta_2 X_2\right).$$

The job of the link function is to make sure that the error is modelled correctly, and it yields predicted values in the range desired. For a binary response, the ideal link function is the *logistic* function. It takes any values, positive or negative, and always return a value between 0 and 1. The logistic function has can be written as:

$$f(z) = \frac{e^z}{1+e^z},$$

for any value of z. The graph of this function for values of z between -5 and 5 is:

```
z <- seq(from = -5, to = 5, by = 0.1)
plot(z, exp(z)/(1+exp(z)), main = "Logistic function",
     type = "l", col = "red", lwd = 2)
```



The logistic link function relates the response Y and the linear prediction (which is the function of the covariates, $\alpha + \beta_1 X_1 + \beta_2 X_2$) in the following way:

$$E(Y) = \frac{e^{\alpha+\beta_1 X_1+\beta_2 X_2}}{1 + e^{\alpha+\beta_1 X_1+\beta_2 X_2}}.$$

The function can be inverted to write:

$$\log(\frac{E(Y)}{1 - E(Y)}) = \alpha + \beta_1 X_1 + \beta_2 X_2.$$

3.2.3 The `glm` function

The logistic regression model is fitted by the function `glm`. This function works in very similar ways to `lm` for linear regression, such as for example using as main input a formula determining the model to be used. It also may use `family` and `link` slots to define which probability distribution is to be used for the errors (`family`) and which link function is to be used (`link`). Per family, one link function is used by default. In the case of the logistic model, it is defined `family = binomial`, and the default link is the logistic.

The response for the logistic model can be given in different ways. We will here use the pair (number dead, number alive) per combination of dose and sex. See the exercises for an example of a binary response.

As with a linear regression, `summary` and `anova` are used to obtain an overview of the model fit and test for effects of variables.

3.2.4 Working example

This example is adapted from an example used in Venables and Ripley (1995):

Venables, W. N. and Ripley, B. D. (1995). Modern applied statistics with S-Plus. Springer-Verlag: New York.

https://www.springer.com/gp/book/9780387954578

Collet (11=991, p. 75) reports an experiment on the toxicity of the tobacco budworm *Heliothis virescens* to doses of the pyrethroid trans-cypermethrin to which the moths were beginning to show resistance. It is of interest to determine which dose level to choose, so as to guarantee a specific death proportion. The response variable is thus dead or alive per moth. Being a binary variable, a logistic regression is an ideal tool to analyse the data.

Batches of 20 months of each sex were exposed for 3 days to the pyrethroid and the number in each batch which were dead (or knocked down) was recorded. The doses are used in two-fold increases, so it is natural to consider those on the log2-scale. The table of total dead moths per dose and sex is:

sex	d=1	d=2	d=4	d=8	d=16	d=32
male	1	4	9	13	18	20
female	0	2	6	10	12	16

We enter the data into R as follows: the dose is entered in its original scale, then log2-transformed. The total numbers of death moths are entered as a single vector, with all males followed by all females. So, the dose vector needs to be repeated (stacked) to produce entries corresponding to all total death count observations. Finally, we enter the sex by repeating `male` or `female` the required number of times.

```r
dose <- c(1, 2, 4, 8, 16, 32)
ldose <- log2(dose)
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
ldose <- rep(ldose, 2)
sex <- rep(c("male", "female"), each = length(dose))
```

To define the response as the pair of (number dead, number alive) we do:

```r
resp <- cbind(numdead, numalive = 20 - numdead)
```

Now we fit the logistic model:

```r
budworm.lg <- glm(resp ~ sex + ldose, family = binomial)
summary(budworm.lg)
```

```
##
## Call:
## glm(formula = resp ~ sex + ldose, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.10540  -0.65343  -0.02225   0.48471   1.42944
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4732     0.4685  -7.413 1.23e-13 ***
## sexmale       1.1007     0.3558   3.093  0.00198 **
## ldose         1.0642     0.1311   8.119 4.70e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 124.8756  on 11  degrees of freedom
## Residual deviance:   6.7571  on  9  degrees of freedom
## AIC: 42.867
##
## Number of Fisher Scoring iterations: 4
```

There may also be an interaction effect between `sex` and `ldose`, meaning that the `ldose` effect may change depending on the sex. To check this, we add the interaction term:

```
budworm.lg.i <- update(budworm.lg, . ~ sex:ldose)
summary(budworm.lg.i)
```

```
##
## Call:
## glm(formula = resp ~ sex:ldose, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.45854  -0.29503  -0.05339   0.44888   1.06990
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.9073     0.3893  -7.468 8.12e-14 ***
## sexfemale:ldose    0.8823     0.1275   6.920 4.52e-12 ***
## sexmale:ldose      1.2893     0.1669   7.723 1.13e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 124.8756  on 11  degrees of freedom
## Residual deviance:   5.0443  on  9  degrees of freedom
## AIC: 41.155
##
## Number of Fisher Scoring iterations: 4
```

Here we fitted a new model by using the function `update`, which added the interaction effect `sex:dose` to the linear predictor already in `budworm.lg`. Note that this is the same as fitting the model by calling `glm` again with the interaction term added.

We can use `anova` to summarize results for a model, as before. Here we will test for the effect of variables in the model fit `budworm.lg`. The function `anova` will perform testing as if variables were added sequentially, one by one. In this case, the effect of `sex` is tested, subsequently the effect of `ldose` given that the `sex` effect is corrected for. Let us try this out:

```
anova(budworm.lg)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: resp
##
## Terms added sequentially (first to last)
##
##
##       Df Deviance Resid. Df Resid. Dev
## NULL                     11    124.876
## sex    1    6.077        10    118.799
## ldose  1  112.042         9      6.757
```

Note that the `anova` function now did not return a p-value for the covariate effects. When applied with generalized linear model fits, this is the default. This forces the user to choose the appropriate test to run (see `help(anova.glm)` for details). The suitable test in this case is the chi-square, so we rerun it using:

```
anova(budworm.lg, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: resp
##
## Terms added sequentially (first to last)
##
##
##       Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                     11    124.876
## sex    1    6.077        10    118.799   0.0137 *
## ldose  1  112.042         9      6.757   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In order to test for the main effects, followed by the interaction, use the entire model fit in one `glm` call:

```
anova(glm(resp ~ sex + ldose + sex:ldose, family = binomial), test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: resp
##
```

```
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                         11    124.876
## sex        1    6.077        10    118.799   0.0137 *
## ldose      1  112.042         9      6.757   <2e-16 ***
## sex:ldose  1    1.763         8      4.994   0.1842
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We conclude that both `sex` and `ldose` have statistically significant effects, but that the interaction does not.

Quick task(s):

Solve the task(s), and check your solution(s) here.

3.2.5 Fitted values

It is useful to make a graph of the predicted values. For this, we use the function `predict`. It needs as input the fitted model, in this case `budworm.lg`, as well as the data for which you want to make predictions. If no data is given, the fitted values corresponding to the observed data are used. If data is provided, it needs to be given as a `data.frame` object, containing variables with the same variable names as in the original data.

Let us first extract the fitted values:

```
myfitted <- predict(budworm.lg)
myfitted
```

```
##          1          2          3          4          5          6          7
## -2.3724119 -1.3081980 -0.2439840  0.8202300  1.8844439  2.9486579 -3.4731553
##          8          9         10         11         12
## -2.4089413 -1.3447274 -0.2805134  0.7837006  1.8479145
```

By default, `predict` returns values on the scale of the linear predictor, i.e. it returns

$$\hat{\alpha} + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2,$$

where $\hat{\alpha}, \hat{\beta}_1, \hat{\beta}_2$ are the parameter estimates from the model fit, as given in the column `Estimate` of the model fit summary:

```
summary(budworm.lg)$coef
```

```
##              Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -3.473155  0.4685202 -7.413032 1.234445e-13
## sexmale      1.100743  0.3558271  3.093478 1.978249e-03
## ldose        1.064214  0.1310775  8.118971 4.701542e-16
```

While this is useful, it is easier to examine the model fit by comparing the fitted values on the scale of the response. For this, we use

```
myfitted <- predict(budworm.lg, type = "response")
myfitted
```

```
##          1          2          3          4          5          6          7
## 0.08530076 0.21278854 0.43930479 0.69428515 0.86812073 0.95020002 0.03008577
##          8          9         10         11         12
## 0.08249341 0.20673372 0.43032791 0.68647712 0.86388206
```

These are fitted probabilities of dead moths for each combination of `ldose` and `sex`.

Quick task(s):

Solve the task(s), and check your solution(s) here.

Chapter 4

Statistical tests - Part 2

4.1 Comparing more than two groups

4.1.1 Motivation

Consider the `quine` data on absenteeism from school in an Australian region. Per child involved in the study, the data includes the number of days absent from school in that year, as well as the age group (in 4 categories). One question is: does the number of absent days change depending on the school year?

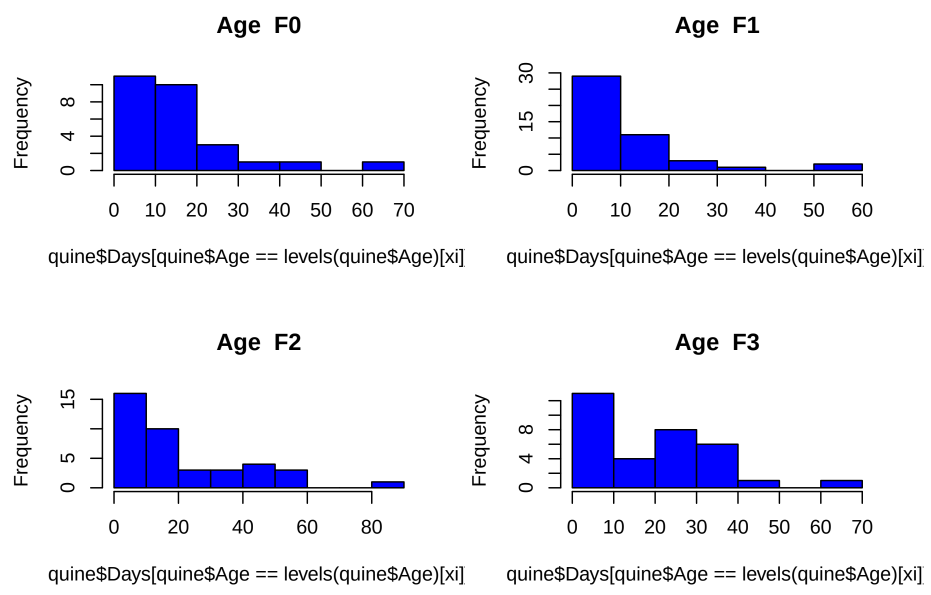
```
library(MASS)
data(quine)
str(quine)
```

```
## 'data.frame':    146 obs. of  5 variables:
##  $ Eth : Factor w/ 2 levels "A","N": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Sex : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Age : Factor w/ 4 levels "F0","F1","F2",..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ Lrn : Factor w/ 2 levels "AL","SL": 2 2 2 1 1 1 1 1 2 2 ...
##  $ Days: int  2 11 14 5 5 13 20 22 6 6 ...
```

```
# Examine how many observations you have per age group
table(quine$Age)
```

```
##
## F0 F1 F2 F3
## 27 46 40 33
```

```
# Now display the data for `Days` per age group
par(mfrow = c(2, 2))
for(xi in 1:nlevels(quine$Age)) {
  hist(quine$Days[quine$Age == levels(quine$Age)[ xi]],
        col = "blue", main = paste("Age ", levels(quine$Age)[ xi]))
}
```



You could try to solve this by applying a Student's-t test, or a Wilcoxon test, to compare each pair of `Age` levels. This would involve 6 tests.

In such situations, it would be better to compare the means of the groups under study in one go. What we wish to know is: is there at least one age group that does not have the same mean number of absent days as the others?

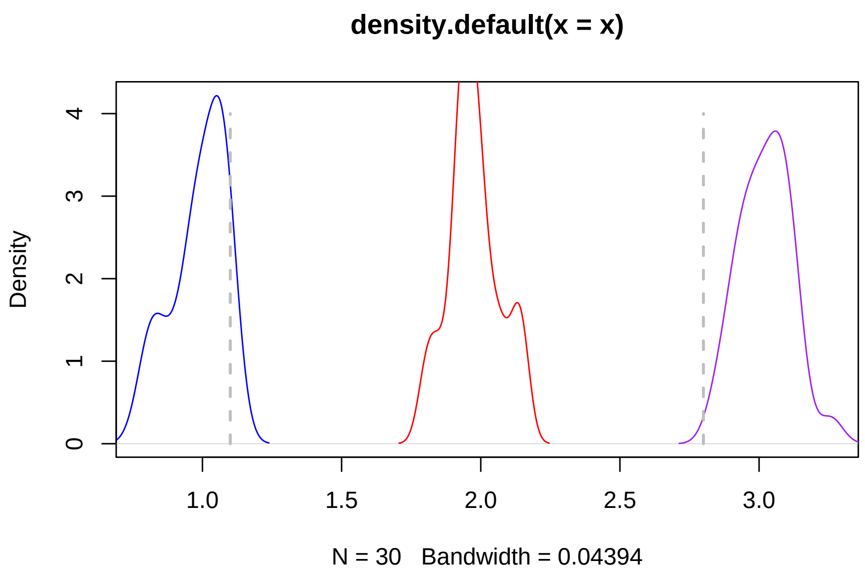
To do this, we need to better understand what we understand by "having the same mean".

4.1.2 Working examples

Consider the following data: a continuous variable is observed for cases within groups `X, Y, Z`. The variable follows a normal distribution with a mean depending on the group: 1, 2 and 3 for groups `X, Y, Z` respectively. Say that we have 100 observations for each group, and that the empirical densities of the values per group aree:

```
set.seed(53412)
ssize <- 30
x <- rnorm(ssize, mean = 1, sd = 0.1)
y <- rnorm(ssize, mean = 2, sd = 0.1)
z <- rnorm(ssize, mean = 3, sd = 0.1)
```

```r
mylims <- range(c(x, y, z))
plot(density(x), xlim = mylims, col = "blue")
lines(density(y), col = "red")
lines(density(z), col = "purple")
segments(1.1, 0, 1.1, 4, lty = "dashed", lwd = 2, col = "gray")
segments(2.8, 0, 2.8, 4, lty = "dashed", lwd = 2, col = "gray")
```

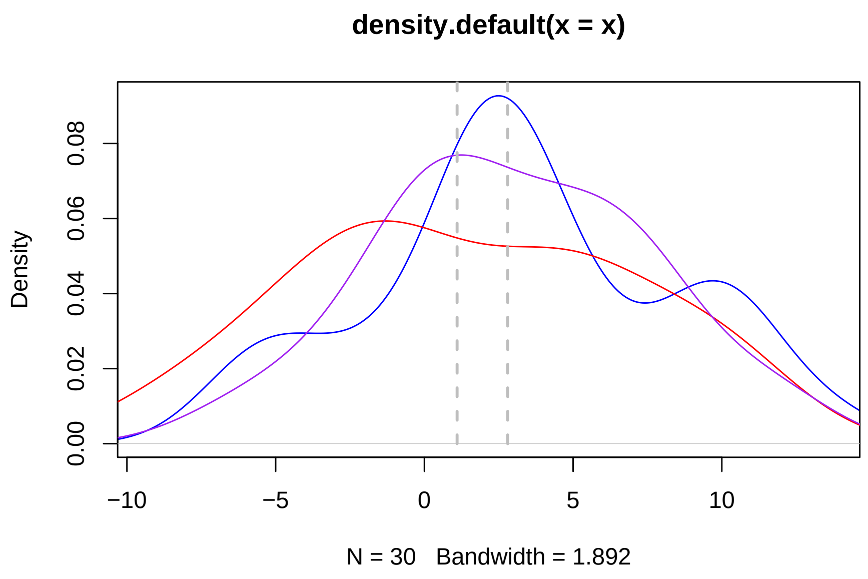


```r
data1 <- data.frame(x, y, z)
```

We can clearly separate observations arising from these 3 groups. Indeed, for example if we knew that a value of 1.1 was observed for one of these variables, we would guess that it corresponded to a case in group `X`. Similarly, if we knew that a value of 2.8 had been observed, we would safely guess it to correspond to a case in group `Z`.

Now let us simulate the data again for the same setup:

```r
set.seed(36912)
x <- rnorm(ssize, mean = 1, sd = 5)
y <- rnorm(ssize, mean = 2, sd = 5)
z <- rnorm(ssize, mean = 3, sd = 5)
mylims <- range(c(x, y, z))
plot(density(x), xlim = mylims, col = "blue")
lines(density(y), col = "red")
lines(density(z), col = "purple")
segments(1.1, 0, 1.1, 4, lty = "dashed", lwd = 2, col = "gray")
segments(2.8, 0, 2.8, 4, lty = "dashed", lwd = 2, col = "gray")
```



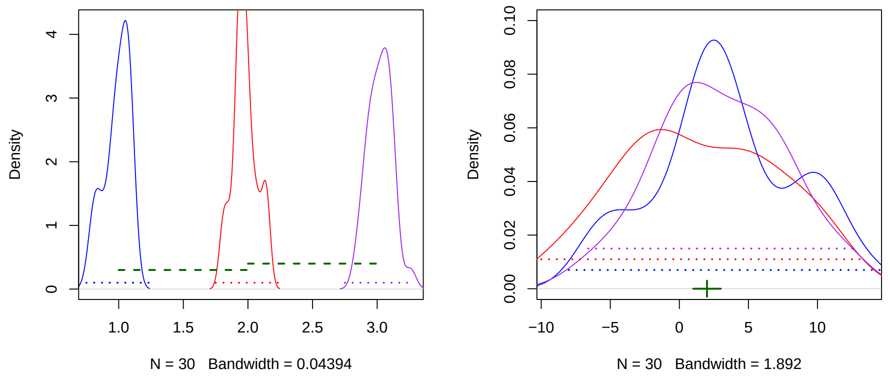
```
data2 <- data.frame(x, y, z)
```

In this case, it is difficult to distinguish data from these 3 groups. Indeed, we could now not guess to which group a value of 1.1 would correspond: it can be easily generated by cases in either `X` or `Y`, and even by some in `Z`. The same is true for 2.8. Note that the group means are the same as before: 1, 2 and 3. However, now the variance within each group is much larger than for the first data.

