Intro to R Programming: Lab 1

More on computers' mistakes

1. You are probably aware that there are two formulae for computing the variance of a sample:

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \bar{x}^2$$

   When performing calculations by hand you would probably use the right hand one. Which one should we use on a computer? We will simulate data with a large mean and a very small variance.

```
n <- 1000                              # Set sample size.
mu <- 1e7                              # Set mean to something very large.
sigma <- 1e-1                          # Set standard deviation.
x <- rnorm(n,mu,sigma)                 # Simulate the data.
```

The following code implements the two formulae above

```
sum((x-mean(x))^2)/n                   # Formula on left hand side.
sum(x^2)/n - mean(x)^2                 # Formula on right hand side.
```

Are the results equal? Compare these with the R built-in function `var`.

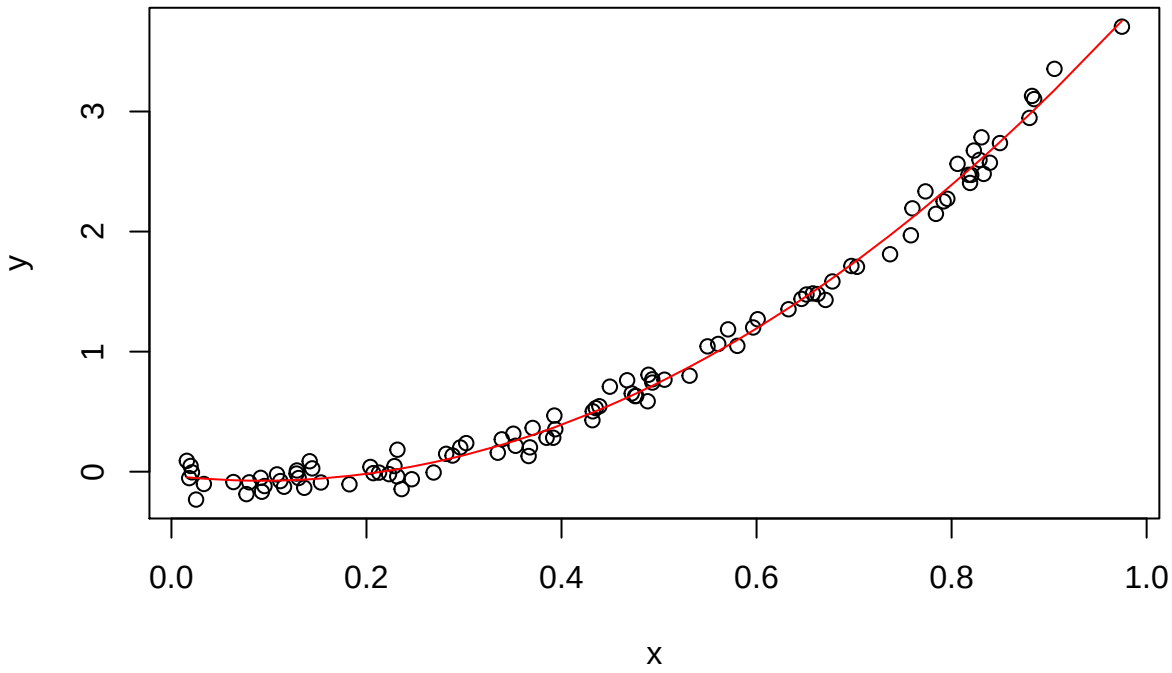
```
var(x)                                 # Use the built-in function.
```

2. Consider the following quadratic regression problem. We wish to use a model of the type

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2.$$

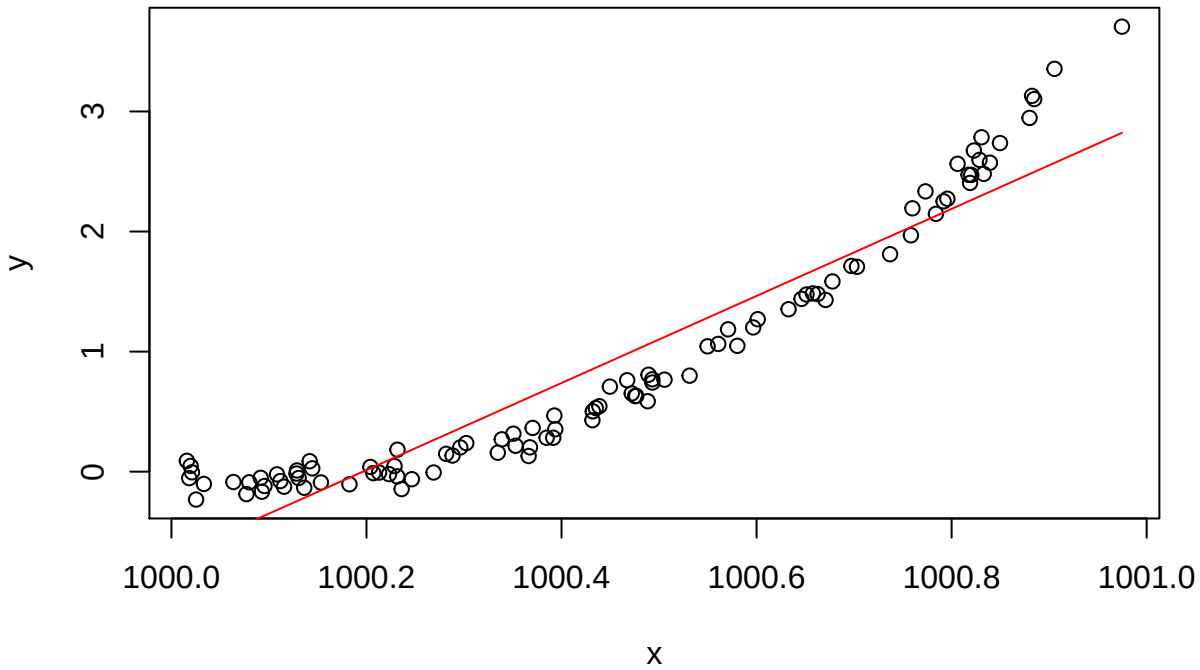
   Use the following code to fit a quadratic regression to the simulated data.

```
n <- 100
x <- sort(runif(n))                    # Create 100 values between 0 and 1.
y <- 5*x^2 - x + 0.1*rnorm(n)          # Create simulated response.
plot(x,y)                              # Plot the data.
reg.model <- lm(y~x+I(x^2))            # Fit the regression model.
lines(x,fitted(reg.model),col="red")   # Plot the fitted function.
```



What if we change the range of x from $[0, 1]$ to $[1000, 1001]$? According to the theory of the linear model the fitted values should be exactly the same. Try using the following code.

```
x <- x+1000                          # Add 1000 to each x.
plot(x,y)                            # Plot the data.
reg.model <- lm(y~x+I(x^2))          # Fit the regression model.
lines(x,fitted(reg.model),col="red")  # Plot the fitted function.
```



You can see that we run into numerical difficulties, as $X^{\mathrm{T}}X$ is almost a singular matrix, so computing the estimated regression coefficients $\hat{\beta}$ fails.

