Lesson 1

Introduction to R

Are you ready to learn a new language?
Getting started with R

1. To get acquainted with R, RStudio and Bioconductor.
2. To install R and RStudio.
3. To learn the basic elements of R: variables, vectors, functions, matrices, factors, data.frames and lists.
4. To understand the concept of object and how to see object properties. Learn to use as() function.
5. To learn punctuation symbols.
6. To learn how to use help in R
Getting started with R

1. To get acquainted with R, RStudio and Bioconductor.
2. To install R and RStudio.
3. To learn the basic elements of R: variables, vectors, functions, matrices, factors, data.frames and lists.
4. To understand the concept of object and how to see object properties. Learn to use as() function.
5. To learn punctuation symbols.
6. To learn how to use help in R
What is R?

R is a programming language and an environment for statistical computing and graphics.

R is a dialect of the S language and was initially written by Robert Gentleman and Ross Ihaka in 1993. Version 1.0.0 came out in 2000.
R is a language

As a language, R has elements, such as vectors, functions, data-frames, etc, and a syntax, or sets of rules that make commands interpretable.

R is an object-oriented language: the elements of R have properties and access functions.

R objects can be saved and are portable.

I speak English...

> for (C in c("MCB4934")) {
  print ("I speak R")
}
R is a language

As in any other language, you can say the same thing in many different ways. Some ways are more efficient than others, but other times it is just a matter of taste. Like you can say “I like spaghetti a lot“ or “I love spaghetti“ or “ Spaghetti is one of my favorite food”

Note these three ways to say “I speak R” in R:

```r
> print("I speak R")
[1] "I speak R"
> cat("I speak R")
I speak R
> a = "I"; b = "speak"; c = "R"
> print(paste(a,b,c))
[1] "I speak R"
```
Advantages and disadvantages

Many existing libraries that can be combined easily to make scripts
Great graphics quality
Powerful to work with data tables
Large contributing community

You need to learn the language
You have to keep pace with language and packages updates
Debugging is sometimes a pain!!
Where to find R?

“CRAN” is the name of the main R repository

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available [here](https://cran.R-project.org/web/documentation/). Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available [here](https://cran.R-project.org/old.html).
- Contributed extension packages
RStudio

RStudio is a free, user-friendly interface for R. It allows to organize your scripts, your data, plots, help, etc.
RStudio interface

RStudio main interface has 4 connected windows

Here you write your scripts

Here is where R actually runs

Here you can see files in your workspace

Here you visualize plots and help functions
Bioconductor

Bioconductor is a repository of R packages in the field of genome analysis. Version 3.1 contains 1,024 packages.
STOP!!

Trouble with some new terms???

**Script**: piece of text with concatenated commands that can be executed as a program.

**Directory**: You always execute R from a physical path in your computer. This is the current directory. There is where your scripts and exported files will be stored if you do not specify an alternative path.
STOP!!

Trouble with some new terms???

**Workspace**: In R, this is the memory place where your variables are stored and your commands are run.

**Package**: A small program to run a specific set of statistical functions. Packages can be downloaded from R repositories and can be used within scripts to perform elaborated analysis.
Getting started with R

1. To get acquainted with R, RStudio and Bioconductor.
2. To install R and RStudio.
3. To learn the basic elements of R: variables, vectors, functions, matrices, factors, data.frames and lists.
4. To understand the concept of object and how to see object properties. Learn to use as() function.
5. To learn punctuation symbols.
6. To learn how to use help in R
Basic R objects

R works with objects or information elements and functions, that operate on objects. There are four basic objects in R: vector, matrix, data.frame and list. Factors and arrays are also objects.

Objects have properties such as length, dimensions and are portable: can be saved and exported. Functions are also portable.
Defining objects (1)

The simplest way to create an object is by assigning values to the object. An object with one element is also known as variable. The way to assign values is with the symbols “<-” or “=”

```
> a <- "Defining object a"
> a
[1] "Defining object a"
> b = "Defining object b"
> b
[1] "Defining object b"
```

Frequently “<-” is used in object assignment and “=” is used to give values to function parameters. For example:

```
m1 <- matrix(data = 1:25)
```

Run this piece of code in RStudio by simply executing the R Class1_2.R. Do not continue the lecture before you have managed to get the script running.
Defining objects (II)

NOTE1 Variables in R can be *numeric* or *character*.

```r
> one <- 1  # one is a numeric variable here
> one + one  # numeric variables accept mathematical operations
[1] 2
> two <- "2"  # two is a character variable here
> two + two
Error in two + two : non-numeric argument to binary operator
> two2 <- as.numeric(two)  # coercing to a numeric variable
> two2 + two2
[1] 4
```

http://www.iconarchive.com/
Defining objects (III)

NOTE 2. R is case sensitive. “A” and “a” are different values.

