



UNIVERSITY OF
BIRMINGHAM

R for Bioinformatics and HDS

R & RStudio:
R Basics



Vasileios Panagiotis Lenis
Laura Bravo Merodio

Course Structure

- Introduction to R & RStudio
- Syntax, Comments, Variables, Data Types and Operators
- Conditions, Loops, Functions and Data Structures
- Working with Data frames



Where to type our commands?

- On the console panel we want to run a command "on fly".
On the left side of your console panel, there is the "**>**" symbol that is called "prompt". Type your command and click "**enter**" to submit it.
- On the script panel in case to keep your commands in a script file.
Type each command in a new line and use the "**run**" option to execute them.
(More instructions later)



R as a calculator

- The simplest thing you could do with R is to do arithmetic:

```
100 + 1
```

- R will print out the answer, with a preceding [1]. Don't worry about this for now, we'll explain that later. For now, think of it as indicating output.

```
[1] 101
```



R as a fancy calculator

- You can make more complicated calculations:

```
3 + 5 * (2 ^ 2)      # if you forget some rules, this might help
```

- The text after each line of code is called a “comment”. Anything that follows the hash (or octothorpe) symbol **#** is ignored by R when it executes code.



Types of Data

- R supports many different types of data with the most important:
 - Integer (e.g., 10)
 - Float point of numbers (e.g., 10.25)
 - Character e.g., “B”)
 - String (e.g., “Bioinformatics”)
 - Logical (TRUE or FALSE)



What is a variable?

- Variables are objects in R that you can use to store values. It can consist of a single value, basic or complex arithmetic operations, or even be more complex such as a column in a data matrix or a data frame.
- The **assignment** to a variable can be done in 2 different but equivalent ways, using either the “`<-`” or “`=`” operators.



Using variables

- Assigning a variable

```
weight_kg <- 65.0
```

- Making calculations with a variable

```
2.2 * weight_kg
```

- Displaying variables

```
print(weight_kg)
```



Variables as **sticky notes**!

- A variable is analogous to a sticky note with a name written on it: assigning a value to a variable is like putting that sticky note on a particular value.

```
# There are 2.2 pounds per kilogram  
weight_lb <- 2.2 * weight_kg
```

65.0

143.0



Variables as **sticky notes**!

- Let's now change the `weight_kg`:

```
weight_kg <- 100
```

- What will be the value of the `weight_lb`???
- Since `weight_lb` doesn't "remember" where its value comes from, it is not updated when we change `weight_kg`.

100.0

weight_kg

143.0

weight_lb



What name can a variable have?

A variable can have a short name (like x and y) or a more descriptive name (age, mass, volume).

Rules for R variables are:

- A variable name must start with a letter and can be a combination of letters, digits, period(.) and underscore(_). If it starts with period(.), it cannot be followed by a digit.
- A variable name cannot start with a number or underscore (_)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- Reserved words cannot be used as variables (TRUE, FALSE, NULL, if...)



What will be the value?

```
mass <- 47.5
```

```
age <- 122
```

```
mass <- mass * 2.0
```

```
age <- age - 20
```



Console is great, but scripting is much better!

- You can use the R script that you have created to:
 - Store your commands
 - Put some comments (free-text lines that start with "#")
 - Re-use it!
- An R script can be simple, or... quite complicated!

```
#this is the script for the transcriptomics-proteomics analysis with the
#NEW dataset of Transcriptomics (file: PAPER/Meningioma_protein_differentially_expressed_genes.R

## load expression data

TRex <- read.csv("~/Documents/Trans_Prot_exprMatrices/GSE74385_expressionMatrix.txt",sep="\t"
PReg <- read.csv("~/Documents/Trans_Prot_exprMatrices/JProteomics_expressionMatrix.txt",sep=",")

length(unique(as.character(TRex$Symbol))) # 31385

PRegAverage <- apply(PReg[,3:27],1,mean)
names(PRegAverage) <- PReg$Symbol
#####  
#####
druggable <- read.csv("Matthias_Analysis/DruggableProteome.txt",sep="\t",header=TRUE)
#####
LFC13T <- read.csv("Trans_Clean.txt", sep = "\t", header = T)
LFC13P <- read.csv("Matthias_Analysis/GIvs3P.rnk", sep = "\t", header = T)
LFC13T <- read.csv("Trans_Clean.txt", sep = "\t", header = T)
LFC13P <- read.csv("Matthias_Analysis/GIvs3P.rnk", sep = "\t", header = T)
```



How to “run” the commands of a script?

The image shows a screenshot of the RStudio interface. At the top, there is a code editor window titled "Untitled1" with the following R code:

```
1 x <- "Vasilis"
2 print(x)
3
```

Two blue arrows point upwards from the text boxes below to the "Run" button in the top right of the RStudio interface. A red arrow points to the "Source on Save" button in the top left of the code editor. To the right of the code editor is the "Environment" tab of the global environment pane, showing a variable "x" with the value "Vasilis".

