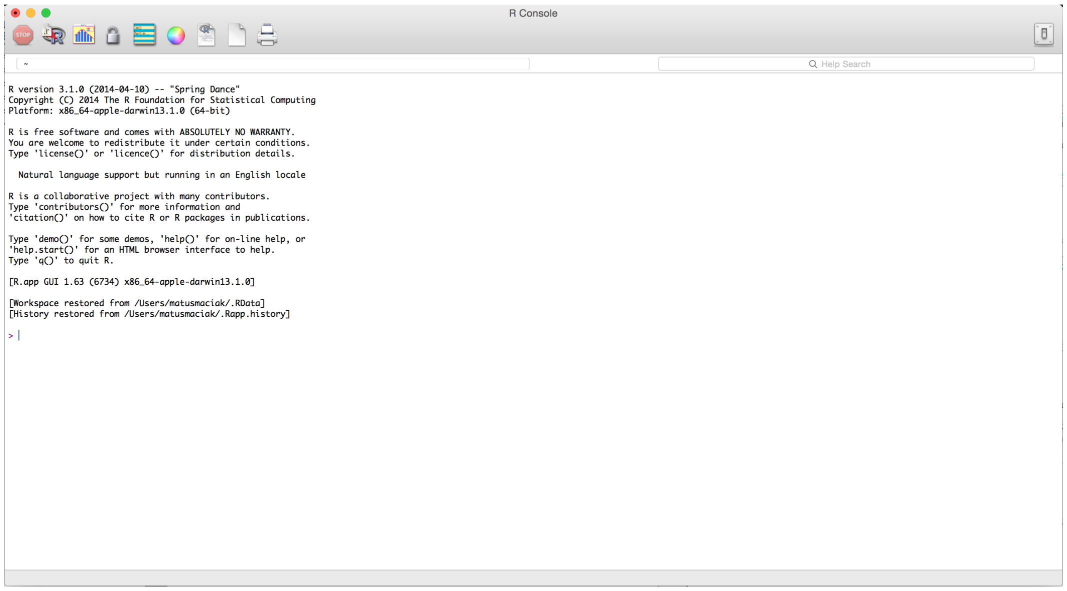
A Very Brief Introduction to Statistical Software R

M. Maciak (Czech University of Life Sciences, Prague)

Lab Session 1 - Summer Term 2015

1 Download & Installation

- the software can be obtained free of charge from `http://www.r-project.org`
  ↪ corresponding installation packages are available for all three platforms (Win, Linux and OS X)
- R is a GNU GPL interactive object-based programming tool for statistical calculations;
- additional graphical interfaces: R-Studio `http://www.rstudio.com`
  R-Commander `http://www.rcommander.com`
  JGR `http://www.rforge.net/JGR/`
- a huge variety of additional packages free to download (over 5000 packages available online at the CRAN mirrors `http://cran.r-project.org/mirrors.html`);
  ↪ they are meant for more sophisticated statistical calculations, no need to install any of those;
- a detailed introduction (online help) available at `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`
  ↪ some other brief introductions are also available at Moodle;
- a detailed installation manual for Windows is also available at Moodle;
- once you install the software and click on the corresponding icon the R environment should automatically open; It will look somehow like this:



```
R version 3.1.0 (2014-04-10) -- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.1.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.63 (6734) x86_64-apple-darwin13.1.0]

[Workspace restored from /Users/matusmaciak/.RData]
[History restored from /Users/matusmaciak/.Rapp.history]

> 
```

2 Basic Description of the R environment

- Now, it is all ready for some first calculations...
- The R software interface consists of three parts (R Console, R Graphics and R Editor), however, not all of them are automatically visible once you open the environment;
  - R Console is used for typing in all necessary commands (visible by default);
  - R Graphics presents graphical outputs, plots (figures) - only visible when you plot something;
  - R Editor is used to keep a record of all the work (commands, comments, etc.) - visible once you manually open it from the main menu;
- R Console can by immediately used s a simple calculator... **However, it can do much much more...**