NOTE 3. Numbers are values as such, and a number cannot be assigned a value with “<-”. Letters can be values and objects. “A” is the value and A is the object.

Valid and invalid variable names

```r
> A
[1] "A"
> "A"
[1] "A"
> A <- "A"
> A
[1] "A"
```

aaa  
aAa  
a.A  
a_a  
a1a  
123  
1;a  
a:a  
A*a  
A/1
**R objects: vectors**

*Vector*: Succession of elements of the same type. The function that creates a vector is `c()` or `vector()`

```r
> a <- c(1,2,3,4,5)
> a
[1] 1 2 3 4 5
> b <- c("A", "B", "C", "D", "E")
> b
[1] "A" "B" "C" "D" "E"
> d <- c("one", "two", "three", "four")
> d
[1] "one" "two" "three" "four"
> empty.vector <- vector("numeric", length = 4)
> empty.vector
[1] 0 0 0 0
```
R objects: factors

A **factor** is a vector of categorical values, meaning that all identical values are given the same level.

```r
> fact1 <- factor(c("one", "two", "three", "two"))
> fact1
[1] one two three two
Levels: one three two
> fact2 <- factor(c(1, 2, 3, 2))
> fact2
[1] 1 2 3 2
Levels: 1 2 3
> fact2 + fact2
[1] NA NA NA NA
Warning message:
In Ops.factor(fact2, fact2) : '+' not meaningful for factors
```

Factors do not accept mathematical operations but can be ordered.
R objects: matrix

A **matrix** is an array of elements of the same type arranged in rows and columns.

A **matrix** can be created with the function `matrix()` or the function `cbind()`.

```r
> m1 <- matrix(data = 1:20, nrow = 4, ncol = 5, dimnames = list(d,b))
> m1
   A  B  C  D  E
one  1  5  9 13 17
two  2  6 10 14 18
three 3  7 11 15 19
four 4  8 12 16 20
> m2 <- cbind(A = c(1:4), B = c(5:8), C = c(9:12), D = c(13:16), E = c(17,20))
> rownames(m2) = d
> m2
   A  B  C  D  E
one 1  5  9 13 17
two 2  6 10 14 20
three 3  7 11 15 19
four 4  8 12 16 20
```
R objects: data.frame

Concatenation of vectors that can be of different type but have the same length

```r
> df <- data.frame(letters = b, numbers = a)
> df
  letters numbers
     A      1
     B      2
     C      3
     D      4
     E      5
```
A **list** is an ordered collection of objects that can have different types and sizes.

```r
> ll <- list(vector1 = a, vector2 = b, matrix = m1)
> ll

$vector1
[1] 1 2 3 4 5

$vector2
[1] "A" "B" "C" "D" "E"

$matrix
     A B C D E
one  1 5 9 13 17
two  2 6 10 14 18
three 3 7 11 15 19
four 4 8 12 16 20
```

The elements of the list are returned starting with the `$` symbol.
R functions

Functions are a collection of commands that perform a data analysis task

```r
CV <- function(x) {
  m <- mean(x)
  s <- sd(x)
  cv <- m/s
  return(cv)
}
> CV(a)
[1] 1.897367
```

This function defines the mathematical expression to compute the coefficient of variance of a numeric vector.

Once you define a function, you can use it the same way as any other R function.