1. Put the cursor in the line that you want to execute

2. Then click “Run”

3. The command will be executed in the console and the output will be printed.

4. Don't forget to save the script and choose an appropriate name

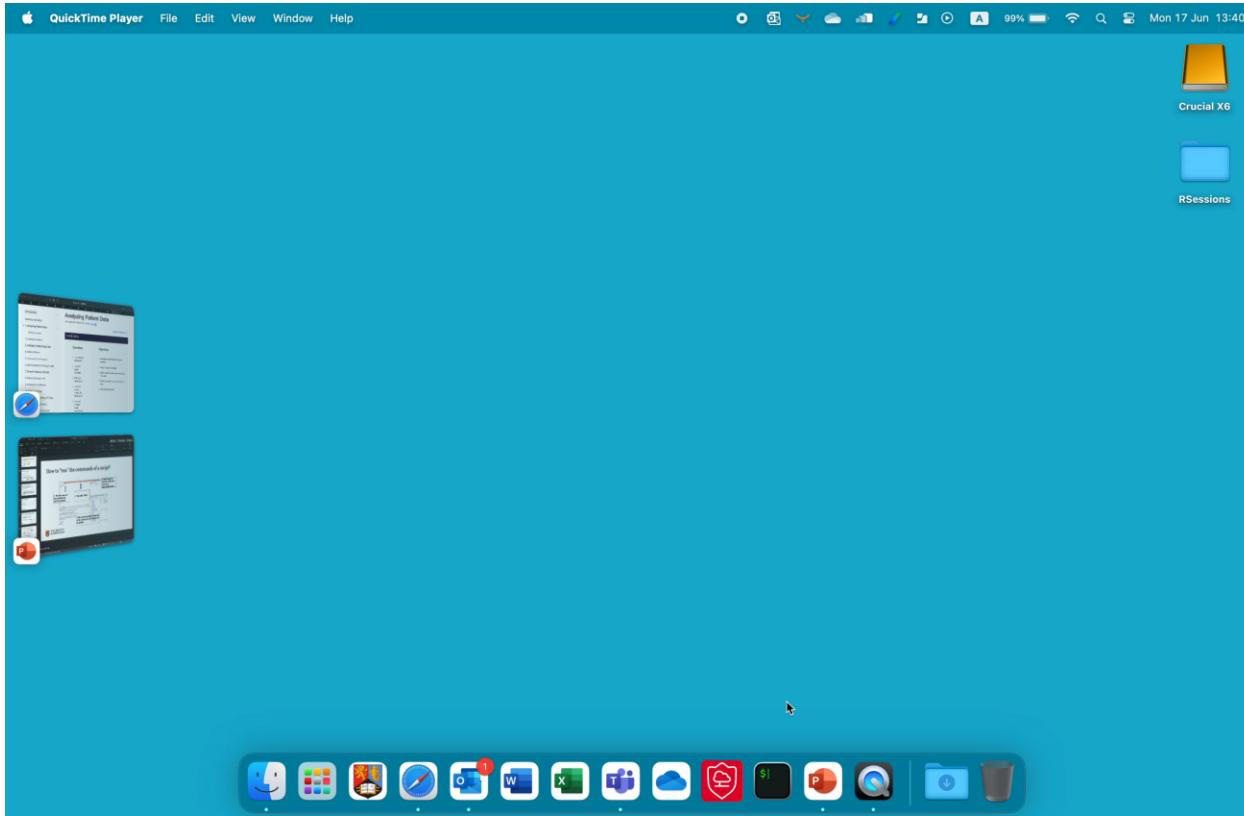
The RStudio interface also includes a "Console" tab at the bottom left showing the command and its output:

```
> x <- "Vasilis"
> print(x)
[1] "Vasilis"
```

Below the RStudio interface is a file browser window showing the directory structure of the current workspace:

Name	Size	Modified
.gitignore	40 B	Nov 14, 2017, 1:1
.Renviron	17 B	Apr 2, 2019, 4:2
.Rhistory	8 KB	Jan 23, 2024, 4:0
anaconda3		
AnacondaProjects		
Applications		
CytoscapeConfiguration		
DataStructures_2.ipynb	4.3 KB	Sep 27, 2022, 9:1
Day_1		
Day_2		
Desktop		
Downloads		
eclipse-workspace		
example.nem	1.6 KB	Apr 18, 2018, 4:1





UNIVERSITY OF
BIRMINGHAM



v.p.lenis@bham.ac.uk (Vasilis)
l.bravo@bham.ac.uk (Laura)