R as a calculator

3. Use R to compute $5 - \frac{7}{8}$, $\frac{5-7}{8}$, $\pi - \frac{333}{123}$, $256^{1/4}$, $\exp(1)$, $\left(1 + \frac{1}{1000}\right)^{1000}$, $\left(\frac{1}{56}\right)^{\frac{1}{4}}$.

4. The "Babylonian method" provides a way of approximating $\sqrt{2}$. The sequence defined recursively by

$$x_n = \frac{x_{n-1}}{2} + \frac{1}{x_{n-1}}$$

tends to $\sqrt{2}$ as $n \longrightarrow \infty$ (provided $x_0 > 0$).

- Define a new variable `x` taking the value 1.
- Update `x` to take the value

$$\frac{x}{2} + \frac{1}{x}.$$

- Repeat the update from the previous part. You should see that x tends to $\sqrt{2}$. (You will learn later on in this course how to use loops to do this more efficiently. If you already know how to use loops, feel free to use them in this question.)

Logical Variables

5. Consider two logical variables `a` and `b` generated using the code below

```
a <- sample(c(TRUE, FALSE), 1)
b <- sample(c(TRUE, FALSE), 1)
```

which randomly sets each of them to either `TRUE` or `FALSE`. Use the operators `&`, `|`, `!` and/or `==` to define a new variable `c` in terms of `a` and `b`, which is `TRUE` if `a` and `b` are either both `TRUE` or both `FALSE`. Otherwise `c` should be `FALSE`.

Can you think of more than one way of defining `c`?

Intro to R Programming: Lab 2

Scalar summary functions

Task 1 Create a vector `x` using the command

```
x <- seq(1, 5, by=0.3)
```

1. Use the commands `mean` and `sd` to compute the mean and standard deviation of `x`.
2. Standardise `x` by subtracting the mean and dividing by the standard deviation. Store this result in a vector called `x.standardised`.
3. Check that the mean and standard deviation of `x.standardised` are 0 and 1, respectively.

Task 2 Create a vector of length 100 from the $\mathsf{N}(1,1)$ distribution using

```
x <- rnorm(100, mean=1, sd=1)
```

and compute the test statistic of the one-sample t-test for the null hypothesis that $\mu = 0$:

$$t = \sqrt{n}\frac{\bar{x}}{\sqrt{s_x^2}}, \qquad \text{with } s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Task 3 Use R to calculate (or approximate the following sums and products)

$$\sum_{i=0}^{9} 2^i, \text{ and } \sum_{i=1}^{\infty}\frac{1}{2^i}$$

Task 4 Use the code

```
x <- rnorm(100)
```

to generate a sample of size 100 from the $\mathsf{N}(0,1)$ distribution.

1. Compute the mean and the median of `x`. Re-run the above line of code and re-compute the mean and the median five times. What can you observe?
2. Rather than generating a sample of size 100 from the standard normal distribution we will now generate a sample from the Cauchy distribution using

```
x <- rcauchy(100)
```

At first sight the Cauchy distribution is just like standard normal distribution symmetric around 0, i.e. it is centred around 0. Compute the mean and the median of `x`. Re-run the above line of code and re-compute the mean five times. What can you observe? Try to explain why this happens.

3. One way of "robustifying" the mean is to compute the so-called 20%-trimmed mean by removing the 10 smallest and the 10 largest observations and computing the mean of the remaining 80 observations. Compute the 20%-trimmed mean of both a sample from the standard normal distribution and the Cauchy distribution. Your code should automatically find the 10 smallest and largest values using the function `order` or `sort`.

Vectors and sequences

Task 5 Create a vector `u` containing all integers between 1 and 100. Replace all elements from `u` that are less than 55 by the number 0.

Task 6 Explain what values the logical vectors `c` and `d` will take before running the code in R.

```
a <- c(TRUE,FALSE)
b <- c(FALSE,FALSE)
c <- (a & !b)
d <- !(a | b)
```

Task 7 Create the following patterned sequences as efficiently as possible.

```
1 2 3 4 5 ... 10
1 2 3 1 2 3 1 2 3 1 2 3
1 1 2 2 3 3
1 2 3 4 ... 18 19 20 19 18 ... 4 3 2 1
1 4 9 16 25 36 49 64 81 100
1 2 4 8 16 32 64 128 256 512 1024
```

Matrices

Task 8 Create the following matrix and store it as `P`.

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

1. Print the first row and the second column of `P`.
2. Print the submatrix that consists of the first three rows and first two columns of `P`.
3. Compute the transpose and inverse of `P`.
4. Replace the first row of `P` by $(1, 2, 3, 4, 5)$.
5. Replace all non-zero entries of the matrix `P` by `1`.

Task 9 Matrix multiplication is associative, i.e. for two matrices $\mathbf{A}$ and $\mathbf{B}$ and a vector $\mathbf{x}$ we have that $(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{x} = \mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{x})$.

Create matrices `A` and `B`, and a vector `x` using the commands

```
A <- matrix(rnorm(9e6), nrow=3e3)     # Create matrix A with random entries (3000x3000)
B <- matrix(rnorm(9e6), nrow=3e3)     # Create matrix B with random entries (3000x3000)
x <- rnorm(3e3)                       # Create vector x with random entries (3000x1)
```

Because of the associativity of matrix multiplication both

`(A%*%B)%*%x`

and

`A%*%(B%*%x)`

give, except for rounding errors, the same answer. However the first command is much slower than the second command (3 seconds and 0.1 seconds on my computer), though both give the same answer. Explain why.

