Lab Assignment Week 1: Introduction to R

Phong Le, Dieuwke Hupkes, Sara Veldhoen, Jelle Zuidema

31 October

In today's computer lab we will learn how to use R to draw graphs and create some basic scripts. Then, we will use R to do some vector and matrix algebra. If you are absolutely unfamiliar with R, we recommend first following a tutorial, such as:

- `http://www.illc.uva.nl/LaCo/CLAS/clc13/assignments/deboer13rtutorial.pdf`

1 Some useful R exercises

In this section we familiarize ourselves with some R commands and packages that we will use later on.

1.1 Vectors and matrices

Vectors and matrices are very important in R. We will store most of our data in vectors. In this section, we will go over some ways to construct vectors, and how to retrieve specific elements within them.

Construction There are many ways to construct a vector. The first, also the simplest, way is to use the `c` function, e.g., `vec <- c(1,2,3)` creates a *column* vector

$$vec = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Another way is to use the `rep`, `seq`, and `runif` functions etc. (Check yourself what those functions do by typing `help(function_name)`.)

> Create the following vectors
>
> $$\vec{a} = \begin{pmatrix} 1.5 \\ -1 \\ 3 \end{pmatrix} \quad \vec{b} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} \quad \vec{c} = \begin{pmatrix} 2 \\ 4 \\ 6 \\ ... \\ 24 \end{pmatrix}$$
>
> and vector $\vec{d}$ containing 12 real numbers drawn from the uniform distribution `U(-1,1)`. (Hint: use `rep, seq, runif` to construct `b, c, d` respectively.)

To create a matrix, we use

```
matrix(vector, number_of_rows, number_of_columns)
```

For instance, the output of `matrix(c(1,2,3,4,5,6),2,3)` is the matrix

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

(Mind the element order in the matrix w.r.t. the order in the given vector.)

Create the following matrices

- $$A = \begin{pmatrix} 3 & 1 & 2 \\ -4 & 2 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 \\ 5 & -3 \end{pmatrix}$$

  Type `dim(A)` to confirm that A is a 2×3 matrix.

- `D` is a 3×5 matrix which the elements are drawn from a uniform distribution.

- `E` is a 5×5 identity matrix (Hint: use the `diag` function).

Indexing We can get an element by using the operator `[]`. For instance, `vec[i]` points to the i-th element of a vector `vec`, `mat[i,j]` points to the element on the i-th row, the j-th column of a matrix `mat`.

The operator `[]` can also be used to get a set of elements. For instance, `a[c(1,3)]` and `a[c(TRUE,FALSE,TRUE)]` point to the first and the third elements of vector `a`; `A[1,c(2,3)]` and `A[1,c(FALSE,TRUE,TRUE)]` point to the second and the third elements on the first row of matrix `A`, and `A[,2]` points to the second column.

1.2 Loops

To execute the same operation a certain number of times, we can use a loop. R has several types of loops available, here we will only mention the `for` loop. The syntax of the `for` statement is as follows:

```
for (i in start:end) {
    statement
}
```

For instance, say we want to create a vector that contains the squares of the natural numbers we could do:

```
squares <- rep(0, 20)          # initialise vector
for (i in 1:len(squares)) {
    squares[i] = i^2
}
```

Loops are expensive, and should be avoided if there is another option.

1.3 Functions

If we want to repeat certain blocks of code, we can encompass it in a function. The structure of a function is as follows:

```
func_name <- function(arg1, arg2, ...) {
    statement
    return(output)
}
```

where `func_name` is the function's name; `arg1`, `arg2`, etc. are the function's arguments (i.e., function's inputs). `return(output)` tells us that the function returns the object `output` (output could, for instance, be a vector). A function that computes $f(x) = x^2 + 5$ would be:

```
f <- function(x) {
    return(x^2 + 5)
}
```

Note that a function does not necessarily have to return something.

