Numerical Methods
5633

Lecture 1

Michaelmas Term 2018

Marina Krstic Marinkovic
mmarina@maths.tcd.ie

School of Mathematics
Trinity College Dublin

R programming

- https://www.r-project.org/

- A Programming Environment for Data Analysis and Graphics
  - ➡ Like you can't learn to swim in an empty swimming pool … you can't learn programming just by reading books/ listening to the courses.
  - ➡ Practice is essential - need to write a lot of scripts!

- (Some) useful references:
  https://cran.r-project.org/doc/manuals/R-intro.pdf
  https://cran.r-project.org/doc/contrib/Seefeld_StatsRBio.pdf
  For Harry Potter fans: www.matthewckeller.com/Lecture1.ppt

Why R?

- https://www.r-project.org/

- Has all the basic functionalities as Matlab (you may use **Matlab/ python** for this course instead of R, if you prefer)
- Free software (easy to install on your laptop/desktop)
- You can add your own packages and share with the community

  "When it comes to software, I much prefer free software, because I have very seldom seen a program that has worked well enough for my needs, and having sources available can be a life-saver." - Linus Torvalds

- Used outside physics & engineering: *environmental statistics, econometrics, medical and public health applications, and bioinformatics…*
- Recently: Data Science - preferred tool

  ➡ might land you a job one day :)

R - Basics

- Effective usage of R requires:
  - ➡ using build-in help system
  - ➡ reusing results (output from most functions can be used as input to other functions)
  - ➡ “(Your favourite search engine) is your friend”
- Starting R:

```
machine:~username$ R
```

- In the “Console” window the cursor is waiting for you to type in some R commands

R - Basics

- You can enter commands one at a time at the command prompt (>) - interactive mode, or run a set of commands from a source file (see last slide).

- There is a wide variety of data types, including vectors (numerical, character, logical), matrices, data frames, and lists.

- To quit R, use

```
> q()
```

R - Data Types

- **Vectors:**

```
> a <- c(1,2,5.3,6,-2,4) # numeric vector
> b <- c("one","two","three") # character vector
> c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE) #logical vector
```

➡ Refer to elements of a vector using subscripts

➡ a[c(3,5)] # 3rd and 5th elements of vector

- **Matrices:**

```
> mymatrix <- matrix(vector, nrow=r, ncol=c,
byrow=FALSE,dimnames=list(char_vector_rownames,
char_vector_colnames))
```

➡ **byrow=TRUE** indicates that the matrix should be filled by rows.

➡ **byrow=FALSE** indicates that the matrix should be filled by columns (the default)

➡ **dimnames** provides optional labels for the columns and rows.

R - Data Types

- **Matrices - examples:**

```
> # generates 5 x 4 numeric matrix
> y<-matrix(1:20, nrow=5,ncol=4)
> # another example
> cells <- c(1,26,24,68)
> rnames <- c("R1", "R2")
> cnames <- c("C1", "C2")
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,
dimnames=list(rnames, cnames))
> #Identify rows, columns or elements using subscripts.
> matrix[,4] # 4th column of matrix
> matrix[3,] # 3rd row of matrix
> matrix[2:4,1:3] # rows 2,3,4 of columns 1,2,3
```

- **Reusing data:**

```
> # plots 2nd column of y as a function of a 1st column of y
> plot(y[,1],y[,2])
```

R - Importing Data

- **From a delimited text file:**

```
> # separated by spaces, no variable names included
> mydata<-read.table("mydata.txt", header=FALSE, sep=" ")
> # separated by commas, variable names included
> mydata_new<-read.table("mydata_named.txt", header=TRUE, sep="\t")
```

- **From a keyboard input (interactive):**

```
> # create a dataframe from scratch
> age <- c(25, 30, 56)
> gender <- c("male", "female", "male")
> weight <- c(160, 110, 220)
> mydata <- data.frame(age,gender,weight)
```

➡ A **data frame** is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.).

- **Other possible inputs: Excel, SAS, SPSS file**

R - Exporting & Missing Data

- **To a delimited text file:**

```
> # writing a previously created table to a TAB delimited .txt
> write.table(mydata_new,"mydata_output.txt",sep="\t")
> # separated by commas, variable names excluded
> write.table(mydata_new,"mydata_output_noname.txt",sep=",",
col.names=FALSE,row.names=FALSE)
```

- **Missing data:**

  ➡ Missing values are represented by the symbol **NA** (not available). Impossible values (e.g., dividing by zero) are represented by the symbol **NaN** (not a number). This holds both for character and numeric data

```
> # testing for missing values
> is.na(x) # returns TRUE of x is missing
> y <- c(1,2,3,NA)
> is.na(y) # returns a vector (F F F T)
```

  ➡ There are ways to exclude them (NA), but we'd rather not create them!