Task 10 Define a (symmetric and positive definite) matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 20 & 26 \\ 3 & 26 & 70 \end{bmatrix}$$

and a vector $\mathbf{b} = [4, 52, 31]$.

1. Use `solve` to compute $\mathbf{z} = \mathbf{A}^{-1}\mathbf{b}$, which is the solution to the system of equations $\mathbf{Az} = \mathbf{b}$:

$$\begin{array}{rcrcrcr} z_1 & + & 2z_2 & + & 3z_3 & = & 4 \\ 2z_2 & + & 20z_2 & + & 26z_3 & = & 52 \\ 3z_1 & + & 26z_2 & + & 70z_3 & = & 31 \end{array}$$

2. Use `chol` and `t` to compute the Choleski factor $\mathbf{L}$.
3. Verify that $\mathbf{LL}^{\mathrm{T}}$ is (except for rounding errors) identical to $\mathbf{A}$.
4. We can now exploit that $\mathbf{z} = \mathbf{A}^{-1}\mathbf{b} = (\mathbf{LL}^{\mathrm{T}})^{-1}\mathbf{b} = \mathbf{L}^{-\top}\underbrace{\mathbf{L}^{-1}\mathbf{b}}_{=\mathbf{v}}$ and use this so calculate $\mathbf{z}$.
5. Compute $\mathbf{v} = \mathbf{L}^{-1}\mathbf{b}$ using the function `solve`. Because of the lower-triangular structure of $\mathbf{L}$, this calculation can be done more efficiently using forwardsolving, `forwardsolve(L,b)`.
6. Compute $\mathbf{z} = \mathbf{L}^{-\top}\mathbf{v}$ using the functions `solve` and `t`. Because of the upper-triangular structure of $\mathbf{L}^{\top}$, this calculation can be done more efficiently using backsolving, `backsolve(t(L),v)`.
7. Compute the square of the product of the diagonal entries of $\mathbf{L}$ and compare it to the determinant of $\mathbf{A}$, which you can calculate using the function `det(A)`.

Intro to R Programming: Lab 3

All data files for this lab need to be downloaded from the module's Moodle page first.

Task 1

Read the data files `health.txt` and `cia.csv` into R. Make sure missing values are read in correctly.

Task 2

Use the function `write.table` to save the data frame `health` you created in the first task as a comma separated file usign column names, but no row names. Use * for missing values. Onve you have created the file, open it in an editor or Excel and check its content.

Task 3

In this task you will learn about the function `cut`, which can be used to discretise a numerical variable.

For instance, consider a variable `speed` created using

```
speed <- c(34, 49, 52, 24, 60, 74, 55)
```

Suppose we want to turn this continuous variable into a discrete variable with only three categories. We can do so using the function `cut`:

```
speed.discretised <- cut(speed, breaks=3)
speed.discretised
```

```
## [1] (23.9,40.7] (40.7,57.3] (40.7,57.3] (23.9,40.7] (57.3,74]   (57.3,74]
## [7] (40.7,57.3]
## Levels: (23.9,40.7] (40.7,57.3] (57.3,74]
```

If we give no additional arguments to `cut`, it tries to determine the cutoff points automatically. You can also set these and the corresponding labels manually:

```
speed.discretised <- cut(speed, breaks=c(0,40,60,100),
                         labels=c("slow", "medium", "fast"))
speed.discretised
```

```
## [1] slow   medium medium slow   medium fast   medium
## Levels: slow medium fast
```

Speeds > 0 and ≤ 40 are classed as `slow`, speeds > 40 and ≤ 60 are classed as `medium`, etc.

Use what you have learned in the example to add a new column called `ExpectancyGroup` that takes the values `low` (`LifeExpectancy`≤ 40), `medium` ($40 <$`LifeExpectancy`≤ 70), and `high` (`LifeExpectancy`> 70) to the data frame `health` from the previous task.

Task 4

The file `maternity.csv` contains data (obtained from `data.gov.uk`) on the number of pregnant women who were still smoking at delivery for each Health Authority in England in the first quarter 2011. The data also contains the number of mothers breastfeeding when their baby is 6–8 weeks old. The file contains the following variables:

`HealthAuthority`	Name of the Health Authority (Primary Care Trust)
`Region`	Name of the Region (Strategic Health Authority)
`Deprivation`	Average deprivation score
`Maternities`	Number of maternities
`Smoking`	– with the mother smoking at delivery
`SmokingUnknown`	– for which the smoking status of the mother could not be determined
`Breastfeeding`	– with the mother breastfeeding at 6 weeks
`BreastfeedingUnknown`	– for which the breastfeeding status of the mother could not be determined

Use R to determine ...

1. for each Health Authority the proportion of smoking pregnant women and the proportion of breastfeeding mothers;
2. the name of the Health Authority which both the smallest and the largest proportion of smoking pregnant women / breastfeeding mothers;
3. the percentage of smoking pregnant women / breastfeeding mothers in the North West and in London;
4. the percentage of smoking pregnant women / breastfeeding mothers for Health Authorities with an average deprivation score of at most 10 and at least 40; and
5. the percentage of breastfeeding mothers for Health Authorities with more than 25% (and less than 15%) smoking pregnant women.

Task 5

In this task you we will implement simple linear regression, both using the "sum formulae" and using matrix algebra. Load in the alligator.csv file from moodle.

`Length`	Snout vent length in inches (distance between back of head and end of nose)
`Weight`	Weight of the animal in pounds

1. Create two vectors `x` and `y`, such that `x` contains the logarithm of the snout vent length (covariate) and `y` contains the logarithm of the weight (response).
2. In the linear regression model $E(y_i) = \beta_0 + \beta_1 x_i$ with one covariate $\mathbf{x} = (x_1, \ldots, x_n)$ and response $\mathbf{y} = (y_1, \ldots, y_n)$ the least-squares estimate of the regression coefficient $\boldsymbol{\beta} = (\beta_0, \beta_1)$ can be found using the formulae

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \cdot \bar{x}.$$

2.1. Compute the least-squares estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ using the above formulae.

2.2. Create a vector `y.hat` that contains the predictions $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$.

2.3. The estimate of the variance of the residuals is $\hat{\sigma}^2 = \dfrac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n-2}$. Use R to compute this estimate.