What this means is that: observations from these 3 groups are different from each other if they can be distinguished from each other, and that can only be done if the variance *between* groups is larger than the variance *within* groups.

```
par(mfrow = c(1, 2))
mydata <- data1
mylims <- range(c(mydata$x, mydata$y, mydata$z))
plot(density(mydata$x), xlim = mylims, col = "blue", main = "")
lines(density(mydata$y), col = "red")
lines(density(mydata$z), col = "purple")
segments(0.75, 0.1, 1.28, 0.1, lty = "dotted", lwd = 2, col = "blue")
segments(1.75, 0.1, 2.28, 0.1, lty = "dotted", lwd = 2, col = "red")
segments(2.75, 0.1, 3.28, 0.1, lty = "dotted", lwd = 2, col = "purple")
segments(1, 0.3, 2, 0.3, lty = "dashed", lwd = 2, col = "darkgreen")
segments(2, 0.4, 3, 0.4, lty = "dashed", lwd = 2, col = "darkgreen")
```

```
mydata <- data2
mylims <- range(c(mydata$x, mydata$y, mydata$z))
plot(density(mydata$x), xlim = mylims, col = "blue", main = "", ylim = c(0, 0.1))
lines(density(mydata$y), col = "red")
lines(density(mydata$z), col = "purple")
segments(-8, 0.007, 15, 0.007, lty = "dotted", lwd = 2, col = "blue")
segments(-10, 0.011, 13.5, 0.011, lty = "dotted", lwd = 2, col = "red")
segments(-6.6, 0.015, 12.8, 0.015, lty = "dotted", lwd = 2, col = "purple")
segments(1, 0.00, 3, 0.0, lty = "solid", lwd = 2, col = "darkgreen")
segments(2, -0.003, 2, 0.003, lwd = 2, col = "darkgreen")
```



Thus, in order to know if observations corresponding to the different groups can be distinguished or not, we need to consider how large the variance *between* groups (in green) is, relative to the variances within groups (in their respective colours). This is what the F test does, via the statistic F = variance between groups / variance within groups. If the variance between groups is large compared to the variance within groups, as with the first dataset, observations can be distinguished between groups. Otherwise, observations from different groups overlap, as in the second dataset. The F test is applied assuming that the groups cannot be distinguished, meaning that the variances within and between groups are similar. When observations from the different groups cannot be distinguished, the value of the F-test statistic is around 1. When observations can be distinguished between groups, the value of the F-test statistic will be relatively large, in particular larger than 1. How large it needs to be depends on the number of samples and the number of groups being compared, determining the degrees of freedom used in the F-test statistic.

The F test is the one used in ANOVA (which stands for ANalysis Of VAriance - now you know why!).

Let us apply this to the first dataset we generated. To do this, we need to fit a linear regression to the data with the observed variable as response and the group as explanatory factor. Then we compute the ANOVA table of the resulting fit:

```
group <- factor(rep(c("X", "Y", "Z"), each = ssize))
mydata <- data1
myvar <- c(mydata$x, mydata$y, mydata$z)
myfit <- lm(myvar ~ group)
anova(myfit)
```

```
## Analysis of Variance Table
##
## Response: myvar
##           Df Sum Sq Mean Sq F value    Pr(>F)
## group      2 62.145 31.0725  3524.3 < 2.2e-16 ***
## Residuals 87  0.767  0.0088
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Examine the ANOVA table. The column "Mean Sq" actually displays the estimated variances. The first row corresponds to `group` so it is the variance between groups `X, Y, Z`. The second row corresponds to `Residuals`, and this is the variance within groups - or the residual variance not explained by `group`. By dividing the variance between groups by the one within, the F-test statistic is obtained (note that the values printed are rounded off). As we expected, the F-test statistic is very large and its corresponding p-value is very small, in this case.

Now for the second dataset:

```
mydata <- data2
myvar <- c(mydata$x, mydata$y, mydata$z)
myfit <- lm(myvar ~ group)
anova(myfit)
```

```
## Analysis of Variance Table
##
## Response: myvar
##           Df  Sum Sq Mean Sq F value Pr(>F)
## group      2   61.75  30.875   1.176 0.3134
## Residuals 87 2284.17  26.255
```

The F test within the ANOVA table relies on the data arising from a normal distribution. If that is not the case, the alternative is to use the Kruskal-Wallis test, which can be thought of as the nonparametric version of the F test - much in the same way as the Wilcoxon test is the nonparametric version of the Student's-t test. Two alternative syntaxes for the Kruskal-Wallis test are:

```
kt <- kruskal.test(myvar, group)
kt
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  myvar and group
## Kruskal-Wallis chi-squared = 2.0227, df = 2, p-value = 0.3637
```

```
kt2 <- kruskal.test(myvar ~ group)
kt2
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  myvar by group
## Kruskal-Wallis chi-squared = 2.0227, df = 2, p-value = 0.3637
```

The Kruskal-Wallis test is based on ranks of the data. In this case, it is more powerful to find differences between groups than the F test from ANOVA.

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.2 Testing independence in 2x2 tables

4.2.1 Motivation

Consider again the `quine` data on absenteeism from school in an Australian region. Per child involved in the study, the data includes the number of students with average or slow learning speed `Lrn`, as well as `Sex`. One question is: do students of a given sex learn faster than students of the other sex? In other words: is there evidence for association between learning speed and sex? This problem is different from the of comparing two groups with a Student's-`t` test, because here both variables (`Lrn` and `Sex`) are grouping variables, whilst for the Student's-`t` test one variable is continuous and the other is a grouping variable.

We can examine the data to check for evidence of this association. This can be done by computing the number of cases within each class of learning speed `Lrn` and, of those, how many are of each `Sex`. We can also display the results as a barplot.

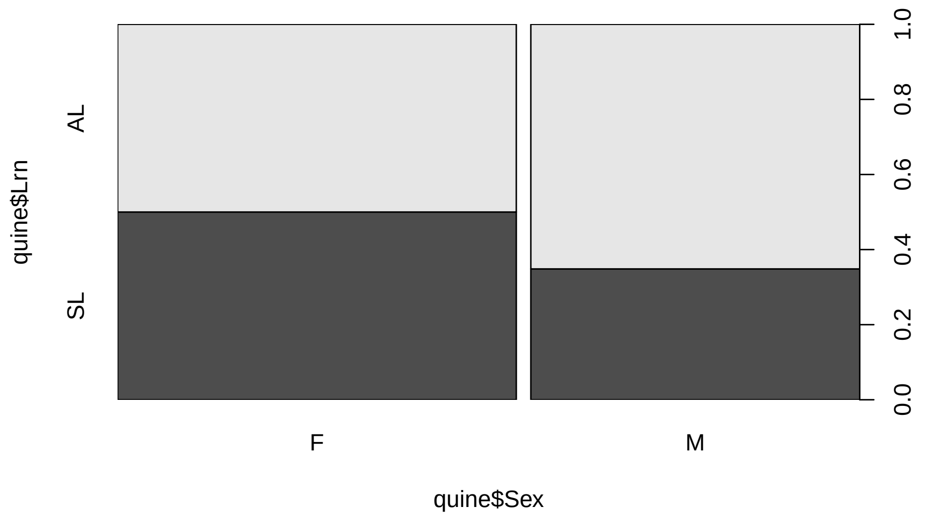
```
library(MASS)
data(quine)
str(quine)
```

```
## 'data.frame':    146 obs. of  5 variables:
##  $ Eth : Factor w/ 2 levels "A","N": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Sex : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Age : Factor w/ 4 levels "F0","F1","F2",..: 1 1 1 1 1 1 1 1 2 2 ...
##  $ Lrn : Factor w/ 2 levels "AL","SL": 2 2 2 1 1 1 1 1 2 2 ...
##  $ Days: int  2 11 14 5 5 13 20 22 6 6 ...
```

```
# Examine how many observations you have per age group
table(quine$Lrn, quine$Sex)
```

```
##
##       F  M
##   AL 40 43
##   SL 40 23
```

```
# Now display the data in this table as a barplot
plot(quine$Lrn ~ quine$Sex)
```



How do we decide, on the basis of the number of cases for each learning speed and sex, whether or not the two display association?

4.2.2 Working examples

```
set.seed(55498)
n <- 100
```

```
x1 <- rbinom(n, prob = 0.3, size = 1)
y1 <- rbinom(n, prob = 0.2, size = 1)

x2 <- x1
y2 <- rbinom(n, prob = 0.8*x2, size = 1)
```

Let us consider studies involving per case the observation of two variables, `X` and `Y`, each one having values $0, 1$. A table of the observed results is, in study 1:

```
table(x1, y1)
```

```
##     y1
## x1   0  1
##    0 53 11
##    1 27  9
```

and in study 2:

```
table(x2, y2)
```

```
##     y2
## x2   0  1
##    0 64  0
##    1  7 29
```

Looking at the tables it is difficult to decide whether or not the variables `X` and `Y` are independent or not. In fact, what sort of table would we expect to obtain, if `X` and `Y` were independent?

4.2.3 General setup

Consider the generic problem where two grouping variables `X, Y` are observed, each with groups labelled by either 0 or 1. If `X` and `Y` are independent, we would expect that the number of cases with `X=0` to be the same, regardless of the value of `Y`. The estimated probability of a case having `X=0` is just the proportion of cases with `X=0` in the study - say `px0`. As a consequence, the probability of a case having `X=1` is `1-px0`. The same holds for the number of cases with `Y=0`. We will represent by `py0` the probability that `Y=0`, so `1-py0` represents the probability that of a case having `Y=1`. If a total of `N` cases are observed, then the expected number of cases having `X=0` is `Npx0`, and the expected number of cases having `Y=1` is `N(1-py0)`.

If `X` and `Y` are independent, it follows the number of cases with both `X=0` and `Y=0` is expected to be `N px0 py0`, obtained from the product of the separate probabilities of each value of `X` and `Y`, in the study. For an overview, here is the table of expected frequencies in terms of these probabilities is:

Frequencies	Y=0	Y=1	Total
X=0	`N px0 py0`	`N px0 (1-py0)`	`N px0`
X=1	`N (1-px0) py0`	`N (1-px0) (1-py0)`	`N(1-px0)`
Total	`N py0`	`N (1-py0)`	`N`

So, if `X` and `Y` are independent, we know what to expect. Tests for independence of `X` and `Y` compare observed tables with this expected table. The expected table is constructed given the total counts for `X` and `Y` per possible value, so the row and column totals. These are called the *marginal frequencies* of `X` and `Y`. So tests for independence consider the row and column totals as fixed.

Note that, given the row and column totals, we need only fix one of the entries in the table, and all others are subsequently determined.

4.2.4 Working examples (cont)

For the first example, the row totals are 64 and 36 for `X=0` and `X=1` respectively, whilst the column totals are 80 and 20 for `Y=0` and `Y=1` respectively. This means that the empirical probabilities `px0, py0` are respectively 0.64 and 0.8.

If `X` and `Y` are independent, the expected table of frequencies given these marginal frequencies is:

Frequencies	Y=0	Y=1	Total
X=0	`N` 0.64 * 0.8	`N` 0.64 * 0.2	`N` 0.64
X=1	`N` 0.36 * 0.8	`N` 0.36 * 0.2	`N` 0.36
Total	`N` 0.8	`N` 0.2	`N`

where `N` in this case equals 100.

The table formed by the observed marginal frequencies, but not taking into account the actual observations, is

Frequencies	Y=0	Y=1	Total
X=0	`x0y0`	`x0y1`	64
X=1	`x1y0`	`x1y1`	36
Total	80	20	100

where `x0y0` and `x0y1` represent the counts on the row corresponding to `X=0`, for `Y=0` and `Y=1` respectively. Similarly, `x1y0` and `x1y1` represent the counts on the row corresponding to `X=1`, for `Y=0` and `Y=1` respectively.

So, it is possible to observe any table of this form, so long as the counts `x0 y0`, `x0 y1`, `x1 y0` and `x1 y1` are such that the marginal frequencies are observed as given.

As we pointed out, we need only choose a value for one of the entries, and that determines all the other entries in the table. For example, say we set `x0 y0` =1. Then the table becomes:

Frequencies	Y=0	Y=1	Total
X=0	1	64 -1	64
X=1	80 -1	36 - 80 +1	36
Total	80	20	100

Check that the same happens, regardless of the entry chosen. Fix a value for `x1 y0` and replace it in the table, and check which values you get for the remaining entries.

4.2.5 The chi-square test

One commonly used test for independence in 2x2 tables is the chi-square test. It involves comparing the observed counts in the table with what would be expected, if `X` and `Y` were independent. Given the marginal frequencies, one one entry in the table is needed to determine the entire table.

The test compares the observed (O) and expected (E) counts of one entry (say `x0 y0`) by means of the test statistic:

$$\chi = \frac{(O-E)^2}{E^2}$$

The null and alternative hypotheses are

H_0: `X` and `Y` are independent *vs.* H_a: `X` and `Y` are not independent

Under the null hypothesis, χ follows *asymptotically* a chi-square distribution with 1 degree of freedom - this because, given the marginal frequencies, the table is determined once one entry is fixed.

The term *asymptotically* means that the distribution of χ can be approximated by the chi-square with 1 d.f., under H_0, if the sample size is large enough. The concept of 'large enough' may differ depending on the researcher. Many researchers see this as requiring at least 5 observations in all entries of the table. For others, this means that all expected frequencies must be at least 5. The function `chisq.test` in R will give you a warning if the table used does not warrant the use of the chi-square test. If indeed the frequencies are too small to guarantee that the distribution of *chi* can be approximated by the chi-square, Fisher's exact test can be used (see below).

4.2.6 Working examples (cont)

We now apply the test to the working examples we had. For the first table we have:

```
table(x1, x2)
```

```
##    x2
## x1   0  1
##   0 64  0
##   1  0 36
```

```
chisq.test(table(x1, y1))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(x1, y1)
## X-squared = 0.45844, df = 1, p-value = 0.4984
```

Now for the second table:

```
myt <- table(x2, y2)
chisq.test(myt)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  myt
## X-squared = 68.754, df = 1, p-value < 2.2e-16
```

Note that the test result can be saved and separate slots of the test can be extracted. Here we extract the test statistic, the degrees of freedom and the p-value:

```
my.chtest <- chisq.test(myt)
my.chtest$statistic
```

```
## X-squared
##  68.75379
```

```
my.chtest$p.value
```

```
## [1] 1.11556e-16
```

```
my.chtest$parameter # this is the slot containing the p-value -- see ?chisq.test
```

```
## df
##  1
```

4.2.7 Fisher's exact test

The Fisher's exact test is another test that can be used to check whether or not two binary variables `X` and `Y` are independent. It calculates the probability of each possible 2x2 table of being observed, given the marginal frequencies. Then it adds up the probability of all tables at least as extreme as the one observed to yield a p-value. As it does not require approximations to compute the p-value, it is referred to as the 'exact' test, in contrast to the chi-square test with an approximated chi-square distribution. Yielding an exact p-value, it can be applied to all datasets.

4.2.8 Working examples (cont)

We will apply it now to the two tables of the examples. For the first example:

```
fisher.test(table(x1, y1))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(x1, y1)
## p-value = 0.4361
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.5169743 4.8487080
## sample estimates:
## odds ratio
##   1.598236
```

Now for the second example:

```
fisher.test(table(x2, y2))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(x2, y2)
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  48.98022      Inf
## sample estimates:
## odds ratio
##        Inf
```

As for the chi-square test result, slots of the test can be extracted. For example, we extract the p-value:

```
fisher.test(table(x2, y2))$p.value
```

```
## [1] 6.726116e-19
```

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.3 Testing independence in nx2 tables

4.3.1 Motivation

Consider now the relationship between learning speed `Lrn` and age `Age` in the `quine` data, from the package `MASS`:

```
library(MASS)
table(quine$Age, quine$Lrn)
```

```
##
##      AL SL
##   F0 19  8
##   F1 15 31
##   F2 16 24
##   F3 33  0
```

As before, we would like to know if data for the two variables involved, `Age` and `Lrn`, suggests that they are independent or not. This table has now more than 2 rows, so Fisher's exact test is not trivial to apply. However, the chi-square test can be used.

4.3.2 Working examples

```
set.seed(55498)
n <- 100
x1 <- rbinom(n, prob = 0.3, size = 2)
y1 <- rbinom(n, prob = 0.2, size = 1)

x2 <- x1
lp <- 0.3+2*x2/10
myprob <- exp(lp)/(1+exp(lp))
y2 <- rbinom(n, prob = myprob, size = 1)
```

Let us consider studies involving per case the observation of two variables, `X` and `Y`, each one having values $0, 1$. A table of the observed results is, in study 1:

```
table(x1, y1)
```

```
##     y1
## x1   0  1
##   0 38  7
##   1 33 10
##   2  9  3
```

and in study 2:

```
table(x2, y2)
```

```
##     y2
## x2   0  1
##   0 21 24
##   1  6 37
##   2  5  7
```

Looking at the tables it is difficult to decide whether or not the variables `X` and `Y` are independent or not. In fact, what sort of table would we expect to obtain, if `X` and `Y` were independent?

4.3.3 General setup

We will extend the setup previously used for 2x2 tables to the case where the table has more than 2 rows. As before, we assume that we are studying two grouping variables `X` and `Y`, where `X` may assume `n` values and `Y` may assume 2.

Say that `X` may assume 3 values, namely 0, 1 and 2. Then, analogously to the 2x2 table, the nx2 table in terms of empirical probabilities is:

Frequencies	Y=0	Y=1	Total
X=0	`N px0 py0`	`N px0 (1-py0)`	`N px0`
X=1	`N px1 py0`	`N px1 (1-py0)`	`N px1`
X=2	`N (1-px0-px1) py0`	`N (1-px0-px1) (1-py0)`	`N(1-px0-px1)`
Total	`N py0`	`N (1-py0)`	`N`

Tests for independence of `X` and `Y` compare observed tables with this expected table. The expected table is constructed given the total counts for `X` and `Y` per possible value, the marginal frequencies of `X` and `Y`. So tests for independence consider the row and column totals as fixed.

Note that, given the row and column totals, we need only fix one of the entries in the table, and all others are subsequently determined.

4.3.4 Chi-square test for nx2 tables

The extension of the chi-square test to nx2 tables is straightforward. We will briefly give the details here.

As before, the marginal frequencies of the observed table are considered as fixed. Using the same reasoning as for 2x2 tables, the probability of each entry in the table is equal to the product of the marginal probabilities.

For each row, since the marginal frequency (total row count) is given, it suffices to fix one of the entries, and the other one is determined. In addition, once one but the last row has one entry fixed, the last row is determined based on the total column counts. This means that, for the first n-1 rows, it suffices to fix one entry to determine the row, and this determines the last row. For this reason, the chi-square test for a nx2 table has n-1 degrees of freedom.

4.3.5 Working examples (cont)

Analogously to the development for 2x2 tables, if `X` and `Y` are independent in this 3x2 table example, the expected table of frequencies given these marginal frequencies is, for study 1:

Frequencies	Y=0	Y=1	Total
X=0	`N` 0.45 * 0.8	`N` 0.45 * 0.2	`N` 0.45
X=1	`N` 0.43 * 0.2	`N` 0.43 * 0.2	`N` 0.43
X=2	`N` 0.12 * 0.8	`N` 0.12 * 0.2	`N` 0.88
Total	`N` 0.8	`N` 0.2	`N`

where `N` in this case equals 100.

The table formed by the observed marginal frequencies, but not taking into account the actual observations, is

Frequencies	Y=0	Y=1	Total
X=0	`x0y0`	`x0y1`	45
X=1	`x1y0`	`x1y1`	43
X=2	`x2y0`	`x2y1`	12
Total	80	20	100

where `x0y0` and `x0y1` represent the counts on the row corresponding to `X=0`, for `Y=0` and `Y=1` respectively. Similarly, `x1y0` and `x1y1` represent the counts on the row corresponding to `X=1`, for `Y=0` and `Y=1` respectively. Finally, `x2y0` and `x2y1` represent the counts on the row corresponding to `X=2`, for `Y=0` and `Y=1` respectively.

So, it is possible to observe any table of this form, so long as the counts `x0y0`, `x0y1`, `x1y0`, `x1y1`, `x2y0` and `x2y1`, are such that the marginal frequencies are observed as given.

As we pointed out, we need only choose a value for one of the entries per row, and that determines the other entry in the row - except for the last row, which is determined if the first n-1 are filled in. For example, say we set all entries on the first column, for all but the last row, equal to 1. Then the table becomes:

Frequencies	Y=0	Y=1	Total
X=0	1	45 -1	45
X=1	1	43 -1	43
X=2	80 -2	12 - 80 +2	12
Total	80	20	100

You may wish to check for yourself that the same happens, regardless of the entries chosen to be fixed.

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.4 Testing symmetry in 2x2 tables

4.4.1 Motivation

Patients suspected of having lung cancer get a scan to check if the scan includes anomalies. The scan is analysed by a radiologist, but this may take a few days.

A computer program has been developed to analyse the images and detect anomalies. This is quicker than the radiologist, as the computer can be left continuously running. Researchers want to know if the computer program detects more anomalies than the radiologist or not. How can we decide on this?

Note: any anomaly detected is a positive result so, for the purpose of this example, there are no anomalies not worth being detected (false positives).

4.4.2 Working example

The data consists of two variables: the result of the radiologist `res.radio` and the result of the computer program `res.comp`. Each variable has one entry per scan, indicating by the value `1` if an anomaly was found in the scan, and by the value `0` otherwise.

To assess how often the two tests give similar results and how often they do not, we make a table of these variables:

```
table(res.radio, res.comp)
```

```
##             res.comp
## res.radio    no anomaly anomaly
##   no anomaly         45      12
##   anomaly             7      36
```

Now we need to decide if the difference observed is large enough to suggest one test detects more anomalies than the other.

4.4.3 General setup

In general, two variables `X` and `Y` are observed, each corresponding to one test test: *pass* or *fail*. A typical table observed is

Frequencies	Y=fail	Y=pass
X=fail	`a`	`b`
X=pass	`c`	`d`

The counts `a` and `d` correspond to cases where both tests yield the same result: in `a` cases both tests indicate a fail, and in `d` cases both tests indicate a pass. So, to decide on whether one of the two tests is better than the other, only counts `b` and `c` are informative.

If the tests disagree in a random manner, without a trend towards one yield a pass more often than the other, we expect to see similar values for `b` and `c`, relative to all results that disagree `b+c`. If test `X` more often yields a pass than test `Y`, we would expect to see `c` relatively larger than `b`, compared with `b+c`.

The statistical test to answer this question is McNemar's test. It is implemented in R via the `mcnemar.test` function. The test computes the probability of observing one of the discordant counts (`b`, say), or a more extreme value, out of a total `b+c` counts, at random. For that, typically an approximation of the true binomial distribution to the chi-square (with 1 degree of freedom) is used. In this course, we will use this function for our illustrations.

4.4.4 Working example (cont)

In the study with the scans evaluated by a radiologist and a computer program, we had:

```
table(res.radio, res.comp)
```

```
##             res.comp
## res.radio    no anomaly anomaly
##   no anomaly         45      12
##   anomaly             7      36
```

Now applying McNemar's test to the data:

```
mcnemar.test(table(res.radio, res.comp))
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  table(res.radio, res.comp)
## McNemar's chi-squared = 0.84211, df = 1, p-value = 0.3588
```

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.4.5 Notes

An exact version of the test makes use of the binomial distribution directly. An implementation of this version is also available in R, for example via the `exact2x2` package from Michael Fay, which can be found via the link below:

https://cran.r-project.org/web/packages/exact2x2

4.5 Relative risk and odds ratio

4.5.1 Motivation

Researchers want to better understand the relationship between smoking and stroke. For this, they have collected data of people who have suffered a stroke in the last 5 years, and from people with similar demographic variables who have never had a stroke. They have then asked these people whether or not they smoked cigarettes regularly, or had done in the past for at least a year, in the 5 years prior to the stroke - these people were considered smokers. All others were considered non-smokers.

This study thus involves cases and controls, and the latter are chosen so as to have a similar profile as that of cases, but there is no 1-1 matching between cases and controls - so no pairs (case, control). The data consists of two variables: `smoke` indicating whether an individual is categorized as a smoker (`yes`) or not (`no`); and `stroke` indicating whether an individual has had a stroke (`1`) or not (`0`).

The observed data can be summarized by tabulating `smoke` and `stroke`:

```
table(smoke, stroke)
```

```
##      stroke
## smoke   0   1
##   no  107  43
##   yes  26  74
```

Some of the questions researchers want to address are:

- is the risk of having a stroke different between smokers and non-smokers?
- are the odds of having a stroke larger for smokers, compared with non-smokers?

The first question relates to the risk of having a stroke given that someone is a smoker, relative to the risk of having a stroke given that the person is not a smoker. This is what we call a *relative risk*. In this context, *risk* has the meaning of probability.

The second question relates to odds, the relation between the number of people who get a stroke and the number of people who do not, given their smoking status. The comparison between the two odds is made via the *odds ratio*.

We will see these concepts in more detail in the following sections.

4.5.2 Relative risk

Consider a generic table observed as

Frequencies	`Y`=0	`Y`=1	Total
Cases	a	b	a+b
Controls	c	d	c+d
Total	a+c	b+d	N

where the patient status (`case` or `control`) is the variable of interest, and `Y` represents the explanatory grouping variable. There is interest in studying the association between the patient status and the explanatory variable.

The risk of being a case, given that `Y`=0, is calculated by a/(a+c). Similarly, the risk of being a case, given that `Y`=1, is calculated by b/(b+d). So, the relative risk is the ratio between these two risks, or (a/b)*(b+d)/(a+c).

If the risk in both groups is similar, the relative risk is approximately 1. Values lower than 1 indicate that the risk amongst observations with `Y`=1 is smaller than the risk amongst observations with `Y`=0. Reciprocally, values larger than 1 indicate that the risk amongst observations with `Y`=1 is larger than the risk amongst observations with `Y`=0.

For the example of stroke and smoking above, we have cases on row 2 and controls on row 1. In this case, the table is

Frequencies	Y=0	Y=1	Total
Controls	107	26	133
Cases	43	74	117
Total	150	100	250

Then the risk of stroke amongst non-smokers is 43 divided by 150, which gives

```r
myt <- table(stroke, smoke)
r.stroke.smoke0 <- myt[2, 1]/colSums(myt)[1]
r.stroke.smoke0
```

```
##        no
## 0.2866667
```

Similarly, the risk of stroke amongst smokers is 74 divided by 100, which gives

```r
r.stroke.smoke1 <- myt[2, 2]/colSums(myt)[2]
r.stroke.smoke1
```

```
##  yes
## 0.74
```

The relative risk of having a stroke in smokers compared to non-smokers is then 0.74 divided by 0.29, yielding

```r
rr <- r.stroke.smoke1/r.stroke.smoke0
round(rr, 4)
```

```
##    yes
## 2.5814
```

So, the chance of someone having a stroke is 2.58 larger amongst smokers than amongst non-smokers.

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.5.3 Odds ratio

Consider again the generic table observed as

Frequencies	Y=0	Y=1	Total
Cases	a	b	a+b
Controls	c	d	c+d
Total	a+c	b+d	N

The odds of being a case amongst individuals with `Y`=1 is b/d, whilst the odds of being a case amongst individuals with `Y`=0 is a/c. Then the odds ratio of being a case, of individuals with `Y`=1 compared to that of individuals with `Y`=0, is the ratio (b/d)/(a/c), qhich is the same as (bc)/(ad).

If the odds in both groups are similar, the odds ratio is approximately 1. Values lower than 1 indicate that the odds amongst observations with `Y`=1 are smaller than the odds amongst observations with `Y`=0. Reciprocally, values larger than 1 indicate that the odds amongst observations with `Y`=1 are larger than the odds amongst observations with `Y`=0.

For the example of stroke and smoking above, we have the odds of having a stroke amongst non-smokers being 43 divided by 107. Similarly, the odds of having a stroke amongst smokers is 74 divided by 26. This leads to the following odds: 0.4 amongst non-smokers and 2.85 amongst smokers. Then the odds ratio of having a stroke, between smokers and non-smokers, is 2.85 divided by 0.4, which gives 7.08.

```
or <- (myt[2, 2] * myt[1, 1]) / (myt[1, 2] * myt[2, 1])
```

Note again that, in this case, cases are on row 2 and controls on row 1.

So, the odds of someone having a stroke is 7.08 larger amongst smokers than amongst non-smokers.

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.5.4 Relative risk *vs.* odds ratio

In principle, both relative risk and odds ratio are statistics that can be computed, given any 2x2 tables. However, their usefulness, and interpretation, depends on the experimental design used.

The relative risk is evaluated assuming that the relative frequencies a/(a+c) and b/(b+d) are good approximations for the probabilities of being a case, given `Y`=0 and `Y`=1, respectively. Since calculations are conditional on the marginal frequencies a+c and b+d, these must represent well the frequencies of `Y`=0 and

Y=1 in the general population, in the sense that (b+d)/(a+c+b+d) is a good estimate of the proportion of individuals with Y=1 in the population under study.

In contrast, the odds ratio does not make use directly of the observed relative frequencies. So it needs not rely on representative (relative) marginal frequencies.

Typically relative marginal frequencies can only be expected in the context of prospective studies. Retrospective studies, especially case-control sudies, do not satisfy this assumption. For this reason, odds ratios are often used in the context of case-control studies, whilst relative risks can only be reliably interpreted in prospective (cohort) studies.

In some cases, the computed relative risk and odds ratio yield similar values. For example, if the marginal frequency a+c is approximately equal to c, and the marginal frequency b+d is approximately equal to d, then it is easy to see that the two statistics will yield similar results. One situation where both $a + c \approx c$ and $b + d \approx d$ occur is when the response involves rare cases, so that both a and c are small, and the marginal frequencies of both Y=0 as well as Y=1 are large enough.

4.5.5 Logistic regression models (cont)

The concepts of relative risk and odds ratio have been introduced above, and we computed them on the basis of 2x2 tables. This helps with understanding what they mean. However, 2x2 tables can only represent the relationship between a response (having a stroke or not) and one explanatory (grouping) variable (smoking habit). Most studies involve more variables, and thus more general methods to compute them are needed.

To compute the odds ratio, a logistic regression can be used, allowing for the inclusion of more explanatory variables. In addition, by making use of estimates arising from a regression model, it is straightforward to construct a confidence interval for, and perform a hypothesis test on, computed odds ratios.

Here we will check that this is the case. Indeed, if we fit a logistic regression to the stroke and smoking habit example, we get:

```r
lgfit <- glm(stroke ~ smoke, family = binomial)
summary(lgfit)
```

```
##
## Call:
## glm(formula = stroke ~ smoke, family = binomial)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.641  -0.822  -0.822   0.776   1.581
##
## Coefficients:
```

```
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.9116     0.1806  -5.049 4.44e-07 ***
## smokeyes      1.9576     0.2908   6.731 1.68e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 345.55  on 249  degrees of freedom
## Residual deviance: 294.35  on 248  degrees of freedom
## AIC: 298.35
##
## Number of Fisher Scoring iterations: 4
```

```
sfit <- summary(lgfit)$coef
```

The coefficient of the `smoke` variable (corresponding to `smoke=yes`) in the model is 1.96. It turns out that this is the same as the logarithm of the odds ratio:

```
# Include computation of or using smoke, stroke
log(or)
```

```
## [1] 1.957597
```

Alternatively, we compute `exp(beta)`, where `beta` is the coefficient of `smoke=yes` estimated by the logistic model. This gives 7.08, which is the same as the odds ratio 7.08 computed from the table.

```
exp(sfit[2, 1])
```

```
## [1] 7.08229
```

```
or
```

```
## [1] 7.08229
```

So, the log-odds ratio corresponds to the coefficient of a binary variable from a logistic regression. If more explanatory variables are included in the logistic regression, the computed log-odds ratio yields a measurement that is corrected for the effects of the remaining variables.

As we already know, a confidence interval for a coefficient of the fitted regression can be computed. We can use the function `confint` on the model fit object to do that:

```
confint(lgfit, 2)
```

```
## Waiting for profiling to be done...
```

```
##    2.5 %   97.5 %
## 1.399669 2.542158
```

This is the 95% confidence interval for the logarithm of the odds ratio. Typically there is interest in testing whether the odds ratio is equal to 1 (in which case the odds are the same in both groups). On this scale, this test is equivalent to testing whether the log-odds ratio is equal to 0. The above confidence interval does not include 0, indicating that the null hypothesis that the odds ratio is equal to 1 can be rejected, with a significance level of 0.05 (because it is a 95% confidence interval).

An approximate confidence interval for the odds ratio is then

```
exp(confint(lgfit, 2))
```

```
## Waiting for profiling to be done...

##     2.5 %    97.5 %
##  4.053859 12.707069
```

4.5.6 Poisson regression models

While odds ratio can be computed by fitting a logistic regression, relative risk can be computed by fitting a Poisson regression.

For some background: the logistic regression assumes that the response observations can be modelled by a binomial distribution, whilst the Poisson regression assumes that the response observations can be modelled by a Poisson distribution. The binomial distribution models counts for a given total, whilst the Poisson distribution models counts without conditioning on a total. So, while both regression models are suitable for a response variable that represents counts, in the logistic regression computations are conditional on the total number of cases and controls observed, which is not the case with the Poisson regression.

We now fit a Poisson regression to the stroke and smoking habits table:

```
lgfit <- glm(stroke ~ smoke, family = poisson)
summary(lgfit)
```

```
##
## Call:
## glm(formula = stroke ~ smoke, family = poisson)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2166  -0.7572  -0.7572   0.2867   1.0355
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.2494     0.1525  -8.193 2.54e-16 ***
## smokeyes      0.9483     0.1917   4.946 7.59e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 177.67  on 249  degrees of freedom
## Residual deviance: 152.01  on 248  degrees of freedom
## AIC: 390.01
##
## Number of Fisher Scoring iterations: 5
```

```
sfit <- summary(lgfit)$coef
```

The coefficient of the `smoke` variable (corresponding to `smoke=yes`) in the fitted Poisson regression is 0.95. It turns out that this is the same as the logarithm of the relative risk:

```
log(rr)
```

```
##       yes
## 0.9483301
```

Alternatively, we can compute the relative risk via `exp(beta)`, where `beta` is the estimated coefficient of `smoke=yes` in the Poisson regression. This yields 2.58, the same as the computed relative risk from the table 2.58.

```
exp(sfit[2, 1])
```

```
## [1] 2.581395
```

```
rr
```

```
##      yes
## 2.581395
```

Similarly to what was done for the odds ratio, we can now build a confidence interval for the coefficient of the fitted regression. We use the function `confint` on the model fit object to do that:

```
confint(lgfit, 2)
```

```
## Waiting for profiling to be done...
```

```
##     2.5 %    97.5 %
## 0.5776317 1.3315650
```

This is the 95% confidence interval for the logarithm of the relative risk. Typically there is interest in testing whether the relative risk is equal to 1 (in which case the risk the same in both groups). On this scale, this test is equivalent to testing whether the log-relative risk is equal to 0. The above confidence interval does not include 0, indicating that the null hypothesis that the relative risk is equal to 1 can be rejected, with a significance level of 0.05 (because it is a 95% confidence interval).

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.6 Power for analysis of count tables

4.6.1 Motivation

We want to know how many samples are needed if we were to repeat the study of learning speed and association with sex, using as pilot the data the `quine` data. Let us first review the data and make the table we will use for analyses:

```
library(MASS)
# Examine how many observations you have per age group
table(quine$Lrn, quine$Sex)
```

```
##
##       F  M
##   AL 40 43
##   SL 40 23
```

```
mytable <- table(quine$Lrn, quine$Sex)
```

The aim is to test if there is a statistically significant difference between the proportions of females, between the groups defined by learning speed. We will test this by using either Fisher's exact test or the chi-square test. Subsequently, we wish to know by how much the sample size would have to change, to modify the test's conclusions.

4.6.2 Fisher's exact test

Fisher's exact test for independence of the marginal counts in this table gives:

```
fisher.test(mytable)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  mytable
## p-value = 0.09285
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.258569 1.099752
## sample estimates:
## odds ratio
##  0.5371981
```

So the test is not significant if we use $\alpha = 0.05$. Now we will evaluate the power of the test to find the difference between proportions to be significant, given the current table. For this, we make use of the function `power.fisher.test` from the package `statmod`, which is automatically installed when you install R. As there is no closed form for the distribution of the Fisher's exact test statistic under the alternative, the power cannot be evaluated on the basis of a mathematical formula. Instead, the function simulates data according to given parameters, and yields the estimated power.

As inputs, the function expects the proportions of females in the two groups (`p1` and `p2`), the sample sizes in the two groups (the row sums), the significance level ($\alpha = 0.05$ is the default), the number of simulations and the alternative (one or two-sided). Below we use the function to compute the power for the table of learning speed and sex from the `quine` data:

```
library(statmod)
power.fisher.test(p1  = 40/83, p2 = 40/63, n1 = 83, n2 = 63)
```

```
## [1] 0.53
```

The estimated power is lower than 0.8, the level typically desired for studies. We could now increase the sample size and check if the desired power of 0.8 would be achieved. Note that we keep the proportions `p1` and `p2` the same, so that the effect size is the same as for the original data. We also keep the relative proportions in groups 1 and 2 the same, i.e., `n1/n2` is kept fixed. We can for example multiply both sample sizes by 2:

```
power.fisher.test(p1  = 40/83, p2 = 40/63, n1 = 83*2, n2 = 63*2)
```

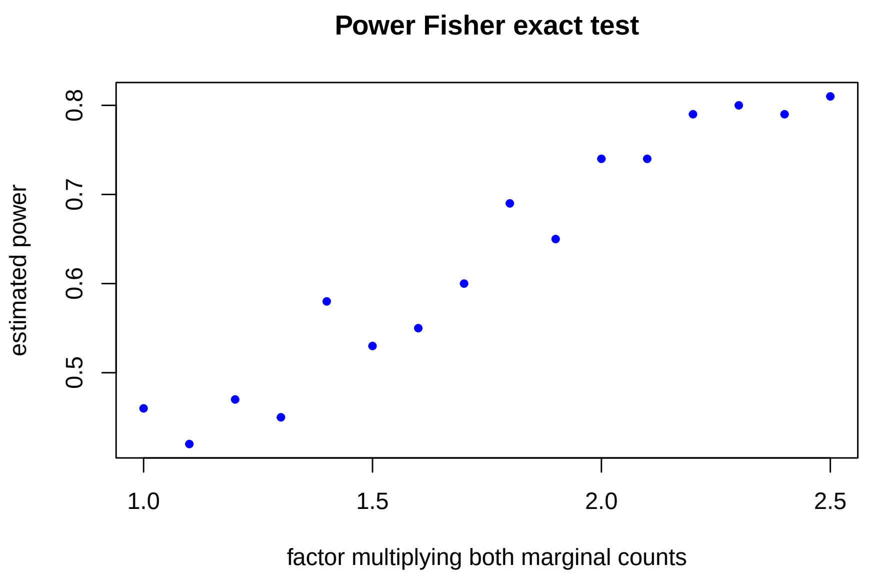
```
## [1] 0.74
```

This yields a number closer to 0.8. Let us now define a vector of values which we will multiply both sample sizes by, and compute the power for each value:

```
sr <- seq(from = 1, to = 2.5, by = 0.1)
powerv <- NULL
for(xi in 1:length(sr)){
  powerv <- c(powerv, power.fisher.test(p1  = 40/83,
              p2 = 40/63,
              n1 = floor(83*sr[xi]),
              n2 = floor(63*sr[xi])))}
```

Now we make a graph of the estimated power, as a function of the multiplying factor:

```
plot(sr, powerv, pch = 20, col = "blue", main = "Power Fisher exact test",
     xlab = "factor multiplying both marginal counts", ylab = "estimated power")
```



We used here a relatively small number of simulations, the default 100. This of course yields variability between estimates, meaning they are not always increasing.

So, a power of at least 0.80 is achieved for the last few multiplying factors, corresponding to minimum sample sizes required of:

```
min(sr[ powerv >= 0.8 ]*83)
```

```
## [1] 190.9
```

for group 1, and

```
min(sr[ powerv >= 0.8 ]*63)
```

```
## [1] 144.9
```

for group 2. This would yield a total sample size for a new study of:

```
min(sr[ powerv >= 0.8 ]*83) + min(sr[ powerv >= 0.8 ]*63)
```

```
## [1] 335.8
```

Note that Fisher's exact test makes no distinction between rows and columns, in the sense that the test statistic is the same, if we invert rows and columns. So, conclusions here would have been the same if we had used as inputs the proportions of learning speeds per sex, and the column sums, instead.

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.6.3 Chi-square test

The power for a chi-square test can be computed by using the function `pwr.chisq.test` of the package `pwr`. This function, as others from the same package, enables us to give all arguments but one, and the function will automatically determine the last argument using all the given ones. Here we will focus on determing the sample size, given the remaining arguments.

We will use in for the same example as above, where we studied the association between learning speed and sex, using the `quine` data. Let us first use the chi-square test to test for the association between the two variables, and save the observed test statistic:

```
chisq.test(mytable)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  mytable
## X-squared = 2.7949, df = 1, p-value = 0.09456
```

```
chisq.stat <- chisq.test(mytable)$stat
```

Based upon this test result, we conclude that there is no statistically significant association between learning speed and sex, with a significance level of 0.05. How large would the sample size need to be, in order to yield a statistically significant result?

We already knew that a chi-square test statistic for a 2x2 table has, under the null hypothesis of no association between rows and columns, asymptotically a chi-square distribution with 1 d.f.. The power we may state as 0.8, and the significance level remains 0.05.

We need also to state which effect size is required. This would be the effect size observed in the current data. It is equal to the square root of the observed test statistic, divided by the sample size of the current study. We have already computed the observed test statistic, and the sample size of the current study can be computed by `sum(mytable)`. Then we have:

```
eff.size <- sqrt(chisq.stat/sum(mytable))
```

Now load the package required and compute the minimum sample size:

```
library(pwr)
pwr.chisq.test(w = eff.size, df = 1, power = 0.8)
```

```
##
##      Chi squared power calculation
##
##               w = 0.1383592
##               N = 410.0061
##              df = 1
##       sig.level = 0.05
##           power = 0.8
##
## NOTE: N is the number of observations
```

Quick task(s):

Solve the task(s), and check your solution(s) here.

4.6.4 Odds ratio

The minimum sample size required to find a specific odds ratio as different from 1 is the same as the one required to find the log-odds ratio as different from 0. Thus, the power analysis required is the same as the one for a coefficient from the logistic regression.

4.6.5 Notes

We have used a choice of functions available in R to explain the concept of minimum sample size, and illustrate how it varies according to various parameters, for the chi-square and Fisher's exact tests. However, there are other choices of functions, which may be more convenient for a particular application.

To estimate the minimum sample size required for McNemar's test of symmetry, we suggest using the `exact2x2` package. We refer to the vignette of the McNemar's test functions implemented in the package for more details:

https://cran.r-project.org/web/packages/exact2x2/vignettes/exactMcNemar.pdf

Chapter 5

Survival analysis

5.1 Survival data - Introduction

5.1.1 Motivation

A migraine study wants to evaluate the time to recovery given a new drug (D2), compared to a commonly used drug (D1). Patients with migraine are invited to take part in the study, where they are randomized into two groups, each given one of the drugs. In each group, participants must take the drug they are given shortly after a new migraine episode starts. They record the time of taking the drug, and the time the migraine has stopped. In the course of the study, some patients could not record the time of recovery as they fell asleep before it happened. In such cases, all patients recorded that they had recovered by the time they woke up.

The response in this case involves the event observed, recovered or not after a single dose of the assigned drug, as well as the time to event. The explanatory variable is the drug received. Of course, it is possible to analyse the association between each of the two variables with the drug used separately from each other, but doing this ignores important issues. Firstly, the event observed (recovered or not) may be related to the time to event. Secondly, there is interest in drawing conclusions for the combination of event as well as time-to-event variable, instead of for them separately. So, there is interest in considering the two variables as a single response. Indeed, a model that takes both event as well as time-to-event into account at the same time has more power to find effects than one that analyses the two variables separately. In addition, it yields conclusions about effects on both variables at the same time.

Another important aspect is that, for some participants, the precise time at which recovery happened was not recorded, but it is known to have been after a certain time point. This means that the time-to-event is not precisely known, but known to be at least a certain value, for these participants. Such observations are said to be *censored*.

We will now see how to analyse such a response variable, formed by a combination of an event variable with the time-to-event, where sometimes the time-to-event is not precisely known.

5.1.2 Working example

Consider the migraine study with the two drugs. The time-to-event is in this case the time (in minutes) between taking the assigned drug and the event. The event can be `recovery` (coded as 1) or `fail` (coded as 0), where the latter corresponds cases where the participant falls asleep without having recovered beforehand. Note that the time-to-event in case of a `fail` event is recorded as the time between taking the drug and the last moment the participant is known to not having recovered, which is when the participant fell asleep. As such, the time-to-event is at least as long as the time recorded, since migraine recovery may have recovered at any time while the participant slept.

```
set.seed(39562) # (493756)
ss <- 400
fdrug <- factor(rep(1:2, each = ss/2), labels = c("D1", "D2"))
event <- c(rbinom(ss/2, 1, prob = .7), rbinom(ss/2, 1, prob = .75))
tte <- round(rgamma(ss, shape = 30, rate = 1+event/4))
migraine <- data.frame(event, time = tte, drug = fdrug)
write.table(migraine, file = "migraine_data.txt", sep = "\t", row.names = FALSE)
```

The observed data for a subset of 10 participants is as follows:

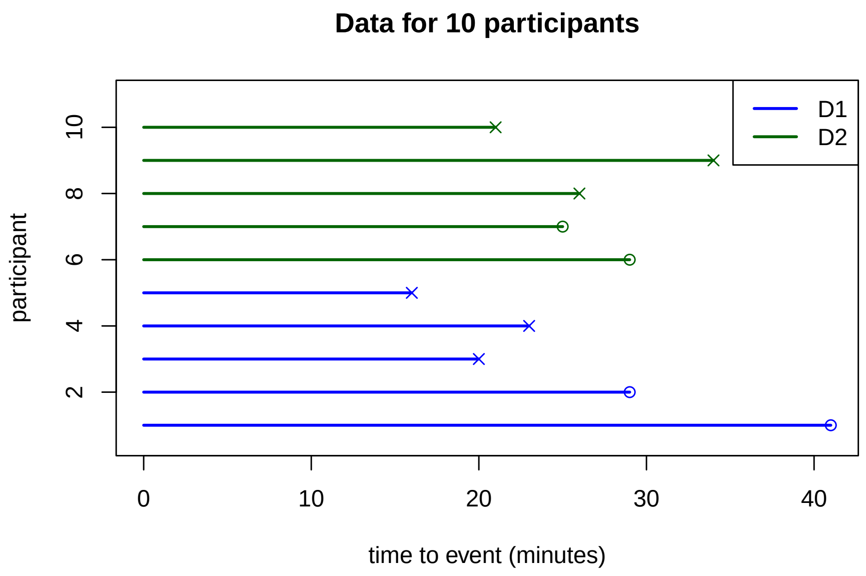
```
mydata <- rbind(migraine[1:5, ], migraine[(ss/2+1):(ss/2+5), ])
mydata
```

```
##     event time drug
## 1       0   41   D1
## 2       0   29   D1
## 3       1   20   D1
## 4       1   23   D1
## 5       1   16   D1
## 201     0   29   D2
## 202     0   25   D2
## 203     1   26   D2
## 204     1   34   D2
## 205     1   21   D2
```

The data can be visualized in the following way:

```
#mydata <- rbind(migraine[1:5, ], migraine[(ss/2+1):(ss/2+5), ])
myxlim <- c(0, max(mydata$time))
mypch <- rep(4, nrow(mydata))
```

```
mypch[ mydata$event == 0 ] <- 1
plot(1, 1, xlim = myxlim, ylim = c(0.5, nrow(mydata)+1), col = "white",
     main = "Data for 10 participants", xlab = "time to event (minutes)", ylab = "participant")
mcol <- rep(c("blue", "darkgreen"), each = nrow(mydata)/2)
for(xi in 1:nrow(mydata)) segments(0, xi, mydata$time[xi], xi,  col = mcol[xi], lwd = 2)
for(xi in 1:nrow(mydata)) points(mydata$time[xi], xi, col = mcol[xi], pch = mypch[ xi ])
legend("topright", legend = c("D1", "D2"), lty = "solid", lwd = 2,
       col = c("blue", "darkgreen") )
```



In the graph, the time point 0 represents the moment the participant took the drug for that migraine event, and the horizontal line indicates the period during which the participant's migraine is known to have persisted. The right-hand point where the line ends includes a symbol representing the event observed: `x` indicates a recovery event, and `o` indicates a `fail` event.

5.1.3 Kaplan-Meier curve

One of the most commonly used methods to analyse survival data is the Kaplan-Meier curve. It combines information from an event variable with the time-to-event to yield a probability of survival per time point. Originally called the product limit estimator, it is a result of ideas proposed independently by E. Kaplan and P. Meier for publication by the Journal of the American Statistical Association, which by suggestion of the journal's editor was combined into a single article published in 1958.

Here we will present an intuitive view of how the Kaplan-Meier curve is computed, and how it can be interpreted. More formal introductions are widely available, for example from chapter 12 of Peter Dalgaard's book.

The Kaplan-Meier curve essentially yields an estimate of the probability of survival beyond each time point. By "survival" here we mean a time-to-event longer than the time point considered. A graph is produced using the estimated probabilities for multiple time points.

Such a probability of survival computed for each time point essentially is the proportion of individuals at risk of having an event just before the time point subtracted from the number of events at the time point, relative to all individuals. Note that the proportion of individuals at risk changes only when an event is observed, and the total number of individuals typically stays fixed. So, the Kaplan-Meier probability of survival estimates only changes at the time points where events are observed.

Censored observations (in the sense defined above) belong to the group of all individuals at risk during the entire duration of the study, since we essentially do not know anything about their time-to-event, except that it was larger than a certain time.

Computing the Kaplan-Meier curve does not require any assumptions about the data distribution (event or time-to-event), apart from the facts that event is a grouping variable (as it defines one of a set of known events, per individual) and time-to-event is a non-negative variable (as it represents time). This means it is a non-parametric method, and it makes it widely applicable.

5.1.4 Working example (cont)

We can illustrate this with the subset of the migraine study, where we first consider participants from a single group, thus making use of the same drug:

```r
mydata <- migraine[1:10, ] #
mydata1 <- mydata[, -3]
mydata1
```

```
##    event time
## 1      0   41
## 2      0   29
## 3      1   20
## 4      1   23
## 5      1   16
## 6      1   15
## 7      0   23
## 8      0   31
## 9      1   22
## 10     1   28
```

We can easily see how to obtain the estimated probability that the recovery takes longer than a given time, per time point. For any time point prior to the first event occurring, the probability of recovery taking longer than that time point is the number of participants at risk at that time point divided by the total of participants, so it equals 1. Subsequently, for each time point shortly before an event (recovery) is observed, the number of participants at risk decreases by the number of events, which equals 1. We illustrate this by displaying the computed survival probabilities below for the subset of the `migraine` data:

```
mydatas <- mydata1[order(mydata1$time), ]
nrisk <- nrow(mydatas) - cumsum(mydatas$event)[-c(3, 5)]
prob.surv <- nrisk/rep(nrow(mydatas), length(nrisk))
prob.surv
```

```
## [1] 0.9 0.8 0.6 0.5 0.4 0.4 0.4 0.4
```

The first two time points correspond to 1 event, but on the third time point 2 events are observed, explaining the larger drop.

Note that, for the time points corresponding to censored time-to-event observations, the probability did not change, as no event was observed. For those, no computation is needed.

Kaplan-Meier curves yield estimates for the survival probabilities. Confidence intervals are often obtained via what is called Greenwood's formula. As the Kaplan-Meier curve, these are nonparametric estimates since they do not rely on any assumption about the data distribution.

Note that estimates of both the Kaplan-Meier curve as well as of the confidence intervals on the right-hand side of the plot typically are associated with high variability. Therefore, care must be taken when interpreting results for the part of the curve with few individuals at risk. More details about this can be found in chapter 12 of Peter Dalgaard's book.

5.1.5 Survival data analysis in R

In R, classic survival data analysis methods are available via the package `survival`. Let us start by loading it.

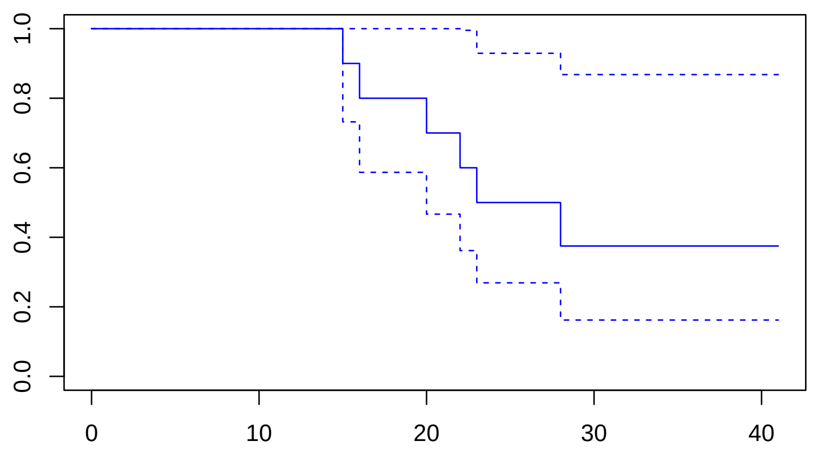
```
library(survival)
```

A survival data analysis starts by the creation of the response object, which combines the event variable with the time-to-event. In the migraine study example, these are variables `event` and `time`. The response object is created by the function `Surv()`:

```
migr.surv <- Surv(time = mydata1$time, event = mydata1$event)
```

With the response object created, it is straightforward to plot the Kaplan-Meier curve yielded by this object:

```
plot(migr.surv, col = "blue")
```



The computation of the Kaplan-Meier curve is actually done using the function `survfit`. It requires as input a formula object, with the `Surv` object as response and for example a grouping variable as covariate. In this case we will consider no grouping, so we use `1` to represent a model with just the intercept:

```
svfit <- survfit( migr.surv ~ 1)
```

The summary of `survfit` yields the computed survival probabilities. In particular, we can now check that the computed probabilites we obtained above are correct:

```
summary( svfit )
```

```
## Call: survfit(formula = migr.surv ~ 1)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    15     10       1    0.900  0.0949        0.732        1.000
##    16      9       1    0.800  0.1265        0.587        1.000
##    20      8       1    0.700  0.1449        0.467        1.000
##    22      7       1    0.600  0.1549        0.362        0.995
##    23      6       1    0.500  0.1581        0.269        0.929
##    28      4       1    0.375  0.1606        0.162        0.868
```

Quick task(s):

Solve the task(s), and check your solution(s) here.

5.1.6 Survival data for groups: the log-rank test

Typically there is interest in studying the association between a survival response and a covariate, say a grouping variable. This was the case in the `migraine` working example, where each group made use of a different drug to treat migraine. There is then interest in comparing the survival functions of the two groups.

We can compute the Kaplan-Meier estimators per group, but how do we compare the two obtained curves? This can be done using the log-rank test.

Let us assume that the covariate splits individuals in the study into two groups, and that Kaplan-Meier estimates of the survival distributions were computed per group. The log-rank test helps us test the null hypothesis that two survival distributions are statistically the same, against the alternative hypothesis that they are different. The concept of “statistically the same” seems rather vague. What we mean by it is: under the null hypothesis, differences between the two estimated survival distributions arise at random. This can be seen as comparing the number of observed events/no events per interval (defined by the time between two consecutive events) between the two groups, in what essentially is a 2x2 table:

Frequencies	Group 2, no event	Group 2, event	Total
Group 1, no event	y11	y12	y1.
Group 1, event	y21	y22	y2.
Total	y.1	y.2	N

For any given interval, this comparison can be done by means of a Fisher’s exact test (which we saw in chapter 4). The log-rank test results if, per interval, the observed statistic is compared to the expected, across all intervals. The distribution of the resulting test statistic can be approximated by the chi-square with 1 degree of freedom, the same that we used for 2x2 tables when the individual frequencies were large enough.

Note that the log-rank test assumes that the censoring is unrelated to the response variables, so both to event as well as to time-to-event. In fact, many of the survival data analysis methods make this assumption. It is therefore crucial to verify that this assumption holds in any data being analysed. This cannot be done by a statistical test, but you could for example

In R, this can be done by using the `survfit` function. The argument is a formula object, defining the comparison to be performed.

5.1.7 Working example (cont)

Let us compare the obtained survival curves between groups using drugs D1 and D2 in a small subset of the migraine example. First select the subset:

```
mydata <- rbind(migraine[1:10, ], migraine[(ss/2+1):(ss/2+10), ])
```

Then create the response, the `Surv` object:

```
migr.surv <- Surv(time = mydata$time, event = mydata$event)
```

Now use `survfit` to compute the Kaplan-Meier per group. This function uses as input a formula, defining the `Surv` response and the grouping variable `drug` indicating the drug used per individual:

```
svfit <- survfit(Surv(migraine$time, migraine$event) ~ migraine$drug)
```

The object yielded by `survfit` is of class `survfit`. We can print out a summary of the object:

```
print(svfit)
```

```
## Call: survfit(formula = Surv(migraine$time, migraine$event) ~ migraine$drug)
##
##                        n events median 0.95LCL 0.95UCL
## migraine$drug=D1 200    149     25      24      26
## migraine$drug=D2 200    146     25      24      26
```

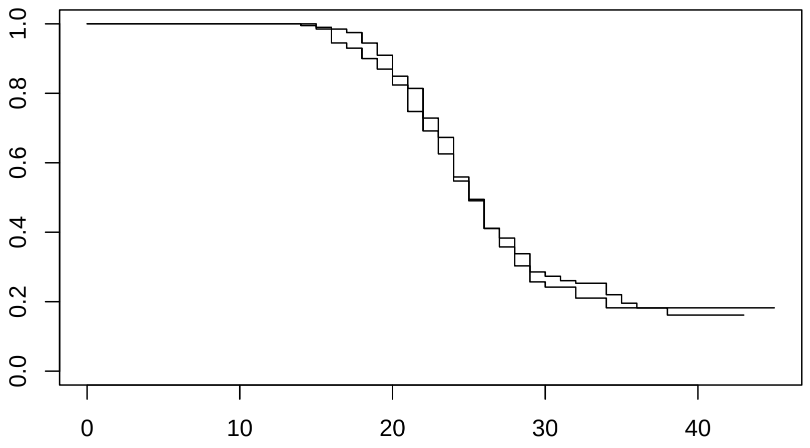
and prind the Kaplan-Meier survival probabilities estimated for the two groups:

```
summary(svfit)
```

```
## Call: survfit(formula = Surv(migraine$time, migraine$event) ~ migraine$drug)
##
##                 migraine$drug=D1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    15    199       2    0.990 0.00707        0.976        1.000
##    16    197       1    0.985 0.00864        0.968        1.000
##    17    196       2    0.975 0.01109        0.953        0.997
##    18    194       6    0.945 0.01620        0.914        0.977
##    19    188       7    0.910 0.02033        0.871        0.950
##    20    181      12    0.849 0.02536        0.801        0.900
##    21    169       7    0.814 0.02758        0.762        0.870
##    22    162      17    0.729 0.03152        0.669        0.793
##    23    144      11    0.673 0.03328        0.611        0.741
##    24    130      22    0.559 0.03542        0.494        0.633
##    25    106      13    0.491 0.03582        0.425        0.566
##    26     92      15    0.411 0.03543        0.347        0.486
##    27     75       5    0.383 0.03512        0.320        0.459
##    28     68       8    0.338 0.03442        0.277        0.413
##    29     58       9    0.286 0.03322        0.227        0.359
##    30     46       2    0.273 0.03292        0.216        0.346
##    31     43       2    0.261 0.03259        0.204        0.333
##    32     35       1    0.253 0.03250        0.197        0.325
##    34     23       3    0.220 0.03338        0.163        0.296
##    35     18       2    0.196 0.03386        0.139        0.275
##    36     14       1    0.182 0.03420        0.126        0.263
##    38      9       1    0.161 0.03586        0.104        0.250
##
##                 migraine$drug=D2
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    14    200       1    0.995 0.00499        0.985        1.000
##    15    199       2    0.985 0.00860        0.968        1.000
##    16    197       8    0.945 0.01612        0.914        0.977
##    17    189       3    0.930 0.01804        0.895        0.966
##    18    186       6    0.900 0.02121        0.859        0.943
##    19    178       6    0.870 0.02384        0.824        0.918
##    20    171       9    0.824 0.02703        0.773        0.879
##    21    162      15    0.748 0.03088        0.689        0.811
##    22    147      11    0.692 0.03286        0.630        0.759
##    23    136      13    0.626 0.03445        0.562        0.697
##    24    120      15    0.547 0.03557        0.482        0.622
##    25    104      10    0.495 0.03584        0.429        0.570
##    26     89      15    0.411 0.03568        0.347        0.488
##    27     69       9    0.358 0.03523        0.295        0.434
##    28     59       9    0.303 0.03423        0.243        0.378
##    29     46       7    0.257 0.03316        0.200        0.331
##    30     34       2    0.242 0.03289        0.185        0.316
##    32     23       3    0.210 0.03326        0.154        0.287
##    34     15       2    0.182 0.03423        0.126        0.263
```

Making a graph of the Kaplan-Meier curves is also straightforward:

```
plot(svfit)
```



The log-rank test is computed by using the function `survdiff` on the comparison to be made, represented by the formula involving the `Surv` object as response, and the grouping variable on the right-hand side:

```
survdiff(migr.surv ~ mydata$drug)
```

```
## Call:
## survdiff(formula = migr.surv ~ mydata$drug)
##
##                  N Observed Expected (O-E)^2/E (O-E)^2/V
## mydata$drug=D1 10        6      5.6    0.0292    0.0572
## mydata$drug=D2 10        6      6.4    0.0255    0.0572
##
##  Chisq= 0.1  on 1 degrees of freedom, p= 0.8
```

Quick task(s):

Solve the task(s), and check your solution(s) here.

5.2 Survival data - regression models

5.2.1 Motivation

We have seen that we can estimate survival probability curves by using the Kaplan-Meier method, and that we can test if they are intrinsically the same using the log-rank test. This test is used for comparing the survival distribution curves between two groups. However, the effect of variables with more groups, or of continuous variables, and that of many variables at the same time, cannot be (directly) tested by means of the log-rank test.

What we want to do is to use a framework like that of regression models, to be able to study the effect on the survival response of multiple variables, and of multiple types of variables, at the same time. Typically regression models only consider one response at the time. So what needs to be done to consider these two variables, in a way so that survival probabilities can be estimated?

The first step is to understand what we want to model. Say that, instead of the survival response, we observe a rate λ_i at which patients die, for samples indexed by $i = 1, \ldots, n$. We will model this using regression as a linear function of a covariate X_i. So we could write

$$\lambda_i = \alpha + \beta X_i, \ i = 1, \ldots, n.$$

Since λ_i is a rate, it is always non-negative. The model above could, however, yield estimates that are negative. To avoid this, we use a logarithmic link, as in:

$$\log(\lambda_i) = \alpha + \beta X_i, \ i = 1, \ldots, n,$$

or, equivalently,

$$\lambda_i = \exp(\alpha + \beta X_i), \ i = 1, \ldots, n.$$

This model arises naturally if patients die at an exponential rate, where the exponential is a probability distribution. Other distributions can be used instead of the exponential, but the basic idea remains to explain the rate at which events are observed by a log-linear function of the covariates. This part of the model represents the contribution of covariates.

A general formulation for modelling survival data is as follows: the hazard $h(t)$ of events at any time t is a product of a baseline hazard $h_0(t)$, common to all observations and varying with t, multiplied by a part that depends only on the covariates and is fixed in time. That is:

$$h_i(t) = h_0(t) \exp(\alpha + \beta X_i), \ i = 1, \ldots, n,$$

where the hazard $h_i(t)$ represents the probability of an event being observed at t for observation i, given that i has not had an event up until t. In this formulation, the baseline hazard function describes how events are generated over time, considering all observations. Then the effect of a covariate just represents a multiplicative effect on the baseline hazard for the entire study period.

Such a formulation is based on a few asumptions:

- the covariate effect is fixed in time;
- there is a common baseline hazard yielding events for all observations, which is independent of the covariates;

The latter implies that

$$h_i(t)/h_j(t) = \frac{h_0(t)\exp(\alpha + \beta X_i)}{h_0(t)\exp(\alpha + \beta X_j)} = \exp(\beta(X_i - X_j)),$$

which means that the hazard ratio for two observations at the same time point t does not depend on t, only depending on their covariates. This means that these observations have proportional hazards.

Most regression models for survival data make use of this assumption, modelling the baseline hazard as the only function of time, multiplied by a log-linear function of the covariates. Classes of models arise depending on how the baseline hazard $h_0(t)$ is modelled.

Parametric models for the baseline hazard include the exponential, as used above, the gamma, the log-normal and the Weibull. These can be fitted with the R function `survreg`. While they may be of interest, in practice non-parametric hazard functions are more commonly used, as they are more flexible. For this reason, we will here focus on survival regression models with non-parametric hazards.

5.2.2 Cox proportional-hazards model

Perhaps the best known approach for estimating a non-parametric baseline hazard function in the context of a proportional-hazards model is the one proposed by D. R. Cox in 1972. The focus of fitting shifts from estimating the baseline hazard to estimating the proportional factors, the coefficients of the covariates in the model.

Two important assumptions of the proportional hazards model remain: the effect of covariates does not vary with time, and the proportionality of hazards.

5.2.3 Working example

To illustrate methods we will use the dataset `Melanoma`. Check what it contains by reading its help file, and examining the structure of the data.

```
# ?Melanoma
library(survival)
library(MASS)
str(Melanoma)
```

```
## 'data.frame':    205 obs. of  7 variables:
##  $ time     : int  10 30 35 99 185 204 210 232 232 279 ...
##  $ status   : int  3 3 2 3 1 1 1 3 1 1 ...
##  $ sex      : int  1 1 1 0 1 1 1 0 1 0 ...
##  $ age      : int  76 56 41 71 52 28 77 60 49 68 ...
##  $ year     : int  1972 1968 1977 1968 1965 1971 1972 1974 1968 1971 ...
##  $ thickness: num  6.76 0.65 1.34 2.9 12.08 ...
##  $ ulcer    : int  1 0 0 0 1 1 1 1 1 1 ...
```

In this example, `time` is the survival time in days, and `status` is an indicator of the patient's status by the end of the study:

- `status` = 1: "dead from malignant melanoma";
- `status` = 2: "alive on January 1, 1978";
- `status` = 3: dead from other causes"

We will first create a `Surv` object having only the event `dead from melanoma`, and both alive as well as dead due to other causes together as censoring. For this, we use:

```
survm <- Surv(Melanoma$time, Melanoma$status == 1)
```

So, in this case the endpoint variable used in `Surv` is a logical vector, equal to `TRUE` if `status` = 1, and `FALSE` otherwise.

Now we want to know if `sex` affects the survival probability of melanoma. To do this, we compute the Cox regression using the `Surv` object defined above, and `sex` as a covariate:

```
mel.cox <- coxph(Surv(time, status == 1) ~ sex, data = Melanoma)
summary(mel.cox)
```

```
## Call:
## coxph(formula = Surv(time, status == 1) ~ sex, data = Melanoma)
##
##   n= 205, number of events= 57
##
##       coef exp(coef) se(coef)     z Pr(>|z|)
## sex 0.6622    1.9390   0.2651 2.498   0.0125 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## sex     1.939     0.5157     1.153      3.26
##
## Concordance= 0.59  (se = 0.034 )
## Likelihood ratio test= 6.15  on 1 df,   p=0.01
## Wald test            = 6.24  on 1 df,   p=0.01
## Score (logrank) test = 6.47  on 1 df,   p=0.01
```

Interpreting the summary output is similar to that of a fitted regression model: you have a table of coefficients with estimates, standard errors and a p-value for the test that the coefficient is equal to zero (or not). The end of the summary lists various tests that evaluate the goodness-of-fit of the model, so indicate whether or not the model represents well the data.

The choice of group of interest will define the sign of the coefficient of `sex` in the Cox model fit. In this example, the event of interest is death and the group corresponding to `sex`=0 (females) is taken by the intercept, representing the baseline. Then the coefficient estimated for `sex` corresponds to the difference in log-hazard for individuals with `sex`=1 (males) compared to those with `sex`=0. In this particular case, the computed coefficient is positive, indicating that male patients have a worse prognosis (higher progression probability to death, taking both time and event into account) than females. So, in this example, *survival probability* actually means *progression to death*, as the event of interest is death.

Note also that, if the group coding is inverted, with the second group now being assigned to the intercept, then the estimated coefficient of the grouping variable is the same as the one with the original coding, multiplied by -1.

The summary of the fitted model has one additional part, compared to other regression models: a table of coefficients and confidence intervals on the exponential scale. This is given to make it easier to interpret the coefficients on the scale of the survival probability.

5.2.4 Group-specific baseline hazards

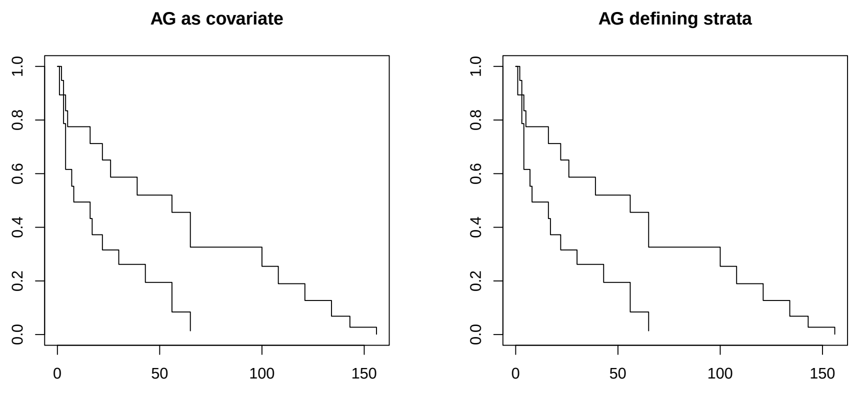
The Cox model is easily extended to allow different baseline hazards for different groups. For example, in the `leuk` data there could be interest in allowing for different event rates in time, depending on the `ag` group of the patient - patients termed `AG` positive display Auer rods and/or significant granulation of their leukemic cells, which may affect the overall event rate. Different hazards according to `ag` group can be fitted by including the argument `strata` as `ag` in the model, as in:

```
leuk.coxs <- coxph(Surv(time) ~ strata(ag) + log(wbc), data = leuk)
summary(leuk.coxs)
```

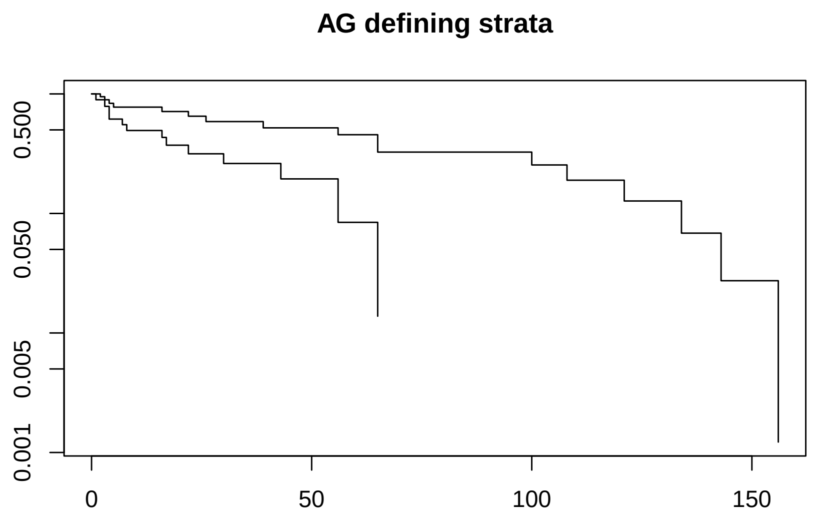
```
## Call:
## coxph(formula = Surv(time) ~ strata(ag) + log(wbc), data = leuk)
##
##   n= 33, number of events= 33
##
##            coef exp(coef) se(coef)     z Pr(>|z|)
## log(wbc) 0.3906    1.4778   0.1426 2.738  0.00618 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##          exp(coef) exp(-coef) lower .95 upper .95
## log(wbc)     1.478     0.6767     1.117     1.955
##
## Concordance= 0.688  (se = 0.067 )
## Likelihood ratio test= 7.78  on 1 df,   p=0.005
## Wald test            = 7.5  on 1 df,   p=0.006
## Score (logrank) test = 7.92  on 1 df,   p=0.005
```

Plots of the estimated survival distributions can be produced by using as input `survfit` applied to the fitted Cox model:

```
par(mfrow = c(1, 2))
plot(survfit(leuk.coxs), main = "AG as covariate")
plot(survfit(leuk.coxs), main = "AG defining strata")
```



```
plot(survfit(leuk.coxs), main = "AG defining strata", log=TRUE)
```



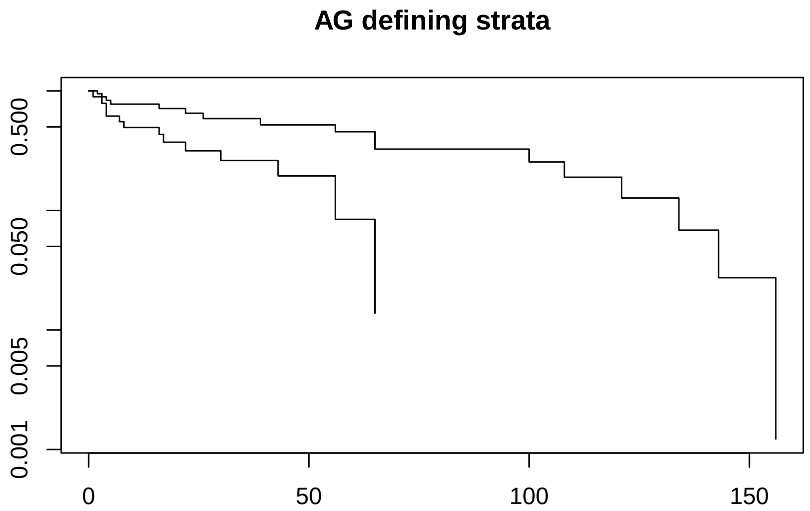
So, from the first model we obtained a single survival distribution, plotted making use of the reference or average value of each covariate. The stratified Cox regression plot yielded two survival distributions, one per stratum (AG group).

5.2.5 The proportional hazards assumption

By fitting a separate baseline hazard per group, we can compare the group-specific baseline hazards. This helps with checking the proportional-hazards assumption. Above we have made a graph of the separate hazards, on the scale of the survival probability. We can remake this plot on the scale of the log-probability. If the proportional hazards assumption holds, this should yield relatively parallel proportional hazards.

In the `leuk` example, this can be done as follows:

```
plot(survfit(leuk.coxs), main = "AG defining strata", log = TRUE)
```



Quick task(s):

Solve the task(s), and check your solution(s) here.

5.2.6 Others

types of censoring

competing risks

5.3 Survival data - power analysis

5.3.1 Motivation

Researchers want to start a study to find if progression is related to a chromosomal aberration sometimes found in cancer patients. Here we mean by *progression* the combination of time from end-of-treatment to relapse, as well as the event, which can be relapse or end-of-study, the latter representing censoring. In addition, individuals are recorded as having the aberration or not. From a previous study, progression data and information about the chromosomal aberration are available. Now they want to know how many patients are needed in their new study, designed to test if this chromosomal aberration affects progression or not.

In addition to the chromosomal aberration, other covariates will be taken into account when studying progression, such as age and BMI. For this reason, the researchers decided that they will use a Cox proportional-hazards model to analyse the data and test for the effect of the chromosomal aberration on progression to relapse. The aim of the study is to test whether or not prognosis is the same in the two groups, against the alternative hypothesis that prognosis is different between the two groups.

Since both pilot data are available and the analysis model has been chosen, a power analysis and/or sample size determination is possible.

5.3.2 Power and sample size for Cox regression

As for other types of power analysis and minimum sample size calculations, we can use a formula that relates power, significance level and some statistics from the pilot study to perform calculations. The formula is:

$$N = \frac{[z(1-\alpha/2) + z(\text{power})]^2}{p(1-p)\beta_1^2 P\{\text{event}\}}$$

where

- α is the desired significance level;
- the power is equal to $1 - \beta$, where β is the type-II error;
- $z(\alpha)$ is the quantile function corresponding to the standard normal distribution, yielding the value x that leaves α probability that a value is observed smaller than, or equal to, x, as implemented by `qnorm`;
- p represents the proportion of cases with the chromosomal aberration in the pilot study or, if known, in this population of patients;
- β_1 represents the log-hazard ratio for the chromosomal aberration effect estimated from a Cox proportional-hazards model for the pilot data;
- $P\{\text{event}\}$ represents the probability of events in the patient population, in this case estimated by the proportion of relapses (the event of interest);
- N is the sample size.

As before, we can input values for some of these values, and evaluate the remaining ones.

The above formula is built assuming that the test is double-sided, i.e. that the null hypothesis that prognosis does not differ between the two groups is tested against the alternative that the prognosis is different between the two groups. If the alternative is that prognosis is *better* in one group than the other, then the formula to be used is mostly the same, except that $1 - \alpha/2$ is replaced by $1 - \alpha$.

Note that the way the groups are defined (which group corresponds to the intercept) does not matter in this context. This is because the coefficient of the grouping variable β_1 is squared in the model, so its sign does not affect the results.

You do not have to learn this formula by heart. But we will practice using it, so that you better understand what the components mean, and how they can be used when designing experiments.

5.3.3 Working example

Let us consider again the Melanoma data. For these data, the event of interest is `status`=1, and all other events are considered together as censoring. We define below the `Surv` object and fit the Cox model, as before:

```
library(survival)
library(MASS)
str(Melanoma)
```

```
## 'data.frame':    205 obs. of  7 variables:
##  $ time     : int  10 30 35 99 185 204 210 232 232 279 ...
##  $ status   : int  3 3 2 3 1 1 1 3 1 1 ...
##  $ sex      : int  1 1 1 0 1 1 1 0 1 0 ...
##  $ age      : int  76 56 41 71 52 28 77 60 49 68 ...
##  $ year     : int  1972 1968 1977 1968 1965 1971 1972 1974 1968 1971 ...
##  $ thickness: num  6.76 0.65 1.34 2.9 12.08 ...
##  $ ulcer    : int  1 0 0 0 1 1 1 1 1 1 ...
```

```
survm <- Surv(Melanoma$time, Melanoma$status == 1)
mel.cox <- coxph(Surv(time, status == 1) ~ sex, data = Melanoma)
mel.cox
```

```
## Call:
## coxph(formula = Surv(time, status == 1) ~ sex, data = Melanoma)
##
##       coef exp(coef) se(coef)     z      p
## sex 0.6622    1.9390   0.2651 2.498 0.0125
##
## Likelihood ratio test=6.15  on 1 df, p=0.01314
## n= 205, number of events= 57
```

The coefficient for `sex`, corresponding to the log-hazard ratio, is statistically significant from 0 at the significance level of $\alpha = 0.05$, but not if we use $\alpha = 0.005$. Let us evaluate what the sample size should be to detect a similar effect size on the log-hazard ratio, if $\alpha = 0.005$ was to be used.

5.3.4 Minimum sample size

Below we write a function to compute the sample size, given the desired power to find a log-hazard ratio from a Cox proportional-hazards model fit. This is written with a binary grouping in mind.

```r
# get.ssize.surv
#
# Function to compute the sample size required to find a log-hazard ratio
#using Cox regression,
# where the ratio is computed between two groups
#
# Inputs
# beta: power, so 1-prob. type-II error
# alpha: desired significance level
# p1: proportion of individuals in group 1 - it does not matter which group
#is taken as group 1,
#     since this enters the formula via p1(1-p1)
# b1: log hazard ratio between the two groups
#   It corresponds to the beta coefficient in the cox ph regression
# pevents: prop events
#
# Output:
# the sample size required

get.ssize.surv <- function(beta, alpha=0.05, p1=0.5, b1=0.5, pevents=1)
{
  num <- ( qnorm(1-alpha/2,lower.tail=FALSE) + qnorm(beta, lower.tail=FALSE) ) ^2
  den <- p1*(1-p1)*( b1^2 )*pevents
  n <- num/den
  n
}
```

This function takes as arguments:

- `beta`: power, so 1-probability of the type-II error. This must be a value between 0 and 1;
- `alpha`: the desired significance level. This must be a value between 0 and 1;
- `p1`: proportion of individuals in group 1 - it does not matter which group is taken as group 1, since this enters the formula via p1(1-p1). This must be a value between 0 and 1;
- `b1`: log hazard ratio between the two groups. It corresponds to the beta coefficient computed by the Cox proportional-hazards regression.
- `pevents`: proportion of events in the data at hand.

5.3.5 Working example (cont)

Let us extract the quantities needed to perform a power calculation. The proportion of events is the proportion of individuals with `status`=1 in the data:

```
p.event <- mean(Melanoma$status == 1)
```

The log-hazard ratio estimated by the Cox model is stored in the slot `coefficients` which, in this case, has a single entry:

```
b1 <- mel.cox$coefficients
```

We also need to compute the proportion of individuals in group 1 (which is arbitrarily chosen). The grouping variable `sex` has values:

```
table(Melanoma$sex)
```

```
##
##   0   1
## 126  79
```

so its mean is equal to the proportion of individuals in group 1:

```
p1 <- mean(Melanoma$sex)
```

Now we can run compute the sample size required, for power = 0.8, say, to detect an effect with a significance level of 0.005.

```
get.ssize.surv(beta = 0.8, alpha = 0.005, p1 = p1, b1 = b1, pevents = p.event)
```

```
##      sex
## 461.0044
```

Quick task(s):

Solve the task(s), and check your solution(s) here.

5.3.6 Reference

Hsieh, F. Y., and P. W. Lavori. 2000. Sample-size calculations for the Cox proportional hazards regression modelwith nonbinary covariates.Controlled Clinical Trials21: 552–560. DOI: 10.1016/s0197-2456(00)00104-5

5.3.7 Notes

Formulae for power analysis and minimum sample size determination based upon the log-rank test also exist. However, these are applicable to the very specific case where a two-group comparison is made, and no covariates are used. For this reason, we have decided to present here the solution for the Cox proportional-hazard case only. We should note that the log-rank calculations are of course useful when proportional-hazards cannot be guaranteed, in which case analyses must rely on nonparametric tests such as the log-rank.

Formulae for this sort of problem are also available for more complex designs, such as those from clinical trials. Many of those, as well as the one for the log-rank test, have been implemented in R via the package `powerSurvEpi`. Their help file is available via the link below:

https://cran.r-project.org/web/packages/powerSurvEpi/powerSurvEpi.pdf

In addition, Stata has many of such formulae implemented. You may wish to check those, or the references in their documentation:

https://www.stata.com/manuals/pss-2powercox.pdf#pss-2powercoxhsiehlavori2000

If using Stata, please make sure you save the scripts used to running analyses!

5.4 Survival analysis - Notes

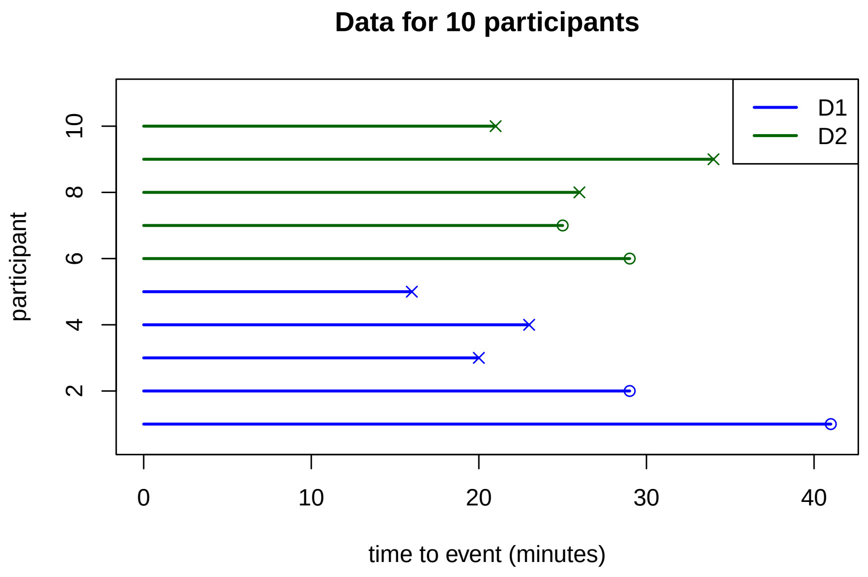
We have just seen an introduction to survival data analysis. It is good to be aware of the things that, in the interest of time, we did not see during this introduction.

5.4.1 Other types of censoring

5.4.2 Right-, left- and interval censoring

In our examples here we have had censored observations. These were observations from which we knew the starting points, but not the time to event - only that the event did not happen until a certain time point. Recall the graoh:

```r
set.seed(39562) # (493756)
ss <- 400
fdrug <- factor(rep(1:2, each = ss/2), labels = c("D1", "D2"))
event <- c(rbinom(ss/2, 1, prob = .7), rbinom(ss/2, 1, prob = .75))
tte <- round(rgamma(ss, shape = 30, rate = 1+event/4))
migraine <- data.frame(event, time = tte, drug = fdrug)
mydata <- rbind(migraine[1:5, ], migraine[(ss/2+1):(ss/2+5), ])
myxlim <- c(0, max(mydata$time))
mypch <- rep(4, nrow(mydata))
mypch[ mydata$event == 0 ] <- 1
plot(1, 1, xlim = myxlim, ylim = c(0.5, nrow(mydata)+1), col = "white",
     main = "Data for 10 participants", xlab = "time to event (minutes)",
     ylab = "participant")
mcol <- rep(c("blue", "darkgreen"), each = nrow(mydata)/2)
for(xi in 1:nrow(mydata)){
  segments(0, xi, mydata$time[xi], xi,  col = mcol[xi],
           lwd = 2)}
for(xi in 1:nrow(mydata)){
  points(mydata$time[xi], xi, col = mcol[xi], pch = mypch[ xi ])
  }
legend("topright", legend = c("D1", "D2"), lty = "solid", lwd = 2,
       col = c("blue", "darkgreen") )
```



This type of censoring is referred to as *right-censoring*, because in the graph the right-hand side of the follow-up is incomplete.

In other cases, the starting time is unknown. This can be the case when the patient enters the study some time after disease onset, and the exact date of disease onset is unknown. This type of censoring is referred to as *left-censoring*, because in the graph the left-hand side of the follow-up would be incomplete. Of course, there are studies where both types of censoring occur.

In R, `Surv` can be defined for left-, right- and interval censoring. Typically Cox regression can be used to analyse data with these different censoring types, so long as the censoring is independent of the event probabilities. However, the Kaplan-Meier method cannot be used directly with left-censored data.

5.4.3 Type I and type II censoring

Most progression studies record time to event, and censoring then occurs when either the study is completed before an individual has had an event, or else the individual has been lost to follow-up due to a reason unrelated to the event. In such cases, this is referred to as *type I* censoring. Kaplan-Meier and Cox regression can be used to analyse such data, as we have seen.

In some cases, a trial is stopped after a certain number of events is reached. As individuals who did not experience an event before the end of the trial are censored, these may correspond to those with longer time-to-event observations. So, the censoring process and the event probability are not independent. This is referred to as *type II* censoring. To analyse data from such studies other models are required than the ones seen here.

5.4.4 Competing risks

Let us look again at the Melanoma data:

```
# ?Melanoma
library(survival)
library(MASS)
str(Melanoma)
```

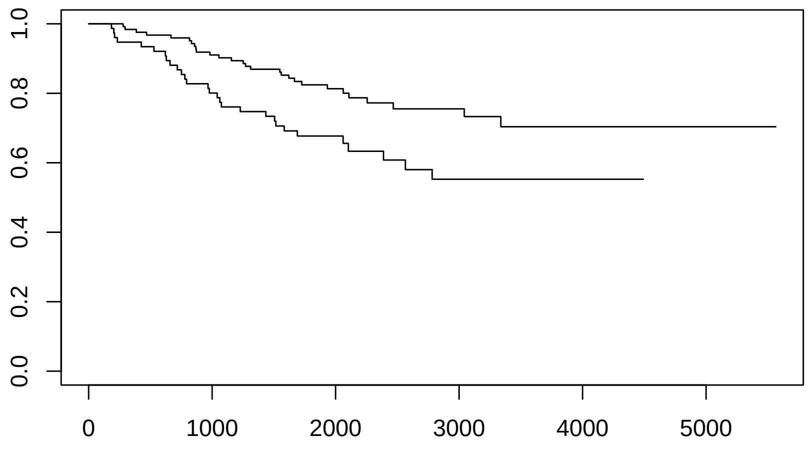
```
## 'data.frame':    205 obs. of  7 variables:
##  $ time     : int  10 30 35 99 185 204 210 232 232 279 ...
##  $ status   : int  3 3 2 3 1 1 1 3 1 1 ...
##  $ sex      : int  1 1 1 0 1 1 1 0 1 0 ...
##  $ age      : int  76 56 41 71 52 28 77 60 49 68 ...
##  $ year     : int  1972 1968 1977 1968 1965 1971 1972 1974 1968 1971 ...
##  $ thickness: num  6.76 0.65 1.34 2.9 12.08 ...
##  $ ulcer    : int  1 0 0 0 1 1 1 1 1 1 ...
```

In this example, `time` is the survival time in days, and `status` is an indicator of the patient's status by the end of the study:

- `status` = 1: "dead from malignant melanoma";
- `status` = 2: "alive on January 1, 1978";
- `status` = 3: dead from other causes"

We previously created a `Surv` object having only the event `dead from melanoma`, and both alive as well as dead due to other causes together as censoring. For this, we used:

```
survm <- Surv(Melanoma$time, Melanoma$status == 1)
plot(survfit(survm ~ Melanoma$sex))
```



```r
mel.cox <- coxph(Surv(time, status == 1) ~ sex, data = Melanoma)
summary(mel.cox)
```