R - Data Manipulation

- **Control Structures:**

if-else

```
> if (cond) expr
> if (cond) expr1 else expr2
```

for

```
> for (var in seq) expr
```

while

```
> while (cond) expr
```

switch

```
> switch(expr, ...)
```

ifelse

```
> ifelse(test,yes,no)
```

➡ **expr** can be multiple (compound) statements by enclosing them in braces { }.

R - Data Manipulation

- **Control Structures - examples:**

```
#transpose of a matrix
> mytrans <- function(x)
    {
    if (!is.matrix(x))
        {
            warning("argument is not a matrix: returning NA”)
            return(NA)
        }
        y <- matrix(1, nrow=ncol(x), ncol=nrow(x))
        for (i in 1:nrow(x))
            {
            for (j in 1:ncol(x))
                {
                y[j,i] <- x[i,j]
                }
            }
            return(y)
    }
```

```
> y<-matrix(1:20, nrow=5,ncol=4)
> mytrans(y)
```

➡ Built in **t()** function would do this faster.

R - Data Manipulation

- **Built in functions:**

Function	Description
abs(*x***)**	absolute value
sqrt(*x***)**	square root
ceiling(*x***)**	ceiling(3.475) is 4
floor(*x***)**	floor(3.475) is 3
trunc(*x***)**	trunc(5.99) is 5
round(*x***, digits=***n***)**	round(3.475, digits=2) is 3.48
signif(*x***, digits=***n***)**	signif(3.475, digits=2) is 3.5
cos(*x***), sin(***x***), tan(***x***)**	also acos(*x*), cosh(*x*), acosh(*x*), etc.
log(*x***)**	natural logarithm
log10(*x***)**	common logarithm
exp(*x***)**	e^*x*

R - Data Manipulation

- **Built in functions:**

Function	Description
mean(*x***, trim**=0**, na.rm**=FALSE**)**	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores mx <- mean(x,trim=.05,na.rm=TRUE)
sd(*x***)**	standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(*x***)**	median
quantile(*x***,** *probs***)**	quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in [0,1]. # 30th and 84th percentiles of x y <- quantile(x, c(.3,.84))
range(*x***)**	range
sum(*x***)**	sum
diff(*x***, lag=***1***)**	lagged differences, with lag indicating which lag to use
min(*x***)**	minimum
max(*x***)**	maximum

R - Graphics

- One of the main reasons data analysts turn to R is for its strong graphic capabilities
- **A Silly Axis Example:**

```
> # specify the data
x <- c(1:10); y <- x; z <- 10/x

># create extra margin room on the right for an axis
>par(mar=c(5, 4, 4, 8) + 0.1)

># plot x vs. y
>plot(x, y,type="b", pch=21, col="red", yaxt="n", lty=3, xlab="", ylab="")

># add x vs. 1/x
>lines(x, z, type="b", pch=22, col="blue", lty=2)

># draw an axis on the left
>axis(2, at=x,labels=x, col.axis="red", las=2)

># draw an axis on the right, with smaller text and ticks
>axis(4, at=z,labels=round(z,digits=2),col.axis="blue", las=2, cex.axis=0.7, tck=-.01)

># add a title for the right axis
>mtext("y=1/x", side=4, line=3, cex.lab=1,las=2, col="blue")

># add a main title and bottom and left axis labels
>title("An Example of Creative Axes", xlab="X values", ylab="Y=X")
```

Non interactive R: Batch Processing

```
    # save as myscript.sh

x <- c(1:10); y <- x; z <- 10/x
par(mar=c(5, 4, 4, 8) + 0.1)
plot(x, y,type="b", pch=21, col="red", yaxt="n", lty=3, xlab="", ylab="")
lines(x, z, type="b", pch=22, col="blue", lty=2)
axis(2, at=x,labels=x, col.axis="red", las=2)
axis(4, at=z,labels=round(z,digits=2),col.axis="blue", las=2, cex.axis=0.7, tck=-.01)
mtext("y=1/x", side=4, line=3, cex.lab=1,las=2, col="blue")
title("An Example of Creative Axes",xlab="X values", ylab="Y=X")

   # plot saved in the directory where the script is run, in file "Rplots.pdf"
```

- **On Linux/OS X:**

```
> q()
machine:~username$ R CMD BATCH [options] myscript.sh [outfile]
```

- **On Windows(adjust the path to R.exe and your script):**

```
"C:\[path to R]\R.exe" CMD BATCH --vanilla --slave "c:\my projects\my_script.R"
```

- **For other things: Use Google/your preferred search engine, ask via email, or in one of the next classes…**