You can see the code of a function by typing the function name without brackets at the console.

```r
> CV
```
Object properties (I)

R objects have properties that can be called or modified with specific R functions
mode() gets or set the storage mode

mode(a)  # numeric
mode(b)  # character
mode(fact1)  # numeric
mode(m1)  # numeric
mode(df)  # list
mode(ll)  # list
mode(CV)  # function
Object properties (III)

class() defines the access functions on the object

class(a)  # numeric
class(b)  # character
class(fact1)  # factor
class(m1)  # matrix
class(df)  # data.frame
class (ll)  # list
class (CV)  # function
Object properties (IV)

length() is the number of elements in the object

length(a) # 5
length(b) # 5
length(fact1) # 4
length(m1) # 20
length(df) # 2
length(ll) # 3
length(CV) # 1

The length of a matrix is the number columns x number rows
The length of a data.frame is the number of columns
The length of a list is the number of elements is the list
The length of a function is 1
Object properties (III)

dim() returns the dimensions of the object

dim(a) # NULL
dim(b) # NULL
dim(fact1) # NULL
dim(m1) # 4 5
dim(df) # 5 2
dim(ll) # NULL
dim(CV) # NULL
Object properties (V)

attributes() lists the object attributes

```r
> attributes (m1)
$dim
[1] 4 5

$dimnames
$dimnames[[1]]
[1] "one" "two" "three" "four"

$dimnames[[2]]
[1] "A" "B" "C" "D" "E"
```

The attributes of a matrix are its dimensions and the names of its two dimensions: names of rows and names of columns
Using the `as()` function

`as()` is a very useful function to coerce an R object of a type to another type.

```r
> a
[1] 1 2 3 4 5
> a2 <- as.character(a)
> a2
[1] "1" "2" "3" "4" "5"
```

```r
class(fact2)
[1] "factor"
> fact2
[1] 1 2 3 2
Levels: 1 2 3
> fact2 * 2
[1] NA NA NA NA
Warning message:
In Ops.factor(fact2, 2) : '*' not meaningful for factors
> fact21 <- as.numeric(fact2)
> class(fact21)
[1] "numeric"
> fact21 * 2
[1] 2 4 6 4
```

coercing a factor to a numeric vector to get mathematical properties

coercing a numeric vector to character
Punctuation symbols: brackets

Brackets `[ ]` are used in R for calling specific elements of an object.

```R
> a[1]  # gets the first element of a
[1] 1
> df[3,2]  # gets element at row 3 and column 2 of df
[1] 3
> m1[,2]  # gets all elements at column 2 of m1
  one two three four
  5   6   7   8
> ll[2]  # gets element 2 of list ll
$vvector2
[1] "A" "B" "C" "D" "E"

> ll[[2]]  # gets the actual elements of element 2 of list ll
[1] "A" "B" "C" "D" "E"
> ll[[2]][3]  # gets element 3 of element 2 of this ll
[1] "C"
```
Punctuation symbols: curly brackets

Curly brackets {} is used to indicate the boundaries of a function or loops.

```
CV <- function (x) {
  m <- mean(x)
  s <- sd(x)
  cv <- m/s
  return(cv)
}
```

The CV function is defined between these curly brackets.
Punctuation symbols: colon

Colon : is used to denote all numbers in between two given numbers

> e <- 1:5  # e is assigned the vector a values between 1 and 5
> e
[1] 1 2 3 4 5
> df[1:3,]  # get rows 1 to 3 of data.frame df
  letters numbers
   1   A   1
   2   B   2
   3   C   3

http://www.iconarchive.com/
Punctuation symbols: semi-colon

Semi-colon ; is used to separate commands within the same line of code. You should use this only sporadically. Here some examples:

```r
mode(a); mode(b); mode(fact1)  # several simple commands in a row
e <- 1:5 ; e # define a variable and call it
```

Number symbol # is used to comment code. All what it is written after # in a row will not be executed. In RStudio commented code appears in green

```r
e <- 1:5  # this command is executed but the comment is not
# e <- 1:5  this command is not executed and also not the comment
```
Getting started with R

1. To get acquainted with R, RStudio and Bioconductor.
2. To install R and RStudio.
3. To learn the basic elements of R: variables, vectors, functions, matrices, factors, data.frames and lists.
4. To understand the concept of object and how to see object properties. Learn to use as() function.
5. To learn punctuation symbols.
6. To learn how to use help in R
Getting help in R