2.4. Use R to compute the coefficient of determination: $R^2 = 1 - \dfrac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$

3. In this part you will compute the least-squares estimate using matrix algebra:

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}.$$

by solving the system of equations

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$$

3.1. Create the design matrix

$$\mathbf{X} = (\mathbf{1}\ \mathbf{x}) = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}.$$

3.2. Create a matrix `XtX` which holds $\mathbf{X}'\mathbf{X}$ and a vector `Xty` which holds $\mathbf{X}'\mathbf{y}$.

3.3. You can now solve the above system of equations by solving $\texttt{XtX}\hat{\boldsymbol{\beta}} = \texttt{Xty}$.

3.4 Compute the fitted values $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ and store them in a vector `y.hat`.

Parts (b) and (c) should give you the same regression coefficients and fitted values.

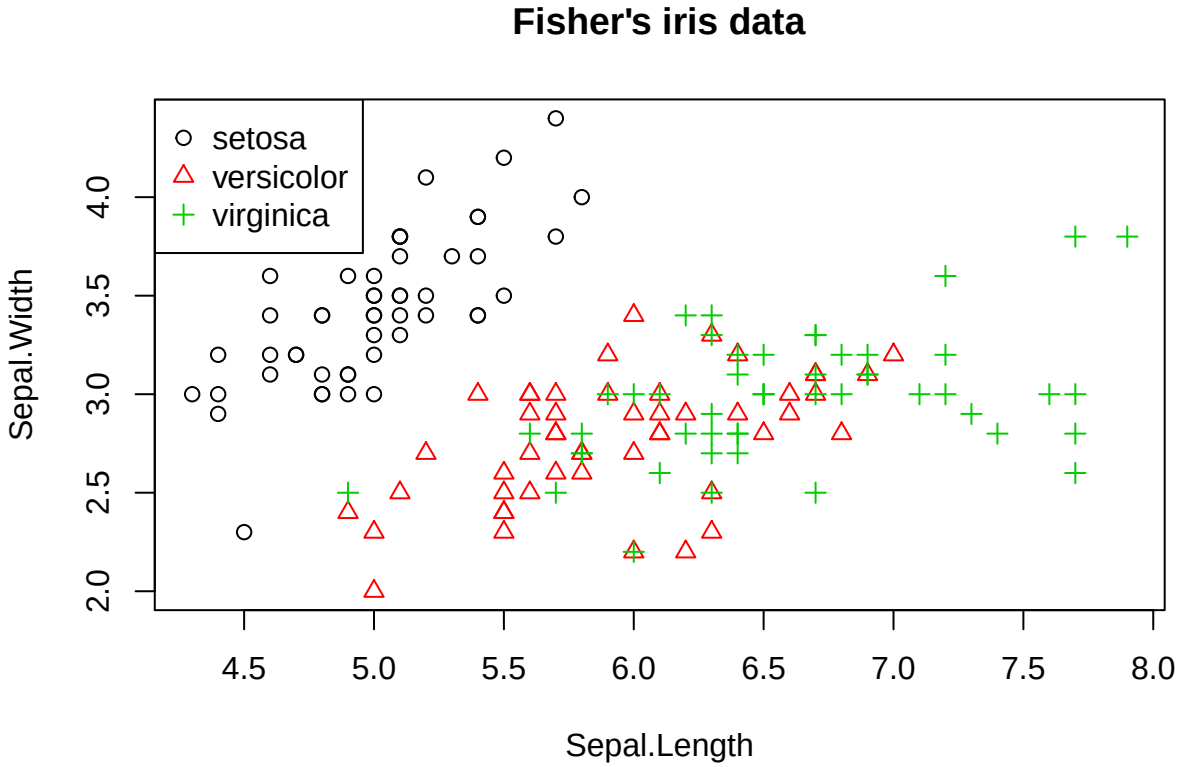
Intro to R Programming: Lab 4

All data files for this lab need to be downloaded from the module's Moodle page first.

Task 1

The `iris` dataset gives, for 50 flowers, four measurements (sepal length and width and petal length and width). The flowers in the data set come from three species: Iris setosa, versicolor, and virginica.

1. Read the data correctly into R.
2. Create a scatterplot of sepal length and sepal width, where different species are denoted by a different plotting symbol and a different color.
3. Add a legend to the plot and the title `Fisher's iris data`. Your final plot should look like the following one.

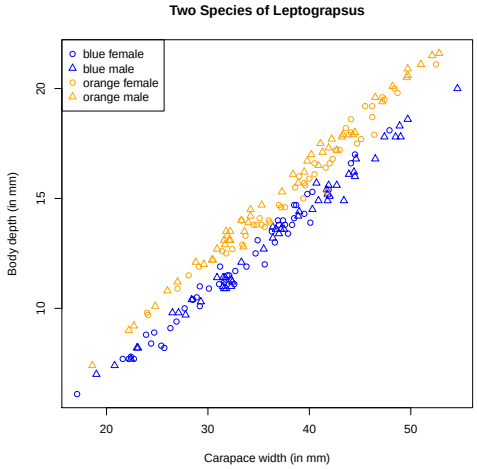


Task 2

The `crab` data contains several morphological measurements of crabs of two species (column `sp`) and both sexes (column `sex`). The statistical objective is to use the morphological measurements to tell the two species apart (though the two species are called `blue` and `orange`, their colour is pretty much the same). The data contains the following variables.