When the output of the function can be directly computed from the input, the resulting function can often also be applied to a vector. The function is then applied to all elements of the vector.

Write a function f(x) that computes the square of a number, and then use it to create a vector with squared numbers, such as the one that we created in Section 1.2.

1.4 Graphics

R makes it very easy to create plots. In this section, we will look at how to make simple line plots.

Line plots If we want to plot the function $f(x) = sin(x)$ we can do that easily by creating a set of x values and their corresponding y values and calling `plot(x, y, type='l')`:

```
1 x <- seq(-5,5,by=0.1)              # create x = (-5,-4.9,-4.8,...,4.9,5)
2 y <- sin(x)                        # compute y values
3 plot(x,y,type='l',col='blue')      # draw the graph, using blue color
4 savePlot("sin.png",type="png")     # save the current plot to a png file
```

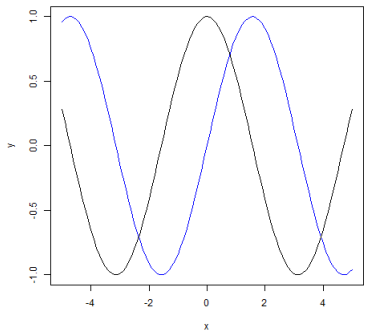


Figure 1: Drawing two functions, $y = sin(x)$ and $y = cos(x)$ in one graph.

If you want to draw another function on the same graph, you need to use the `lines` function. For instance, insert `lines(x,cos(x))` before the last line. Now, you should get a graph similar to Figure 1. Note that `lines` can only be called when a plot-grid already exists. You can create an empry grid by setting `type` to 'n'.

Arrows You can add an arrow to an existing plot using `arrows(x0, y0, x1, y1)`, where `(x0,y0)` and `(x1,y1)` are the coordinates of the points from and to which to draw, respectively. Again, you can use a color, and other properties to (see documentation online or prompt `?arrows`)

1.5 Scripts

Programming directly in the console is not very effective. If you make a mistake you have to start all over again, you cannot re-use code and you cannot bring your program to another computer. Therefore, you usually want to store your source code in a file, and execute it from the console by calling `source("file")`.

Store the code of the previous exercise in a file "sin.R" and execute it from the console by typing `source(''sin.R'')`. If the error "cannot open file" occurs, you should make sure the file sin.R is in the working directory. The current working directory can be queried with `getwd()`. To change the current working directory, you can use `setwd(path_to_directory)`. If you press tab, your console should give you a list of options.

2 Linear algebra

2.1 Vector arithmetic

- **Dimensionality**
  This sometimes referred to as 'length'. For example, `length(c(0,0,0))` returns `3`.
- **Length or (Euclidean) norm**
  The length of a vector can be computed as $|\vec{v}| = \sqrt{v_1^2 + \ldots + v_n^2}$
- **Scaling**
  You can scale a vector v by a scalar value a: $a \times \vec{v} = \vec{u}$ with $u_i = a \times v_i$
- **Vector addition**
  Vectors of the same dimensionality can be added: $\vec{v} + \vec{w} = \vec{w} + \vec{v} = \vec{u}$ with $u_i = v_i + w_i$
- **Vector multiplication (dot product)**
  Vectors of the same dimensionality can be multiplied, which results in a scalar: $\vec{v} \times \vec{w} = \vec{w} \times \vec{v} = a$, with $a = \sum v_i \times w_i$

Exercise 1. Write a function that computes the length of a vector (hand in your code). Use it to compute the length of the vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ from the first section, and show that $|a \times \vec{v}| = a \times |\vec{v}|$ (provide a table with a few values of a)

Exercise 2. Visualize some vector arithmetics

- Create the vectors: $\vec{v} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ and $\vec{w} = \begin{pmatrix} -0.1 \\ 0.4 \end{pmatrix}$
- Initialize a plot (draw invisible points) as follows: `plot(c(-1,1),c(-1,1),col='white')`
- Plot the vectors $\vec{v}$ and $\vec{w}$ in black (use the function arrows, e.g. `arrows(0,0,v[1],v[2])`)
- In the same plot, draw $0.5 * \vec{v}$ in red.
- In the same plot, draw $\vec{v} + \vec{w}$ in blue.