3 Some Standard Mathematical Calculations in R

- try some basic mathematical calculation - type them into R Console and press Enter to confirm; You will get the result directly on the screen;
- standard mathematical operators are $+$, $-$, $*$, and $/$;

```
> 2 + 4
> 60 - 12
> 3 * 12
> 60/15
>
> 3^2 # three to the second power
> 2^10 # two to the power of ten
> sqrt(81) # square root of 81
```

- note, that anything which follows after # symbol (on the given line) is ignored by the software; Thus, you can use it to enter some notes, comments, etc.;
- to keep the result rather than just printing it on the screen you can save it ito some well defined object using "$<-$" or "$=$" symbol:

```
> result1 <- 2 + 4
> result2 <- 3 * 12
>
> myFirstResult_001 <- sqrt(1200 / 4)
```

- try to see what happens now if you type `result1` or `result2` in the R Console.
- it is possible to use your previous results for further calculations:

```
> (result1 + result2)/2
> (result2 + 100)^result1
```

  and to store it as another result (new object):

```
> newResult <- result1 + result2
```

- use the R help to see what are these functions for: `exp()`, `log()`, `abs()`;
  Try to use them for some calculations and store your results in `result3` and `result4`;

- beside a classical mathematical calculations you can use any of pre-programmed functions in R (e.g. `mean()`, `sum()`, `min()`, `max()`, etc.);

```
> max(1,2,3,4,5)
> sum(10,20, 20, 40)
> mean(1,1,2,2,3)
```

- for an instant help on some command or function in R, just type in the R console a question mark followed by the name of the command or function (e.g. `?mean`, or `?min()`)
- for a hypertext help (online help which will be opened in your browser) you can use command `help.start()` (e.g. `help.start(mean)`);
- **All functions pre-defined in R Software are very intuitive - names of the functions are directly derived from English $\rightarrow$ to sum some numbers $\Rightarrow$ function `sum()`; to get a mean value of same values $\Rightarrow$ `mean()`; to get minimum value out of some numbers $\Rightarrow$ `min()`; etc.**

4 Putting more numbers together - Vectors

- sometimes it is convenient to consider more numbers at once and to apply some mathematical operations with all of them at once; That is the reason for using vectors.
- in R there are various pre-defined commands (functions) to create vectors (e.g. `c()` which comes from 'column', `seq()` which comes from 'sequence', `rep()` which comes from 'repeat', and others);
Use the R help (e.g. `?c`, `?seq`) to see what they do...
- it is important to know what calculations are possible with vectors and how does it work;
Try the following and try to figure out the logic used behind the results:

```
> 1:10
> seq(1,10)
> seq(1,10, length=100)
> seq(1,10, by = 0.05)
>
> c(1, 2, 3) + 1
> c(1, 2, 3) + c(1, 1, 1)
> c(1, 2, 3) + c(4, 5, 6) + 7
> c(1, 2, 3) * 2
> c(1, 2, 3) * c(1, 2, 3)
```

- what happens if you try to type in the following: `c(1, 2, 3) + c(1, 1)`
- to check the vector's length use command `length()`;
- use the combination of vector creating commands and see how it works together:

```
> rep(c(result1, result2, result3), 3)
> rep(seq(result1, result2), 2)
> rep(seq(result1, result2, length=4), 3)
```

- to store the results, use "$<-$" or "$=$" symbol again

```
> vector1 <- rep(c(result1, result2, result3), 3)
> Vector1 <- rep(seq(result1, result2), 2)
> Vector2 <- rep(seq(result1, result2, length=4), 3)
```

- to see what objects are already used, see the list with `ls()` command; to delete some objects, use `rm()` - e.g. `rm(Vector2)`
- to refer to a specific element of some vector rather than the whole vector itself, use the corresponding index:

  ```
  > first_element <- vector1[1]
  > second_element <- vector1[2]
  ```

  or a list of some specific elements

  ```
  > vector1[1:5] # returns the first five elements of vector1
  > vector1[c(2,4,6,8)] # returns the second, fourth, sixth and eight element
  ```