Column name	Variable
`sp`	species – "B" for "blue" and "O" for "orange"
`sex`	gender of the animal
`index`	index counter
`FL`	frontal lobe size (mm)
`RW`	rear width (mm)
`CL`	carapace length (mm)
`CW`	carapace width (mm)
`BD`	body depth (mm)

1. Create boxplots of the variable `FL`, `RW`, `CL`, `CW` and `BD`.
2. Create two vectors `CW.orange` and `CW.blue` containing the measurements of the carapace width for the two species, respectively. Finally, create boxplots of both variables. To make comparing the box plots easier, it is best to place them in the same figure.
3. Create the same boxplots as in part 2 without creating the vectors `CW.orange` and `CW.blue`.
4. Create boxplots of the other variables for each of the two species. Based on the boxplots, does it seem possible to tell the two species apart?
5. Create a scatter plot of the columns `CW` (carapace width) and `BD` (body depth).
6. It is usually a good idea to both have self-explanatory axis labels and to give the units of measurement. Change your command from part 5 so that the axis label on the horizontal axis is `Carapace width (in mm)` and that the label on the vertical axis is `Body depth (in mm)`. The title of the plot should be `Two Species of Leptograpsus`.
7. Change your plotting command from the previous task such that the plotting symbol reflects the sex and that the colour reflects the species.
8. The two species are actually called `blue` and `orange`. Change your plotting command so that it uses these two colours.
9. Add a legend to your plot, which explains both the plotting symbols and the colours used, as in the plot below.



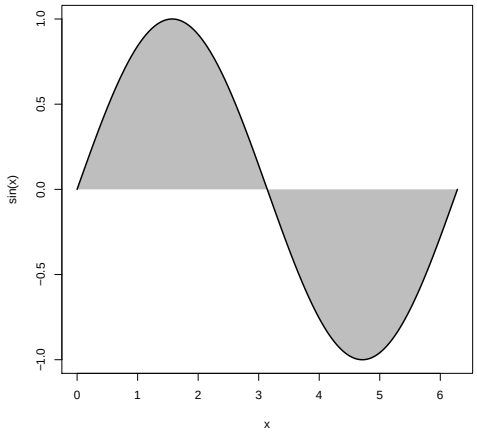
10. What happens when you run the commands `rug(crabs$CW)` and `rug(crabs$BD, side=2)` after you have created the scatter plot?
11. Use the function `pairs` to create a scatter plot of all measurements in the data set. Just like in the previous task, the plotting symbol should reflect the sex and the colour should reflect the species.

12. Finally, interpret the plots you have created. What is the best way of telling blue and orange crabs apart?

13. One possibility of how one might be able to distinguish between the two species is to study the ratio carapace width/body depth. Create a new variable `ratio` which contains this ratio. Create a boxplot of the ratio for each of the two species. Can this ratio be used to tell the two species apart?

Task 3

Draw the function $\sin(x)$ for $x \in [0, 2\pi]$. Fill the area between the x axis and $\sin(x)$ (i.e. the area corresponding to the integral $\int_0^{2\pi} \sin(x)\, dx$) in grey. The plot should look like the figure below.



Task 4

The function `locator(n, type)` reads the coordinates of the mouse when the left-mouse button is pressed. locator reads in total `n` clicks unless the user aborts by pressing the *Esc* key or by clicking the right mouse button. The argument `type` controls whether (and how) the points are plotted. `type="n"` (default) plots nothing, `type="p"` plots the points (see `?locator` for more details). `locator` returns a list containing the coordinates of the points as x and y.

1. Set up an empty plotting area with range $(-1, 1) \times (-1, 1)$ (you can use the option `NULL` to do this). Use the function `locator` to read 15 in points from mouse clicks.

2. After having read the coordinates, colour the points above the bisector $y = x$ in blue and the other points in red.

3. Use the function `abline` to draw the bisector $y = x$ as a dotted line.

Intro to R Programming: Lab 5

Task 1

Create vectors `x` and `y` using the commands

```
x <- c(1,2,9)
y <- c(2,6,4)
```

Write a `for` loop to compute the element-wise product $z = x \cdot y$. You can do this as follows:

1. Create a vector `z` which has the same length as `x` and `y`.
2. Write a `for` loop which goes through the vector `z` and sets the i-th entry of `z` to $z_i = x_i \cdot y_i$.

Compare your result to `x*y`.

Task 2

The R function `cumsum(x)` computes the cumulative sum of a vector `x`. z is the cumulative sum of x, if $z_1 = x_1$, $z_2 = x_1 + x_2$, ..., $z_k = \sum_{i=1}^{k} x_i$, ... So for example is `x` is $(1, 2, 3, 4)$, then cumulative sum `z` is $(1, 3, 6, 10)$. You can compute the cumulative sum of the vector `x<-c(1,2,9)` using `cumsum(x)`. Implement a `for` loop instead of the built-in function `cumsum` to compute the cumulative sum of `x`.

Hint: If $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ *denotes the cumulative sum of* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$*, then set* $c_1 = x_1$ *and* $c_i = c_{i-1} + x_i$ *for* $i = 2, \ldots, n$.

Task 3

For $x_0 > 0$ the recursive sequence defined as

$$x_n = 1 + \frac{1}{x_{n-1}}$$

can be shown to tend to the "golden ratio" $\frac{1+\sqrt{5}}{2}$ as $n \longrightarrow +\infty$.

1. Write a loop that approximates the golden ratio by computing x_{50}.
2. Modify your code from part (a) such that the loop stops as soon as either $|x_n - x_{n-1}| < 10^{-10}$ or 50 iterations haven been carried out (whichever occurs first).

Task 4

In this task you will plot the blancmange function (named after a French dessert called *blancmange*, which is similar to the Italian *panna cotta*). The blancmange function is for $x \in [0, 1]$ defined as

$$b(x) = \sum_{n=0}^{+\infty} \frac{s(2^n x)}{2^n}, \qquad \text{where } s(z) = \min_{m \in \mathbb{Z}} |z - m|$$

The blancmange function is a fractal function which is uniformly continuous but nowhere differentiable.

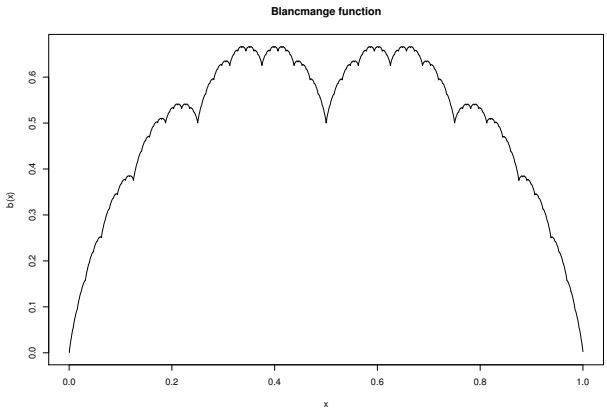
To evaluate the blancmange function at 4096 equally spaced values between 0 and 1 the following algorithm can be used.