2.2 Matrix arithmetic

- **Size**
  Height (number of rows)× width (number of columns) of the matrix
- **Scaling**
  $c \times A = B$, with $B_{i,j} = c \times A_{i,j}$
- **Addition**
  $A + B = B + A = C$, with $c_{i,j} = a_{i.j} + b_{i,j}$, only if A and B same sizes
- **Multiplication** $A \times B = C$, with $c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$, if no. columns A = no. rows B = p

  Example: $\begin{pmatrix} -1 & 2 \\ 4 & 1 \end{pmatrix} \times \begin{pmatrix} 4 & 1 \\ 3 & 5 \end{pmatrix} = \begin{pmatrix} -1 \cdot 4 + 2 \cdot 3 & -1 \cdot 1 + 2 \cdot 5 \\ 4 \cdot 4 + 1 \cdot 3 & 4 \cdot 1 + 1 \cdot 5 \end{pmatrix} = \begin{pmatrix} 2 & 9 \\ 19 & 9 \end{pmatrix}$

In R, use `%*%` to compute matrix product, as `*` returns the *element-wise* product (only for matrices or vectors of the same dimensionality). This can be a bit confusing, keep it in mind in subsequent labs!

Play around to understand the difference between `%*%` and `*`.

Make a list of the appropriate (built-in) functions in R to compute all the functions listed under 'Vector arithmetic' and 'Matrix arithmetic' above.

Exercise 3. Use R to compute $A \times B$ and $B \times A$. Is the result the same? Explain why (not).

2.3 Vector Transformations

Multiplying a vector with a *square* matrix transforms the vector:

$$A \times \vec{v} = \vec{w}$$

Usually the new vector has a new direction or length, or both.

Create a square matrix, e.g. B we saw earlier, and apply it to different vectors to see what linear transformations it defines. Plot your results to get a better understanding

You can define a matrix A that only rotates (and does not scale) a vector as follows. After applying this matrix to a vector, the new vector has the same length but a different direction. We thus call this a *rotation matrix*.

$$A\vec{v} = \begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} v_x cos\phi - v_y sin\phi \\ v_x sin\phi + v_y cos\phi \end{pmatrix}$$

Exercise 4. Like in exercise 2, create an empty plot and draw the vector `v=c(0.5,0.5)`. Create a matrix that rotates a vector by 45 degrees. Apply it to the vector $\vec{v}$, and draw the resulting vector in red in the same figure.

Eigendecomposition Let us now focus on a matrix $M = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$.

Create an empty plot and find out what happens for various vectors in the range from -1 to 1 if they get multiplied by the matrix M. For most vectors, both the length and the direction changes. Note however that some of the vectors do not get not rotated by mutliplication with M. For these vectors, it holds that $M \times \vec{v} = \lambda\vec{v}$. In other words, the vectors get scaled by the matrix. These are so-called *eigenvectors*. The corresponing scaling factor λ is called the *eigenvalue*.

Exercise 5. Report the eigenvectors that you found.

As you should have noted, some eigenvectors are scaled versions of others. In fact, we can state that if $\vec{v}$ is an eigenvector of a matrix M, that for an arbitrary number k it should be the case that $k\vec{v}$ is also an eigenvector, with the same eigenvalue. In other words, eigenvectors are not unique. An $n \times n$ square matrix has a maximum of n eigenvalues, and equally many *linearly independent* eigenvectors.