```
## Call:
## coxph(formula = Surv(time, status == 1) ~ sex, data = Melanoma)
##
##   n= 205, number of events= 57
##
##       coef exp(coef) se(coef)     z Pr(>|z|)
## sex 0.6622    1.9390   0.2651 2.498   0.0125 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## sex     1.939     0.5157     1.153      3.26
##
## Concordance= 0.59  (se = 0.034 )
## Likelihood ratio test= 6.15  on 1 df,   p=0.01
## Wald test            = 6.24  on 1 df,   p=0.01
## Score (logrank) test = 6.47  on 1 df,   p=0.01
```

So, in this case the endpoint variable used in `Surv` is a logical vector, equal to `TRUE` if `status` = 1, and `FALSE` otherwise. However, by considering censoring together with death due to other causes, the probability of survival is biased.

Specifically, the assumption that observations are censored independently of the event probability is violated. This is because individuals who are censored are still at risk, whilst those who died due to other causes are not. As a result, the probability of an event becomes inflated. That is the case with both Kaplan-Meier as well as standard Cox regression

To analyse survival data that includes competing risks, typically people use the Fine and Gray model. There is a large body of research done in this area, so more information can be obtained from various good sources. Here I suggest the following article if you wish to read more about the problem and some solutions:

https://academic.oup.com/ndt/article/28/11/2670/1823847

The Fine and Gray model is available in R via the CRAN package `cmprsk`:

https://cran.r-project.org/web/packages/cmprsk/cmprsk.pdf

Chapter 6

Exercises

6.1 Chapter 1

6.1.1 Exercise 1

Load the dataset mtcars in the R memory using the command (datamtcars)

1.1 How many variables and observations are contained in the dataset? Print the 5 first rows of the dataset

1.2 Plot the histograms of the continuous numerical variables of the dataset (type `?mtcars` to obtain the description of the dataset).

1.3 We can clearly distinguish 3 right skewed distribution. Which are they?

1.4 The variables `vs` and `am` both split the data in 2 categories plot the boxplots of the variable `qsec` for each group define by `vs` and `am`:

1.5 Plot boxplots using the command `boxplot(qsec ~ vs:am)`. What is the difference with the plots obtained previously?

1.6 compute the mean and standard variation for each continuous variable of the dataset for the 2 different engine types.

6.1.2 Exercise 2:

We continue with the same dataset:

2.1 compute the correlation matrix associated for the three different correlations for all variables except the `vs` and `am` variables.

2.2 Display the results using heatmaps.

6.1.3 Exercise 3(*)

3.1 Perform a pca analysis and plot the resulting scree plot.

3.2 display the eigenvalues of the oca analysis. How many principal components would you keep?

3.3 Plot the correlation circle for the first two principal components. What is your interpretation of the first principal component?

3.4 Display the contribution of each variable on the principal components

6.2 Chapter 2

6.2.1 Exercise 1

Load the `ToothGrowth` dataset using the command line `data(ToothGrowth)`. This dataset is the result of an experiment on Guinea Pigs. It looks at the length of odontoblast cells after animals received a dose of vitamin C by one of two delivery methods. Namely, Orange juice (noted OJ) and ascorbic acid noted VC.

Source: Crampton, E. W. (1947). The growth of the odontoblast of the incisor teeth as a criterion of vitamin C intake of the guinea pig. The Journal of Nutrition, 33(5), 491–504. doi: 10.1093/jn/33.5.491.

1.1 Plot the histogram of the length of the odontoblast in the entire dataset.

1.2 Plot the boxplot of the length of the ordotonblast for each method of delivery of the Vitamin C. Compute the quartiles for both distributions.

1.3 Compute the mean absolute deviation in each group

1.4 Perform a Student't test to compare the mean of the length of the cells in both groups

1.5 Extract the value of the t statistics, the standard error and the degree of freedom from the test statistics. Display the results in the sentence:

The t statistics has a value of `r`, the estimated degrees of freedom of the Student's t distribution are `r`. The final p.value of the test is `r`.

1.6 Plot the distribution of the Student's t distribution under the null hypothesis

1.7 What are the values of the quantiles corresponding to boundaries between significane and non-significance?

1.8 what would be the number of Guinea Pigs per group required in order to have a power of 80% to detect a significant difference between both groups? And for a power of 90%?

1.9 Perform a Wilcoxon-Mann-Whitney test to compare the distribution of the length of the cells between both delivery methods

6.2.2 Exercise2

2.1 Create a plot that represent the power as a function of the sample for an experiment with an estimated difference in mean of 2, a standard deviation of 1.2, and a significance level of 0.05 of a two sided Student's t-test.

2.2 Create the same plot for a standard deviation of 2.4.

2.2 Create the same plot for a significance threshold of 0.1.

6.3 Chapter 3

6.3.1 Exercise 1

1.1 We go back to the `ToothGrowth` dataset. We saw that there is a significant difference in the average length of the odontoblast between the two delivery methods of vitamin C. We are now interested in the relationshipt between the dose of vitamin C and the length of odontoblast. Perform a linear regression model to test for association between these 2 variables.

1.2 Extract the estimates and corresponding statistics and p-values frol the regression

1.3 Create Plots in order to check the assumptions of the model

1.3 We can see that the residuals does not really follow a gaussian distribution. Use a log transformation on the length variable and compare the results

1.4 The transformation slightly improved the results. From the previous exercises we have observed an association between the delivery method and the length of the cells. Ass the delivery method in the linear regression:

1.5 It could be relevant to add a possible interaction between the dose given and the delivery method. Add an interaction effect in the model:

1.6 What do you conclude from this analysis?

6.3.2 Exercise 2

2.1. Return to the budworms example. Using the model fitted including the main effects of `sex` and `ldose` as well as their interaction, extract the fitted probabilities. Make a graph of observed and expected probabilities, and include a legend. Make sure to distinguish the groups, for example by using different plotting symbols for `sex` and different colours depending on the dose. Note that the observed probabilities are given by `numdead/20`.

6.3.3 Exercise 3

Analyse now the `birthwt` data from the package `MASS`, to investigate factors that may affect the chance of low birth weight.

3.1 Fit a logistic regression model to the variable representing low weight, using `smoke` as covariate in the model. Note that the response variable now is binary, not counts. Examine the model fit to check if it represents a good approximation of the data.

3.2 Extract fitted probabilities and plot those together with the observed data. Note that the latter is now simply a binary variable.

3.3 Fit now a logistic regression model to the variable representing low weight, using all covariates in the model. Examine the model fit to check if it represents a good approximation of the data. Extract fitted probabilities and plot those together with the observed data. Note that the latter is now simply a binary variable.

3.4 We think that the model can be improved further, but it is not immediately clear which variables should be included, and in which order. For this, we can use the function `step`, which performs stepwise regression. This involves comparing model fits by using the Akaike Information Criterion (AIC). A statistical test helps deciding whether the AIC value indicates a statistically significant improvement between model fits. Try to improve the basic model which includes `smoke` and `lwt`, by adding variables `age`, `ptl`, `ht` and `ftv` one at a time. Check the help file for `step`, and call the function using as `object` the fitted model with `smoke` and `lwt`, and as `scope` the formula for the model including these two, plus the extra covariates.

6.4 Chapter 4

6.4.1 Exercise 1

A study has looked at the relationship between aspirin use and heart attacks via a randomized clinical trial. The aim is to test whether aspirin taken regularly reduces mortality from cardiovascular disease. Study participants did not know if they used aspirin or a placebo. The table below summarizes some of their findings, according to fatal miocardial infarcts (FMI) and non-fatal ones (NFMI).

	FMI	NFMI	No attack
Placebo	18	171	10845
Aspirin	5	99	10933

Source: example 2.2.4 (pp. 16-17) of Agresti, A. (1990) Categorical data analsis. Wiley, New York.

1.1. Compute the proportions of heart attack depending on the drug used. Hint: After entering the data, add up the columns corresponding to heart attack and use `prop.table`. Save the result as an object. Write a small bit of text where you indicate the proportions found.

The study found that there were `r` of those using aspirin, and `r` of those using placebo.

1.2 Compute the relative risk of heart attacks in the placebo group, compared with the aspirin group.

1.3 Compute the sample odds ratio of having a heart attack in the placebo group.

1.4 Write a short piece of text where the computed relative risk and odds ratio are included, with inline code.

1.5 Test the hypothesis that the use of aspirin, compared to placebo, has no effect on the chance of a heart attack, against the hypothesis that aspirin modifies the chance, without making an assumption about in which direction the modification may occur.

1.6 Extract the odds ratio estimates from 1.3 and 1.6, and include them as inline code in the text below.

The odds ratio estimated by us from the table was `r`, and the one estimated by Fisher`s exact test wasr`.

6.4.2 Exercise 2

A study looked at the association between job satisfaction and income, gathering the data in the table below.

	Very dissatisf.	Little dissatisf.	Moderately satisf.	Very satisf.
`<6,000`	20	24	80	82
`6,000-15,000`	22	38	104	125
`15,000-25,000`	13	28	81	113
`>25,000`	7	18	54	92

Source: example 2.3.2 (pp. 20-21) of Agresti, A. (1990) Categorical data analsis. Wiley, New York.

2.1 Enter the data into R by making a matrix. Assign meaningful row and columns names. Print out the matrix to check that it has been entered correctly.

2.2 Add the counts from the dissatisfaction columns up, keeping the split between income levels. Do the same for the satisfaction columns. Make a new table (or a matrix) with these two new columns.

2.3 Use the chi-square test to compare the spread of people across income classes between satisfied and dissatisfied groups. Print out the test result and save the p-value in an object.

2.4 Complete the text below with the values of this example using inline code.

This study recorded job satisfaction from `r` people across `r` income classes. A chi-square test for the count distribution across income classes yielded a p-value of `r`.

2.5 Compute the total sample size that would be required, in order to find the observed effect size as significant with a significance level $\alpha = 0.05$ and power 0.8. Note that the number of degrees of freedom is equal to the number of rows in the table, minus 1.

2.6 Compute log-odds ratios for each income class, compared with the lowest income class.

2.7 Complete the following text with inline code.

This study found that odds of being dissatisfied, compared with being satisfied, is for the income class `>25,000` `r` that for the income class of `<6,000`.

6.4.3 Exercise 3

The table below gives the grade into which left and right eye of the same person were classified. Use an appropriate test to check if there is evidence that eye grades of one side are typically better than the other side, against the null hypothesis that the better grade is observed at random for the left and for the right eye.

	Left eye graded best	Left eye graded worst
Right eye graded best	3532	700
Right eye graded worst	597	2648

3.1 Enter the data as a matrix and define row and column names. Check that your definition is correct.

3.2 Use an appropriate test to check if there is evidence that eye grades of one side are typically better than the other side, against the null hypothesis that the better grade is observed at random for the left and for the right eye.

6.4.4 Exercise 4

A study compared radiation therapy with surgery in treating cancer of the larynx. The data is given below.

	Cancer controlled	Cancer not controlled
Surgery	21	2
Radiation therapy	15	3

4.1 Enter the data as a matrix, and define row as well as column names.

4.2 Use a test to check if there is evidence that the response (cancer controlled or not) is independent of therapy.

4.3 Compute the power of the test given the current data to find a difference between the two observed proportions.

4.4 Evaluate the power for larger sample sizes, keeping the relative sample size for the first and second groups the same. Make a graph of results, with the total expected sample size on the x-axis and the power on the y-axis.

6.5 Chapter 5

6.5.1 Exercise 1

Consider again the leukemia data in dataset `leuk`. We will now compare survival probabilities between the groups given by `ag`, representing the test result for the presence of Auer rods and/or significant granulation of leukaemic cells.

1.1 Plot the Kaplan-Meier survival curves separately per group defined by the AG test. Compute the log-rank test and add the test result to the graph's title.

1.2 By default the curves are plotted with the same colour. Add colours to the plot by using the slot `col` in the `plot` call. Note that curves are plotted using the order of the factor levels in the grouping variable, which is also the order used by `print` and `summary` - in this example D1 comes first, followed by D2. Add a legend to the plot indicating which group corresponds to which colour.

1.3 Note that in this case no confidence interval was displayed. Indeed, only when there is a single curve are confidence intervals displayed by default. Display confidence intervals in the plot, using the option `conf.int = TRUE`.

Colours defined for the groups we now used also for the confidence interval curves.

1.4 Use the log-rank test to compare the survival probabilities of the two groups defined by `ag`. Save the p-value corresponding to the test as an object.

1.5 Plot the Kaplan-Meier curves separately per group, and add the p-value to the plot title. Use different colours to display the curves of different groups.

1.6 Now fit a Cox regression model to the `Surv` response using `log(wbc)` as covariate, and another using both `log(wbc)` and `ag` as covariates. Save the fitted models as objects.

1.7 The model is fitted using methods similar to those used for generalized linear models. In particular, we can here also use ANOVA to compare models that are nested. Use `anova` to compare these two model fits. Can you conclude that the model with both `log(wbc)` and `ag` as covariates yields a better fit than the model with only `log(wbc)`?

1.8 Let us compare survival probabilities estimated by Kaplan-Meier with those estimated by the Cox model. Make the graph of the Kaplan-Meier curves separately per `ag` group, as well as of the Cox model's baseline hazards estimated separately, without correcting for `log(wbc)`. How do the survival probabilities estimated by Kaplan-Meier compare with those estimated by Cox regression?

1.9 Now repeat this plot, using the Cox model with separate baseline hazards depending on the group defined by `ag` as before, and correcting for the effect of `log(wbc)`. How do these results compare with those from the previous exercise?

6.5.2 Exercise 2

Read in the migraine data again. Check that the Kaplan-Meier curve produced by using the `survfit` object with just an intercept is the same as the one produced by plotting the `Surv` object directly.

6.5.3 Exercise 3

Researchers want to start a study to find if progression is related to a chromosomal aberration sometimes found in cancer patients. Here we mean by *progression* the combination of time from end-of-treatment to relapse, as well as the event, which can be relapse or end-of-study, the latter representing censoring. In addition, individuals are recorded as having the aberration or not. From a previous study, progression data and information about the chromosomal aberration are available. Now they want to know how many patients are needed in their new study, designed to test if this chromosomal aberration affects progression or not.

In addition to the chromosomal aberration, other covariates will be taken into account when studying progression, such as the study. For this reason, the researchers decided that they will use a Cox proportional-hazards model to analyse the data and test for the effect of the chromosomal aberration on progression to relapse.

3.1. Read the data in from file `data_progression_2groups.txt`.

3.2. Now explore the data. Check which variables it contains, compute the number of events equal to `yes`, and the proportion of them.

3.3. Create a numeric variable representing the event (=1 if an event ocurred, =0 otherwise). The number of days of progression-free survival is given in the variable `pfs`. Create a `Surv` object and make a Kaplan-Meier curve of the data.

3.4. Check how many cases there are in each group, and make separate Kaplan-Meier plots for the groups.

3.5 Perform a log-rank test to compare the progression data between the groups.

3.6 Now fit a Cox proportional-hazards model to estimate the difference in log-hazards between the two groups. Check if there is evidence that the proportional hazards assumption holds.

3.7 Now use these results to determine how many samples would be required, in order to have the power of 0.80 to find an effect of the same size of the current effect, with a significance level of 0.01. Do not forget to load the function to estimate the sample size using the available parameters.

3.8 Now use these results to determine how many samples would be required, in order to have the power of 0.80 to find an effect of half the size of the current effect, with a significance level of 0.01.

3.9 Make a graph of the sample size necessary to find an effect size of values between the current log-hazard ratio and half of its size, using all other parameters the same as before.

Chapter 7

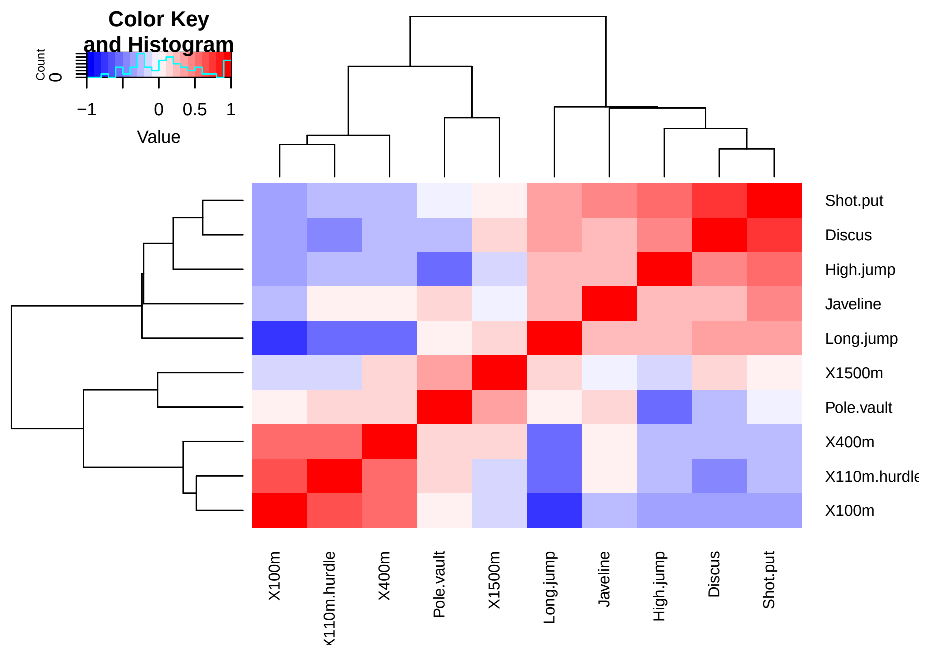
Appendix

7.1 PCA Analysis

7.1.1 Motivation

After having an overview of the relationship between the variables of the dataset `decathlon2`, the searcher wants to extract more information and a better visualization from this correlation structure for both samples and variables.

```
library(FactoMineR)
library(gplots)
library(factoextra)

data(decathlon2)



data.events <- decathlon2[,c(1:10)]
correlation.pearson <- cor(data.events, method = 'pearson', use = 'complete.obs')
heatmap.2(correlation.pearson, trace = 'none', cexRow = 0.8, cexCol = 0.8,
          col= colorRampPalette(c("blue","white","red"))(20))
```



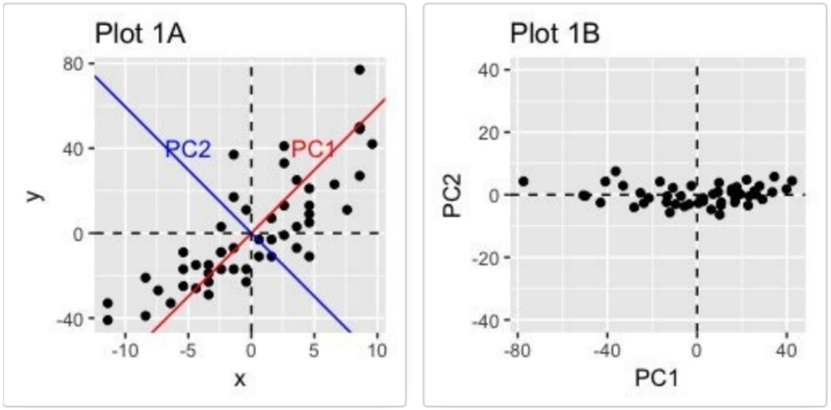
To do so we will use the principal components analysis commonly called PCA.

7.1.2 Working example

We will use the example that we created in the previous section on correlation:

```
library(mvtnorm)
set.seed(352)
sample.size <- 60

correlations <- runif(45, -1, 1)
correlation.matrix <- matrix(0, 10, 10)
correlation.matrix[upper.tri(correlation.matrix, diag=FALSE)] <-
  correlations
correlation.matrix[lower.tri(correlation.matrix, diag=FALSE)]  <-
  t(correlation.matrix)[lower.tri(correlation.matrix)]
diag(correlation.matrix) <- 1
standard.deviations <- rep(1,10)
covariance.matrix<- diag(standard.deviations) %*%
  correlation.matrix %*%
  diag(standard.deviations)

dataset<-as.data.frame(rmvnorm( sample.size, mean=rep(0,10), covariance.matrix))
colnames(dataset) <- paste('Variable', c(1:10), sep='')
```

The principle of principal components analysis relies heavily on mathematical algebra that we won't detail in this course. The main idea of principal components analysis is to identify new variables carrying most of the variance existing in your dataset. These new variables are created by linear combinations of the variables available in the dataset and can be used to identify hidden patterns, dimension reduction and the correlated variables in your dataset.



In this example we can see that by redefining the variables and using the principal component 1 (a linear combination of variable x and y) we capture most of the variation.

The principal component analysis consist of finding the principal components and by looking at contribution of each variables on this principal components understanding the structure of the data.

We will perform a principal component analysis on the dataset we simulated. This can be done by using the function PCA of the package `FactoMineR`.

Note that to perform a PCA, it is important to scale the data. Indeed, the PCA results can be strongly influenced by different variances.