1. Create a vector $\mathbf{b}$ of length 4096 consisting of 0's only.
2. Create another vector $\mathbf{t}$ of length 4096, which consists of the numbers $\frac{1}{4096}(1, 2, \ldots, 2047, 2048, 2048, 2047, \ldots, 2, 1)$.
3. Set $w = \frac{1}{2}$
4. For $i = 1, 2, \ldots, 10$:
   (a) Set $\mathbf{b}$ to $\mathbf{b} + \mathbf{t}$.

(b) Set $\mathbf{t}$ to $w \cdot (t_2, t_4, \ldots, t_{4096}, t_2, t_4, \ldots, t_{4096})$.

The vector $\mathbf{b}$ then contains the values of the blancmange function at $\mathbf{x} = \frac{1}{4096}(1, 2, \ldots, 4096)$.

First implement the above algorithm and plot the blancmange function. The plot should look similar to the plot below. What happens if you set $w = \frac{1}{4}$ instead of $w = \frac{1}{2}$?

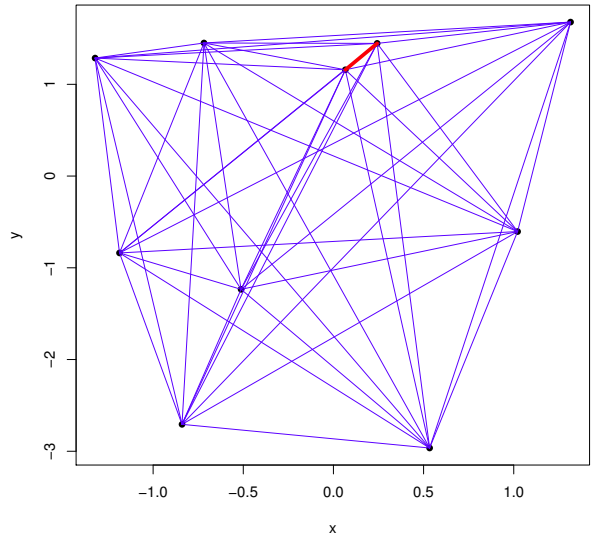


Task 5

Simulate and plot ten points from $\mathbb{R}^2$ using

```
n <- 10
coords <- matrix(rnorm(2*n), ncol=2)
plot(coords)
```

1. Connect all pairs of points using blue lines. Your plot should look like the one shown below (except for the thick red line). *Hint: You will need two nested* `for` *loops.*
2. *(much harder)* Connect the two closest points with a thick red line. Your code should find the two closest points automatically. Your plot should now look like the plot shown below (it will depend on your simulated values). *Hint: You need to loop through all pairs of points and identify the pair with smallest Euclidean distance.*



Intro to R Programming: Lab 7

Task 1

Read the help file of the function `dbinom`, which evaluates the p.m.f. of the $\mathsf{Bi}(n, \theta)$ distribution.

1. Use the function `dbinom` to evaluate the p.m.f of the $\mathsf{Bi}(10, 0.5)$ distribution at $x = 3$. Specify the arguments to `dbinom` once using the named form and once using the positional form.
2. What does `dbinom(size=4, 1, 0.5)` compute?

Task 2

Consider the function

```
my.function <- function(z) {
  m <- Inf
  for (i in 1:length(z))
    if (z[i]<m)
      m <- z[i]
  m
}
```

What does the function `my.function` do?

Task 3

The following simple function computes the least squares estimates of a regression of `y` against a vector or matrix `x` of covariates.

```
least.squares <- function(x, y) {
  X <- cbind(1, x)
  XtX <- t(X)%*%X
  Xty <- t(X)%*%y
  solve(XtX, Xty)
}
```

The linear model requires that the length of `y` equals the number of rows of `X` and that `X` has not more columns than rows. Add two `if` statements to the above definition of the function `least.squares` that check whether these conditions hold and, in case at least one does not hold, produce a meaningful error message.

Task 4

Create a matrix `A` with entries

$$\begin{pmatrix} 1 & 8 & 5 \\ 4 & 3 & 6 \end{pmatrix}$$

```
A <- rbind(c(1, 8, 5),
           c(4, 3, 6))
```

What do each of the following commands do?

```
apply(A, 2, sum)
apply(A, 1, sum)
sweep(A, 2, c(1, 3, 5), "-")
sweep(A, 1, c(8, 6), "/")
```

Task 5

1. The binomial coefficient is defined as
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$
Write a function `binomial.coefficient` that takes `n` and `k` as arguments and that returns $\binom{n}{k}$. Use it to compute $\binom{6}{3}$. *Hint: $x!$ can be computed using the function* `factorial`.

2. Write a function `binary.entropy`, which takes a number `p` ($0 < p < 1$) as argument and returns the binary entropy
$$H(p) = -p\log(p) - (1-p)\log(1-p)$$

3. Write a function `approx.lbincoef`, which takes integers `n` and `k` as arguments and returns the approximation
$$n \cdot H\left(\frac{k}{n}\right)$$
to $\log\binom{n}{k}$, the logarithm of binomial coefficient. $H(\cdot)$ is the function defined in part 2 of this question.

4. Use your functions from part 1 and 3 to compute (an approximation to) $\log\left(\binom{9000}{4000}\right)$. *Hint: The exact answer is* $\log\left(\binom{9000}{4000}\right) = 6177.88\ldots$.

Task 5

The Box-Cox transformation of a data vector $\mathbf{y} = (y_1, \ldots, y_n)$ with positive entries is defined as
$$y_i^{(\lambda)} = \begin{cases} \frac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log y_i & \text{if } \lambda = 0, \end{cases}$$
where λ is a parameter controlling the shape of the transformation.

1. Write a function `box.cox` that takes a vector `y` and a scalar constant `lambda` as only arguments and which returns the Box-Cox transform of `y`. The default value of `lambda` should be 0.

2. The package `MASS` contains a data set `mammals`, which you can make available in your R session using the commands
```
library(MASS)
data(mammals)
```
Use the function `box.cox` you have written in part 1 to compute the Box-Cox transformation of the variable `brain` of the data set `mammals` both for $\lambda = 0$ and $\lambda = 0.1$.