Determinant The *determinant* of a 2×2 matrix, is computed as follows:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad \det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - cb$$

Exercise 6. Confirm this definition using the R function `det()`. Compute the determinant for matrix B and for the following matrix: $\begin{vmatrix} -1 & 2 \\ 4 & 1 \end{vmatrix}$

Finding eigenvalues We will only be concerned with 2×2 matrices. One way to compute the eigenvalues for such a given matrix, makes use of the determinant computation treated above.

$$A\vec{v} = \lambda\vec{v} \text{ can be rewritten as: } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \lambda \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

$$\text{So, } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} - \lambda \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} a - \lambda & b \\ c & d - \lambda \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

From which we can derive the so-called *characteristic equation*:

$$(a - \lambda)(d - \lambda) - bc = 0 \tag{1}$$

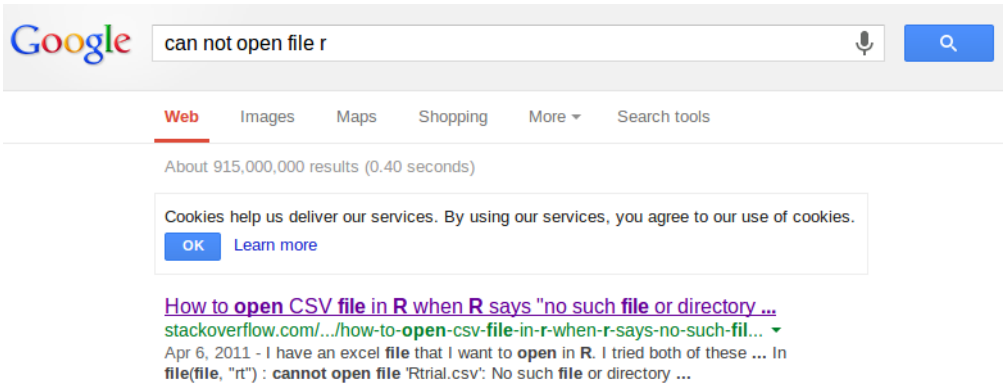
This equation with one unknown variable can be solved to yield the eigenvalue(s) λ.

The corresponing eigenvalues are $\begin{pmatrix} -b \\ a - \lambda_{1,2} \end{pmatrix}$ or $\begin{pmatrix} -d + \lambda_{1,2} \\ c \end{pmatrix}$

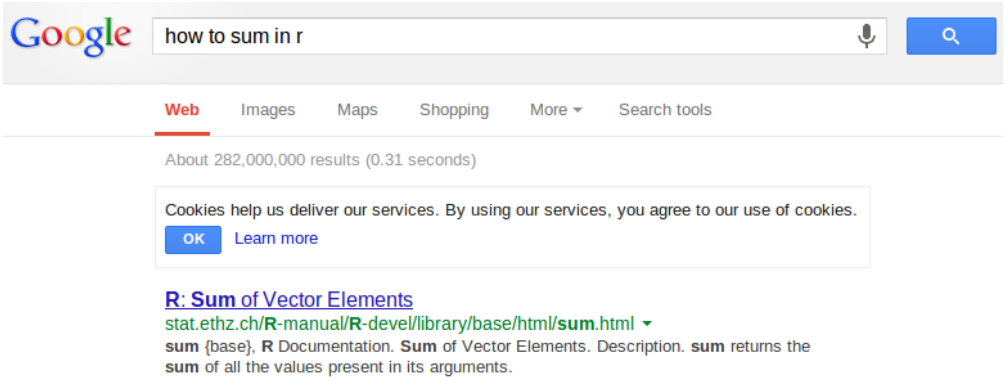
Exercise 7. Use R to compute the eigenvalues of the matrix B.

3 What should you do when you get stuck?

Google your problem! R is widely used, and its user community is very large. Therefore, you can easily find someone that had the same problem and successfully solved it. For instance,



If you can't find any solution on the Internet, post your problem on the course forum.



4 Submission

Submit your solutions to the exercises (marked as **Exercise x.**) via blackboard before the lecture on Thursday (15:00). Please make sure your answers are easy to find and not hidden between blocks of code!