There are three major sources of help in R

# Help about functions within R
# Official R manuals
# Discussion lists and internet

http://www.iconarchive.com/
Help within R

There are several commands that get you help within R

help (function)  # gets help on a defined function
?function    # the same as above
help (package = ggplot2)  # all functions in a package
help.search ("function")  # search on a keyword
apropos ("function")  # search on a fuzzy keyword
example (function)  # runs function examples
vignette (package="packagename")  # the vignette is a tutorial on the package

You can practice R help with the script Class1_3.R script available at CANVAS.
Official R manuals
Web R help

FAQ at CRAN or Bioconductor:
https://cran.r-project.org/
http://www.bioconductor.org/help/faqs

R mailing lists
R-help@stat.math.ethz.ch

Bioconductor support
https://support.bioconductor.org/

Mailing lists achieves at R and Bioconductor sites
Just type in Google: R help concept
Mailing Lists

Please read the instructions below and the posting guide before sending anything to any mailing list!

Thanks to Martin Maechler (and ETH Zurich), there are five general mailing lists devoted to R.

R-announce

This list is for major announcements about the development of R and the availability of new code. It has a low volume (typically only a few messages a month) and everyone mildly interested should consider subscribing, but note that R-help gets everything from R-announce as well, so you don't need to subscribe to both of them.

Note that the list is moderated to be used for announcements mainly by the R Core Development Team. Use the web interface for information, subscription, archives, etc.

R-help

The 'main' R mailing list, for discussion about problems and solutions using R, announcements (not covered by 'R-announce' or 'R-packages', see above), about the availability of new functionality for R and documentation of R, comparison and compatibility with S-plus, and for the posting of nice examples and benchmarks. Do read the posting guide before sending anything!

This has become quite an active list with dozens of messages per day. An alternative is to subscribe and choose daily digests (in plain or MIME format). Use the web interface for information, subscription, archives, etc.
### Bioconductor support site

#### Recent...

<table>
<thead>
<tr>
<th>Title</th>
<th>Views</th>
<th>Votes</th>
<th>Answers</th>
<th>Created At</th>
<th>Author(s)</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biostrings pairwiseAlignment() output shorter than source sequence</td>
<td>34</td>
<td>0</td>
<td>2</td>
<td>1 day ago</td>
<td>jean.peccoud</td>
<td>0</td>
</tr>
<tr>
<td>RUVseq for microRNAs</td>
<td>30</td>
<td>0</td>
<td>2</td>
<td>18 days ago</td>
<td>sergio.espeso-gil</td>
<td>0</td>
</tr>
<tr>
<td>GEOquery: 0 features in the gset object</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>12 hours ago</td>
<td>alakatos</td>
<td>50</td>
</tr>
<tr>
<td>DESeq2: Abnormally skewed distribution of p-values</td>
<td>36</td>
<td>1</td>
<td>2</td>
<td>1 day ago</td>
<td>Gilgi</td>
<td>0</td>
</tr>
<tr>
<td>SCAN.UPC giving error for multiple input files</td>
<td>9</td>
<td>0</td>
<td>3</td>
<td>7 hours ago</td>
<td>Vani</td>
<td>10</td>
</tr>
<tr>
<td>RDAVIDWebService Error 301: Moved Permanently</td>
<td>368</td>
<td>3</td>
<td>10</td>
<td>7 hours ago</td>
<td>Vivek.b</td>
<td>20</td>
</tr>
<tr>
<td>WGCNA network creation error</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>10 hours ago</td>
<td>akridgerunner</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Votes
- GEOquery: 0 features in the...: 50 votes
- RUVseq for microRNAs: 30 votes
- DESeq2: Abnormally skewed distribution of p-values: 36 votes
- SCAN.UPC giving error for multiple input files: 9 votes
- RDAVIDWebService Error 301: Moved Permanently: 368 votes
- Biostrings pairwiseAlignment() output shorter than source sequence: 34 votes
- GEOquery: 0 features in the...: 16 votes

#### Awards
- Scholar @ to Julian Gehring: 940 points
- Scholar @ to ilari.scheinin: 10 points
- Scholar @ to Gordon Smyth: 24k points
- Scholar @ to Matthew Ritchie: 570 points
- Scholar @ to Michael Lawrence: 6.7k points
- Teacher @ to Aaron Lun: 3.2k points
End of Introduction

We will move now to RStudio and start speaking R...