Task 7

Consider a sample of observations $\mathbf{x} = (x_1, \ldots, x_n)$ coming from a symmetric distribution and suppose we want to estimate its mean. The following three quantities all provide an unbiased estimate of the mean of the x_i:

- the sample mean,
- the sample median, and
- the sample midrange $\dfrac{\min\{x_1, \ldots, x_n\} + \max\{x_1, \ldots, x_n\}}{2}$.

The three quantities all have different variance, and we want to find out which has the lowest variance. The answer to this question depends on the distribution of the x_i, but for simplicity we will assume that the x_i come from a normal distribution.

Write an R script or function which implements the following. Draw 10,000 random samples of 100 observations each from the $\mathsf{N}(0, 1)$ distribution. For each sample compute the mean, median and midrange. Compute the variance of the computed means, medians and midranges.

Intro to R Programming: Lab 8

Task 1

The coefficient of variation of a variable is defined as the ratio of the standard deviation and the mean.

1. Define a function `cv` which takes as input a vector `x` and which returns the coefficient of variation.
2. Use the function `cv` computed in part 1 and the function `apply` to compute the coefficient of variation of the numeric variables of the `iris` dataset you have already encountered in the module. `iris` is freely available from R.

Task 2

In this question you will write a function which computes "direct" asymptotic confidence intervals for data from a binomial distribution. If $X \sim \mathsf{Bi}(n, \theta)$ and x is the observed number of successes, then

$$\left(\hat{\theta} - z_{1-\alpha/2} \cdot \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{n}}, \hat{\theta} + z_{1-\alpha/2} \cdot \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{n}}\right)$$

is an asymptotic $1-\alpha$ confidence interval for the proportion θ ("probability of success"), where $\hat{\theta} = \frac{x}{n}$ and $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the normal distribution *(see* `?qnorm` *for details)* .

The following script computes the confidence interval:

```
n <- 30                                          # Sample size
theta <- 0.6                                     # Observed proportion
alpha <- 0.05
sd.theta <- sqrt(theta*(1-theta)/n)              # Compute standard deviation of x/n
ci <- theta + c(-1,1) * qnorm(1-alpha/2) * sd.theta # Compute confidence interval
names(ci) <- c("lower", "upper")                 # Label the two values in ci
ci
```

1. Write a function `ci.proportion.direct` that takes `n`, `theta`, and `alpha` as arguments and returns the confidence interval as a vector. Compute the 95% confidence interval for an observed proportion of $\theta = 0.7$ and a sample size of $n = 50$. The default value of `alpha` should be 0.05.
2. *(harder)* Change your function from part 1 such that the user can either specify the proportion `theta` or the count `x`. If `theta` is not specified, it is computed from `x` as $\theta = x/n$. You should be able to call the function using either `ci.proportion.direct(n=20, x=10)` or `ci.proportion.direct(n=20, theta=0.5)`.
3. When using the direct method for obtaining confidence intervals, one might obtain bounds less than 0 (e.g. if $x = 1$ and $n = 100$) or larger than 1 (e.g. if $x = 99$ and $n = 100$). Change your function from part 2 so that the confidence interval is capped at 0 and 1.

Task 3

Zeller's congruence formula is an algorithm for determining the day of the week for any given date. Let q be the day of the month, m the month and y the 4-digit year. Define

$$j = \begin{cases} \lfloor \frac{y-1}{100} \rfloor & \text{if } m \in \{1,2\} \\ \lfloor \frac{y}{100} \rfloor & \text{if } m \in \{3,4,\ldots,12\} \end{cases} \quad k = \begin{cases} (y-1) \mod 100 & \text{if } m \in \{1,2\} \\ y \mod 100 & \text{if } m \in \{3,4,\ldots,12\} \end{cases} \quad n = (m+9) \mod 12+1$$

$\lfloor a \rfloor$ hereby denotes the floor of a, i.e. the largest integer not larger than a. The floor of a can be obtained in R using `floor(a)`. a mod b denotes the remainder after integer division and can be computed in R using `a%%b`.

The day of the week is then given by the formula

$$h = \left(\lfloor 2.6n - 0.2 \rfloor + q + k + \left\lfloor \frac{k}{4} \right\rfloor + \left\lfloor \frac{j}{4} \right\rfloor - 2j + 6 \right) \mod 7 + 1,$$

where $h = 1$ stands for Monday, $h = 2$ stands for Tuesday, etc.

Worked example: For 27/11/2017, we have that $q = 27$, $m = 11$ and $y = 2017$. Thus $j = 20$, $k = 17$ and $n = 9$ and thus

$$h = \left(\lfloor 2.6 \cdot 9 - 0.2 \rfloor + 27 + 17 + \left\lfloor \frac{17}{4} \right\rfloor + \left\lfloor \frac{20}{4} \right\rfloor - 2 \cdot 20 + 6 \right) \mod 7 + 1 = 42 \mod 7 + 1 = 0 + 1 = 1$$

Thus for 27/11/17 we obtain 1 (Monday).

Write a function `zeller`, which takes in the day of the month, month (as integer) and year (as four digit integer) and which returns the day of the week as plain English text (e.g. `"Monday"`).

Task 4

By its very nature a computer cannot generate random numbers: a computer is a fully deterministic machine. However, computers can generate pseudo-random numbers. A pseudo-random number generator (RNG) is an algorithm for whose output the $\mathsf{U}[0,1]$ distribution is a suitable model. In other words, the numbers generated by the pseudo-random number generator should have the same *relevant* statistical properties as independent realisations of a $\mathsf{U}[0,1]$ random variable: pseudo-random numbers should not be predictable and should be spread uniformly across the unit interval $[0,1]$.

A simple, but not very good, type of random number generator is the linear congruential generator.

- Choose parameters $a, M \in \mathbb{N}$, $c \in \mathbb{N}_0$, and the initial value ("seed") $z_0 \in \{1, \dots M-1\}$.
- For $i = 1, 2, \dots, n$ set $z_i = (az_{i-1} + c) \mod M$, and $x_i = z_i/M$. $a \mod b$ computes the remainder after integer division, which can be obtained using `a%%b` in R.