```
library(FactoMineR)
pca.analysis <- PCA(dataset, scale.unit = TRUE, graph = FALSE)
```

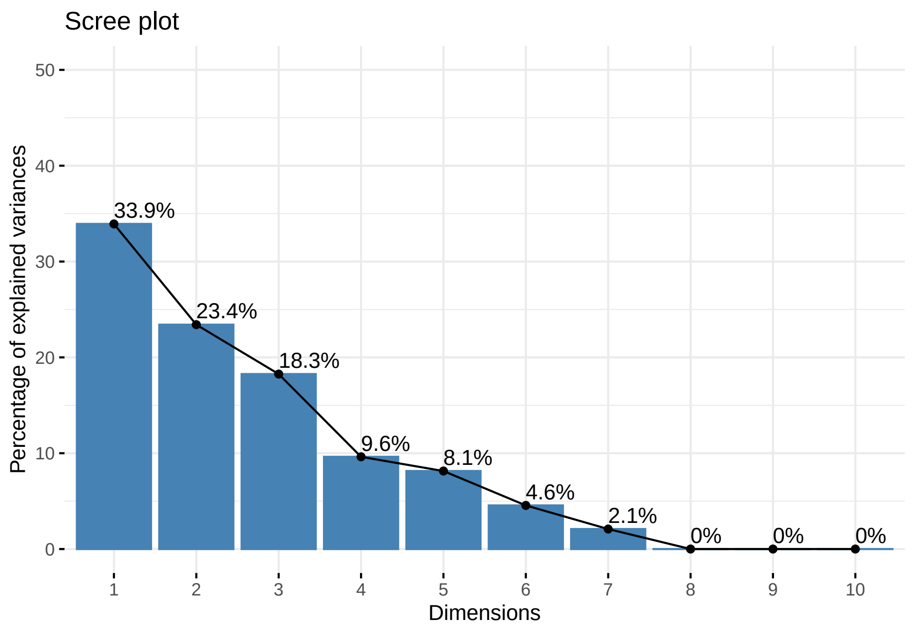
Typically there is a number of principal components equal to the number of variables in your dataset. However, not all of them retain the same amount of variance. The amount of variation retained by each component is called eigenvalue and can help us to determine the number of principal components needed to describe properly the data. By definition the principal component 1 is the one retaining most variance.

```
eigenvalues <- get_eigenvalue(pca.analysis)
eigenvalues
```

```
##           eigenvalue variance.percent cumulative.variance.percent
## Dim.1   3.392355e+00     3.392355e+01                    33.92355
## Dim.2  2.341258e+00     2.341258e+01                        57.33613
## Dim.3  1.825827e+00     1.825827e+01                        75.59440
## Dim.4  9.626093e-01     9.626093e+00                        85.22049
## Dim.5  8.140583e-01     8.140583e+00                        93.36108
## Dim.6  4.551795e-01     4.551795e+00                        97.91287
## Dim.7  2.087128e-01     2.087128e+00                       100.00000
## Dim.8  1.114076e-31     1.114076e-30                       100.00000
## Dim.9  6.117524e-32     6.117524e-31                       100.00000
## Dim.10 4.155684e-32     4.155684e-31                       100.00000
```

We see that only 5 principal components are needed to capture 93.36% of the total variance present in the dataset. This can be better seen via the scree plot:

```
fviz_eig(pca.analysis, addlabels = TRUE, ylim = c(0, 50))
```



There is unfortunately, no consensus on the exact number of principal components to keep. 3 possibilities can be used to select the number of principal components. The first one is to select the number of principal components to retain a sufficient amount of variance present in the dataset such as 80%. In our case 4 principal components

The second approach is to look at the scree plot to determine the number of components. The number of component is determined at the point, beyond which the remaining eigenvalues are all relatively small and of comparable size. In our case 4 principal components.

Finally the third approach is to look at the eigenvalues of the different principal components and to keep the principal components having a eigenvalue higher than 1. Indeed an eigenvalue higher than 1 indicates that the related principal components have more variance than one of the original variables. In our case 4 principal components.

Now that we have obtain the number of principal components to keep, we can use the principal component analysis as a dimension reduction approach and keep these new variables for statistical models. But we can also investigate the contribution of these variables in each components in order to understand better the relationship between our variables in the dataset.

To do so we need to extract the results for the variables from our PCA.

```
pca.analysis <- PCA(dataset, ncp=4, scale.unit = TRUE, graph = FALSE)
variable.analysis <- get_pca_var(pca.analysis)
variable.analysis
```

```
## Principal Component Analysis Results for variables
##  ===================================================
##   Name       Description
## 1 "$coord"   "Coordinates for the variables"
## 2 "$cor"     "Correlations between variables and dimensions"
## 3 "$cos2"    "Cos2 for the variables"
## 4 "$contrib" "contributions of the variables"
```

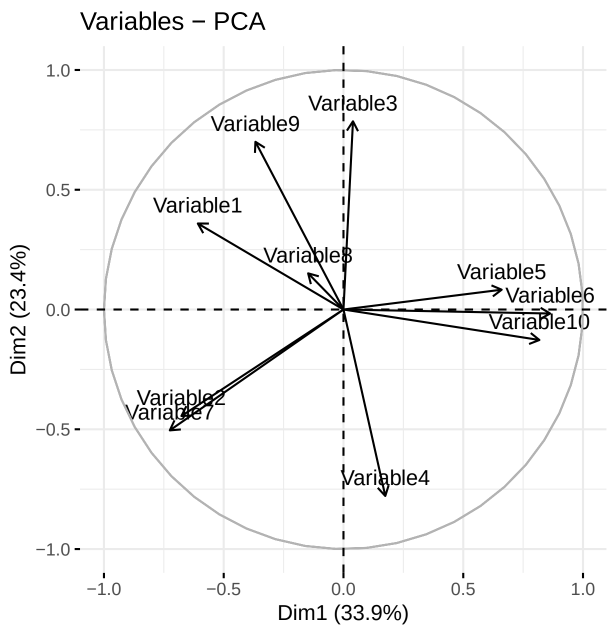
```
head(variable.analysis$coord, 4)
```

```
##                 Dim.1      Dim.2       Dim.3      Dim.4
## Variable1 -0.60818282  0.3592227  0.47779077  0.3806998
## Variable2 -0.67661752 -0.4446899  0.28175127 -0.4878183
## Variable3  0.03963396  0.7858208 -0.10046524  0.3549273
## Variable4  0.17456857 -0.7784345 -0.03390743  0.4826220
```

The different available results are the coordinates of the variables in the new dimension (important to compute the values of the observation for the new variables) and the contribution of each variable to the different principal components.

With this information, we can create plots to understand the relationship between the variables and the principal components such as the correlation circle.

```
#Plot correlation circle
fviz_pca_var(pca.analysis, col.var = "black", axes = c(1,2))
```



This plot shows the relationship existing between the variables. Positively correlated variables will be grouped together whereas negatively correlated variables will be on the opposite side of the plot origin. The closest the variable is from an axis, the more correlated this variable is with the principal component. Finally, the longer the arrow is the better represented this variable is by the two principal components.

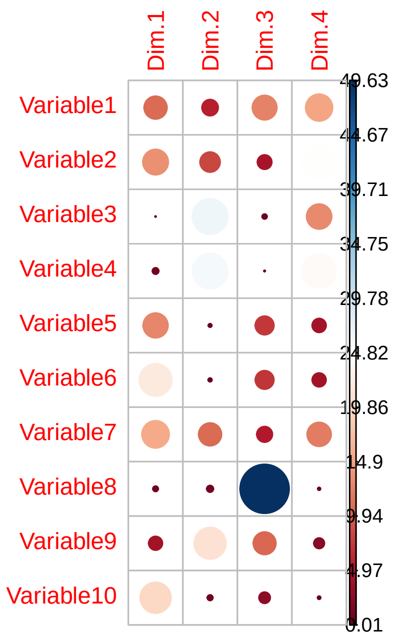
We can see also by plot the amount of contribution that each variable provide to each principal component:

```
library("corrplot")
```

```
## Warning: le package 'corrplot' a été compilé avec la version R 4.1.1
```

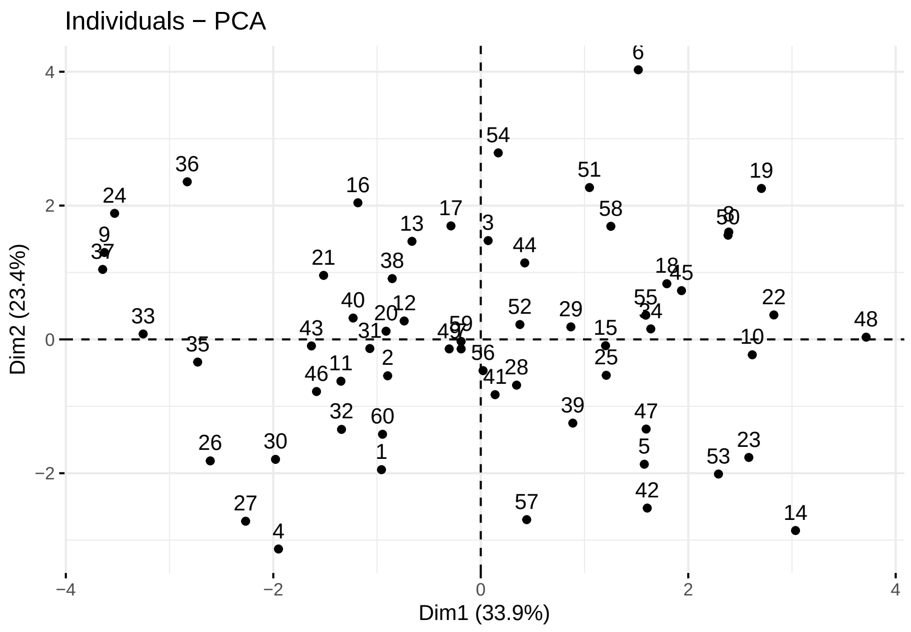
```
## corrplot 0.90 loaded
```

```
corrplot(variable.analysis$contrib, is.corr=FALSE)
```



Mow that we looked at the relationship existing between the variables and the principal components we can do the same thing for the individuals.

```r
#Scatterplot in the 2 first principal components plane
fviz_pca_ind(pca.analysis, col.var = "black", axes = c(1,2))
```



By looking at the coordinates of the individual in the new space we can understand how these samples are clustered. By then looking at the contribution of each variable for the principal components we can then derive conclusion on the relatedness of the individuals based on the original variables.

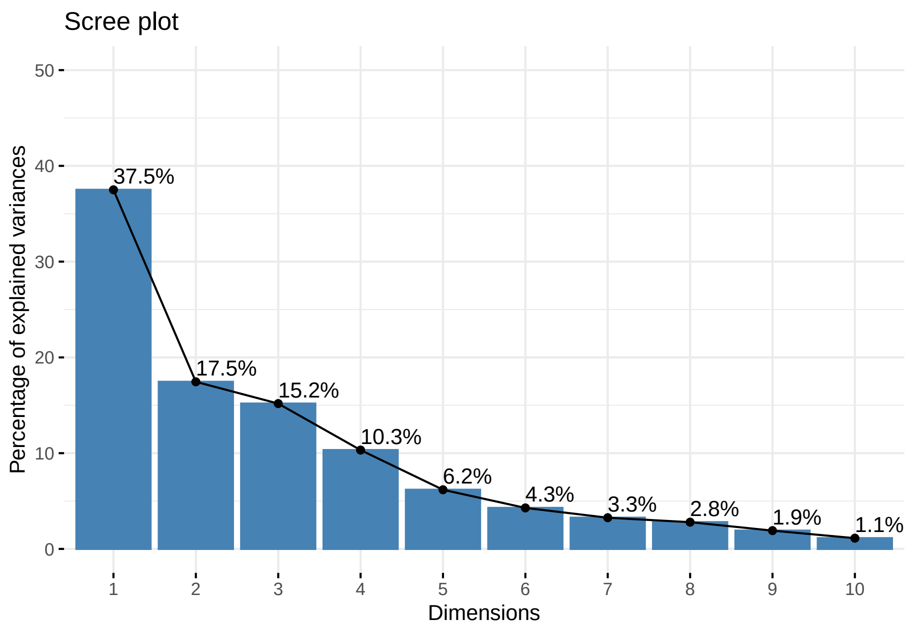
7.1.3 Quick tasks

Perform a principal component analysis on the events of the `decathlon2` dataset. Print the eigenvalues table and make a scree plot to determine the number of principal components to retain.

```
library(factoextra)
library(FactoMineR)
data.events <- decathlon2[,c(1:10)]
pca.analysis.deca <- PCA(data.events, scale.unit = TRUE, graph = FALSE)
eigenvalues <- get_eigenvalue(pca.analysis.deca)
eigenvalues
```

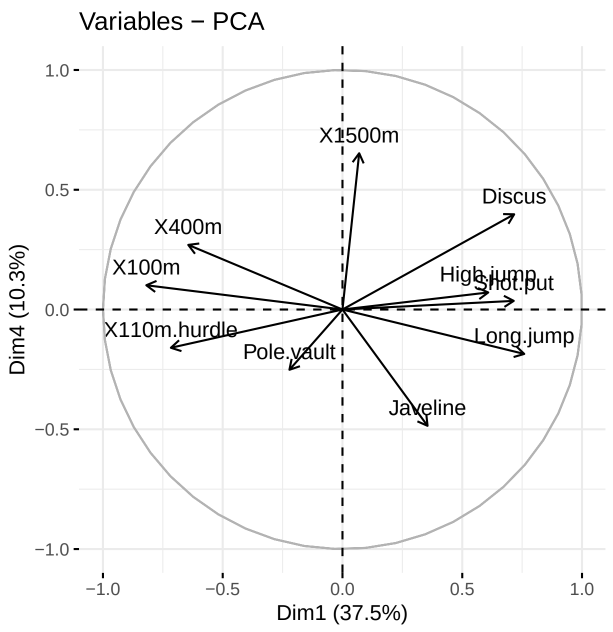
```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1   3.7499727        37.499727                    37.49973
## Dim.2   1.7451681        17.451681                    54.95141
## Dim.3   1.5178280        15.178280                    70.12969
## Dim.4   1.0322001        10.322001                    80.45169
## Dim.5   0.6178387         6.178387                    86.63008
## Dim.6   0.4282908         4.282908                    90.91298
## Dim.7   0.3259103         3.259103                    94.17209
## Dim.8   0.2793827         2.793827                    96.96591
## Dim.9   0.1911128         1.911128                    98.87704
## Dim.10  0.1122959         1.122959                   100.00000
```

```
fviz_eig(pca.analysis.deca, addlabels = TRUE, ylim = c(0, 50))
```

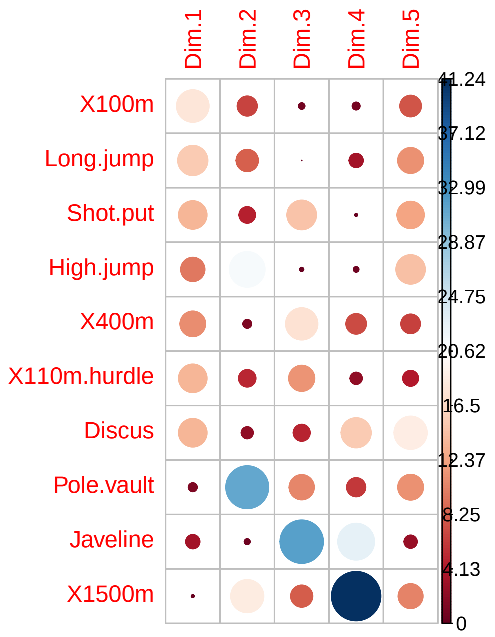


Plot the correlation circle of the first two principal components as well as the contribution plot

```
pca.analysis.deca <- PCA(data.events, ncp=5, scale.unit = TRUE, graph = FALSE)
variable.analysis.deca <- get_pca_var(pca.analysis.deca)
fviz_pca_var(pca.analysis.deca, col.var = "black", axes = c(1,4))
```



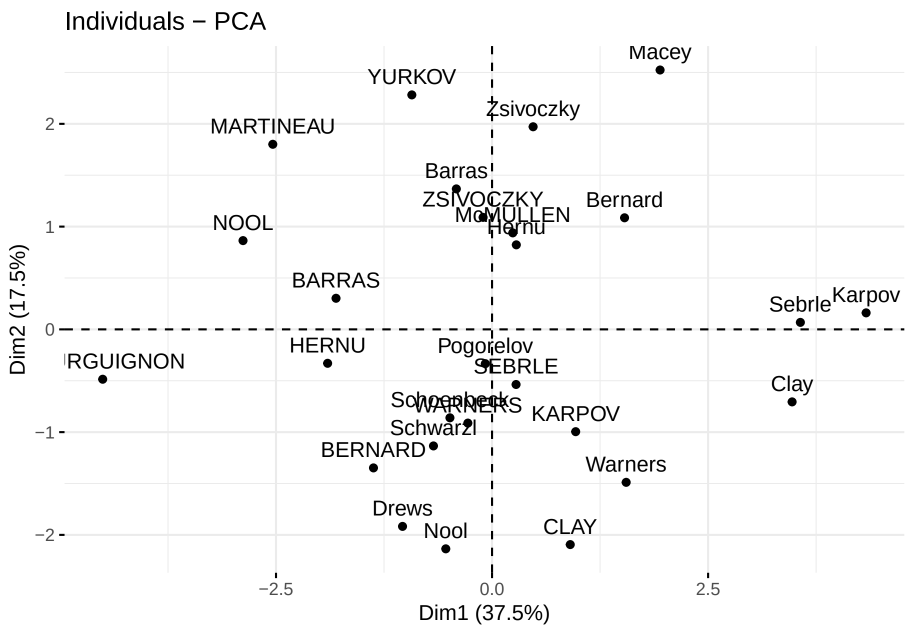
```
corrplot(variable.analysis.deca$contrib, is.corr=FALSE)
```



What can you conclude from the first two principal components? What are they mostly measuring?

Plot the representation of the individuals on these two principal components

```
fviz_pca_ind(pca.analysis.deca, col.var = "black", axes = c(1,2))
```


Individuals – PCA
Macey
YURKOV
Zsivoczky
MARTINEAU
Barras
ZSIVOCZKY
McMULLEN
Bernard
Hernu
NOOL
BARRAS
Sebrle
Karpov
RGUIGNON
HERNU
Pogorelov
SEBRLE
Clay
Schoenbeck
WARNERS
KARPOV
Schwarzl
BERNARD
Warners
Drews
Nool
CLAY
Dim2 (17.5%)
Dim1 (37.5%)