- notice, that everything that follows after # is ignored by the software. You can use it to comment your code, to make more clear to another users what is going on in some specific part of the code;

  ```
  > # now, we are calculating the mean of vector1
  > mean(vector1)
  >
  > # using the definition of mean we obtain the same result by typing in
  > sum(vector1) / length(vector1) # explain why?
  >
  > # the last element of vector1 is
  > vector1[length(vector1)]
  ```

5 Working with Matrices

- in statistics it will come handy to even consider more vectors together ⇒ matrices;
- use the R help to see what are the following commands for: `rbind()` which comes from 'bind rows', `cbind()` which comes from 'bind columns', `matrix()` and `t()` which comes from 'transpose';
- try to create a matrix out of two arbitrary vectors
  (hint: use the following symbol for the matrix multiplication: $\% * \%$)

  ```
  > matrix1 <- c(1,2,3) %*% t(c(1,2))
  ```

- we can again refer to specific elements of rows (columns) of the matrix:

  ```
  > # the first element (top left) of matrix1 is
  > matrix1[1, 1]
  > # the first row in matrix1 is
  > matrix1[1, ]
  > # the second column in matrix1 is
  > matrix1[ , 2]
  ```

- to check the matrix dimension use command `dim()`, to print the matrix, type `matrix1` again;
- in R one can easily change the matrix entries, or to change and expand the matrix:

  ```
  > matrix1[1,1] <- 0 # replaces the first element in matrix (top left) with zero
  > matrix1[1, ] <- 2 # replaces the whole first row with twos
  > matrix1[, 2] <- c(2,4,6) # replaces the whole second column with new vector
  >
  > big_matrix <- cbind(matrix1, matrix1)
  > small_matrix1 <- big_matrix[1:3, 1:3] # takes a submatrix out of a bigger matrix
  > small_matrix2 <- big_matrix[c(1,3), c(1,3)]
  ```

6 Working with Datasets

- there are many different datasets already available in R packages; you can use them to practice your skills... see the list of available data by typing `data()` into the R console;
- to get a brief description on some specific data, use the question mark (as calling for R help) with the name of the data set (e.g. `?Orange` or `?CO2`); print the data on screen by typing the name of the dataset (e.g. `Orange`, or `CO2`);
- you can create your own dataset using command `data.frame()`: it is a special type of a matrix, where columns can be of different types, however, the same length; rows represent observations and columns represent covariates;

```
> dataset1 <- data.frame(cbind(c("male", "female", "female"), big_matrix))
> # we can also give names to all columns:
> names(dataset1) <- c("gender", "covariate1", "covariate2",
                              "covariate3", "covariate4")
```

- in a similar way we can also assign names to observations:

```
> dataset2 <- data.frame(dataset1, row.names = c("observation1",
                                                 "observation2",
                                                 "observation3" ))
```

  See the difference between dataset1 and dataset2 by typing in `dataset1` and `dataset2` respectively; it is possible to refer to dataset's covariates by calling their names after attaching the dataset:

```
> attach(dataset2)
> names(dataset2)
> gender
> # similarly, the same output is obtained by typing in
> dataset2[ , 1] # which prints the first column of dataset2, which is gender
```

- use some dataset (your own or some from the list of available ones in R) to get some basic statistical characteristics for your data;
- try the command `summary()` when called on some dataset (e.g. `summary(CO2)`)
- use the R help to see what are the following commands for: `mean()`, `median()` `var()`, `sd()`, `min()`, `max()`, `sort()`, `order()`
- try to get again the sample mean value, sample median, variance, standard error, minimum and maximum value, however, avoid using `mean()`, `median()`, `var()`, `sd()`, `min()` and `max()` commands;