The integers z_i generated by the algorithm are from the set $\{0, 1, \dots, M-1\}$ and thus the x_i are in the interval $[0, 1)$.

1. Write a function `lcg`, which takes the required number of pseudo-random numbers n, the parameters a, M and c, and the initial value z_0 as arguments. The function should return a sequence of n pseudo-random numbers generated from the above algorithm. The default values of the arguments should be $a = 2^{16} + 3$, $M = 2^{31}$, $c = 0$ and $z_0 = 1$.
2. Create a vector $\mathbf{x} = (x_1, x_2, \dots, x_{300000})$ of length 300,000 filled with pseudo-random numbers generated using your function from part 1 using the default values for a, M, c and z_0.
3. Using the vector $\mathbf{x}$ from part 2, create a $100{,}000 \times 3$ matrix with entries

$$\mathbf{X} = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \\ \vdots & \vdots & \vdots \\ x_{299998} & x_{299999} & x_{300000} \end{pmatrix}$$

4. Load the package `rgl` and create a three-dimensional scatterplot of the columns of the matrix $\mathbf{X}$ using the code

```
library(rgl)
plot3d(X)
```

   Use your mouse to rotate the three-dimensional scatterplot. What can you observe?

R has a much more powerful RNG, so please use the RNG built into R (and used by functions like `runif` and `rnorm`); do not use the linear congruential generator described above for anything important.

Intro to R Programming: Lab 9

Task 1

Consider the following function $f : \mathbb{R} \to \mathbb{R}$ and its derivatives

$$f(\theta) = \sin(3\theta) - \theta^2$$
$$f'(\theta) = \frac{\partial}{\partial\theta} f(\theta) = 3\cos(3\theta) - 2\theta \qquad f''(\theta) = \frac{\partial^2}{\partial\theta^2} f(\theta) = -9\sin(3\theta) - 2$$

1. Write three functions `f`, `f.d`, and `f.dd` which take `theta` as argument and which return $f(\theta)$, $f'(\theta)$, and $f''(\theta)$, respectively.
2. Use R to create a sketch of the function $f(\theta)$ for $\theta \in [-3, 3]$.
3. Use the function `optimize` to find the maximum of $f(\cdot)$.
4. Implement Newton's method to find a local extremum of $f(\cdot)$. Run Newton's method with different starting values. Can you find the global maximum you have identified in part 3?

Task 2

Store numerical data of your choice in a vector `x`.

1. Compute the mean of `x` and use the function `optimize` to find the value m minimising the objective function
$$\sum_{i=1}^{n} (x_i - \theta)^2.$$
Hint: Create a function which takes the arguments `theta` *and* `x` *as arguments (in that order), and which returns the value of the objective function. Then use* `optimize` *to find its minimum.*
2. Compute the median of `x` and use the function `optimize` to find the value θ minimising the objective function
$$\sum_{i=1}^{n} |x_i - \theta|.$$

Task 3

In this question you will compute the 95% quantile of the normal distribution, i.e. we want to solve

$$\Phi(z) = 0.95$$

for z, where $\Phi(z) = \int_{-\infty}^{z} \phi(t)\, dt$ is the c.d.f. of the standard normal distribution. The function $\Phi(\cdot)$ can be calculated in R using the function `pnorm`.

1. Use the function `uniroot` to compute a solution to the above equation. *Hint: Define a function which takes* `z` *as argument and which returns* $\Phi(z) - 0.95$ *and apply* `uniroot` *to it. You can compare your result to the quantile obtained using* `qnorm(0.95)`.
2. Implement Newton's method to solve the above equation. *Hint:* $\Phi'(z) = \phi(z)$*, which can be found in R using the function* `dnorm`.

Task 4

A two-component Gaussian mixture model is a probabilistic model which assumes that the data is in two clusters.

The model assumes for scalar data that the probability density function of X is a weighted sum of two Gaussians with different mean and standard deviation, i.e.

$$f(x_i) = p \cdot f_{(\mu_1,\sigma_1^2)}(x_i) + (1-p) \cdot f_{(\mu_2,\sigma_2^2)}(x_i),$$

where $f_{(\mu,\sigma^2)}(x) = \frac{1}{\sqrt{2\pi\cdot\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ is the probability density function of the $N(\mu,\sigma^2)$ distribution evaluated at x. $0 < p < 1$ is the assumed proportion of observations coming from the first cluster. μ_1 and $\sigma_1 > 0$ are the mean and standard deviation of the data in first cluster with μ_2 and $\sigma_2 > 0$ being the mean and standard deviation of the data in the second cluster.

For data $\mathbf{x} = (x_1, \ldots, x_n)$ the loglikelihood of the two-component Gaussian mixture model is

$$\ell(p, \mu_1, \sigma_1, \mu_2, \sigma_2) = \sum_{i=1}^{n} \log f(x_i) = \sum_{i=1}^{n} \log\left(p \cdot f_{(\mu_1,\sigma_1^2)}(x_i) + (1-p) \cdot f_{(\mu_2,\sigma_2^2)}(x_i)\right)$$

1. Write a function `gmm.loglik` which takes the parameter vector `par` $= (p, \mu_1, \sigma_1, \mu_2, \sigma_2)$ as first argument and the data `x` as second argument and which returns the loglikelihood $\ell(p, \mu_1, \sigma_1, \mu_2, \sigma_2)$.*Hint: Use* `dnorm(x, mu, sigma)` *to evaluate the p.d.f of the* $N(\mu, \sigma^2)$ *distribution at* x. *Part 2 requires the function to take the parameters as one vector* `par` *of length 5.*

2. Use the function `optim` to maximise the loglikelihood function $\ell(p, \mu_1, \sigma_1, \mu_2, \sigma_2)$ over the parameter vector $\boldsymbol{\theta} = (p, \mu_1, \sigma_1, \mu_2, \sigma_2)$. Use the data contained in the vector `mixturedata` as data `x`.

3. Using the estimated parameters plot the estimated p.d.f. of X

   $$f(x) = p \cdot f_{(\mu_1,\sigma_1^2)}(x) + (1-p) \cdot f_{(\mu_2,\sigma_2^2)}(x)$$

   for $x \in [-3, 4]$. Your plot should look like the plot shown below.