

Tutorial and Laboratories

11520G Data Capture and Preparations

Week 2

Introduction

This tutorial is designed to get you started with the statistical programming language R and the RStudio Interface. R is an open-source, statistical analysis software. You can work directly in R but we recommend using RStudio, a graphical interface. RStudio is an open-source, integrated development environment (IDE) for R. RStudio combines a powerful code/script editor, special tools for plotting and for viewing R objects and code history, and a code debugger. This tutorial is suitable for those who have not worked with R/Rstudio before.

Skills Covered in this tutorial include:

- How to install RStudio
- Using the RStudio IDE
- Basic operations in R

Why learn R?

- The style of coding is very easy to learn.
- It's open source. No need to pay any subscription fees or licences.
- R is a comprehensive statistical platform, offering all manner of data-analytic techniques.
- Hundreds of packages are available for various computational tasks. New methods become available for download on a weekly basis.
- There is a large community support. There are numerous forums to help you out.
- R has state-of-the-art graphics capabilities.
- One of highly sought skill by analytics and data science companies.

1. How to install R/Rstudio

R and Rstudio are already installed on all the computers in the lab, so you can skip this section. If you require to install R and Rstudio in your laptop, please follow the steps below:

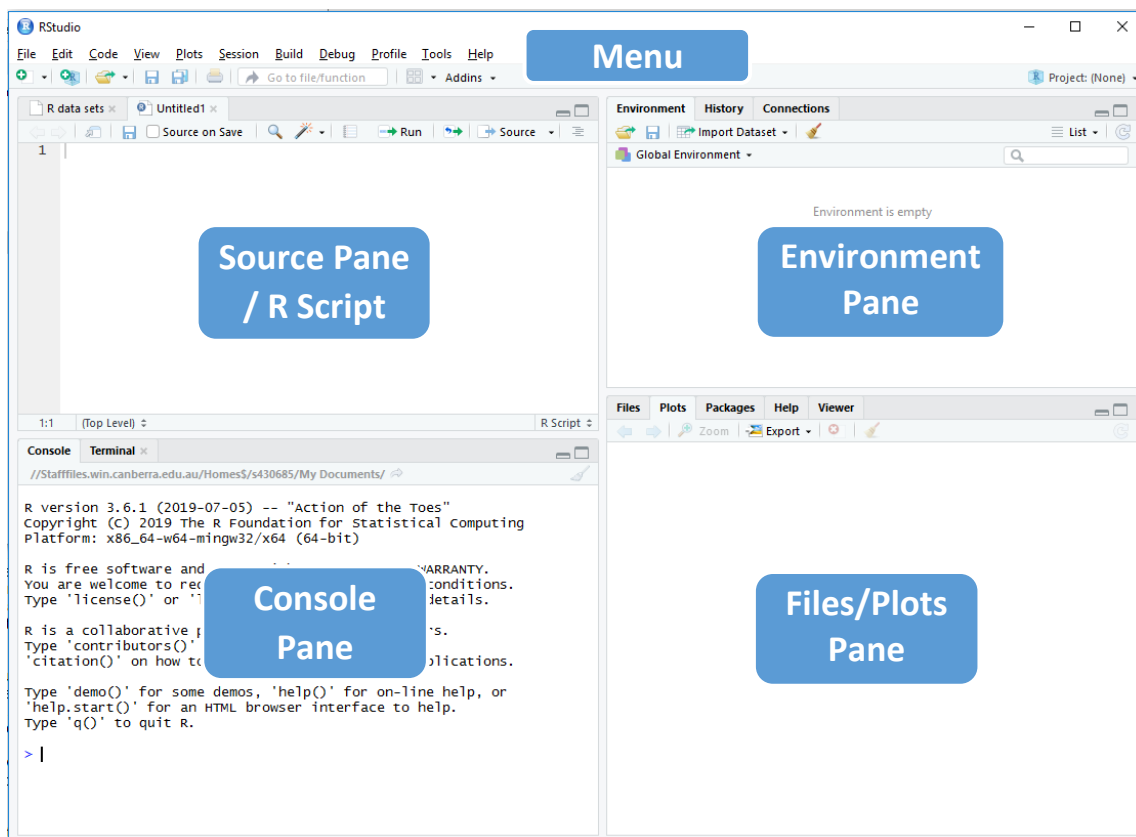
1. To download R go to <https://cran.r-project.org/bin/windows/base/>
2. Click on "Download R 4.0.4 for Windows" (or latest) and save the file. For different operating systems use: <https://cran.r-project.org/bin/>.
3. Once download is completed, run .exe file and follow the instructions.
4. To download RStudio go to: <https://www.rstudio.com/products/rstudio/download/>
5. In 'Choose Your Version', section, choose and click 'Download' under Rstudio Desktop (Open Source Licence).

6. Choose the correct installer based on your operating system. The download should begin as soon as you click.
7. Once download is completed, run .exe file and follow the instructions.


Please note: In this unit, we use RStudio on Windows to do all of our programming practices. The instructions in labs and tutorials assume you are working on Windows operating system, so if you are using any other operating system, please adjust accordingly.

2. Starting Rstudio



To Start RStudio, click on its desktop icon or use 'search windows' to access the program. When you first open Rstudio, you will see the Menu, the Console Pane, the Environment Pane, and the Files/Plot Pane. You should now see the following window:



The Console pane: This shows the output of code you run. Also, you can directly write line-by-line commands in console. Code entered directly in R console cannot be traced later. This is where script area comes to use.

The Source pane/script area: The Source Pane is a text editor where you can type your code before running it. You can save your code in a text file called a script. Scripts have typically file names with the extension .R. To run those lines, simply select them and press (Ctrl + Enter). Alternatively, you can click on  **Run** button location at top right corner of script area.

Environment pane: The Environment Pane includes an Environment and a History tab. This space displays the set of external objects added. This includes data set, variables, vectors, functions etc. To check if data has been loaded properly in R, always look at this area.

Files and plots pane: Although, the Files Pane includes several tabs, we will focus (for now) on two tabs, the files tab and the plots tab. 1) The Files tab displays the contents of your working directory. R reads files from and saves files into the working directory. You can find out which directory R is using by typing `getwd()` in the command line. To change the working directory, type `setwd("H:/IntroR")` in the command line. Notice that you need a forward slash and not a backslash (/ instead of \) when setting the working directory. 2) The Plot tab shows all graphs that you have created. If you have multiple plots, you can navigate through them by clicking  and . 3) This pane also offers a list of packages to select, help with embedded R's official documentation, and a viewer tab that displays HTML output.

Before we start with our examples, let's create a folder where we can save our scripts. Please do the following:

- Create a sub-directory named "Lab_w2" in your "Documents" folder
- From RStudio, use the menu to change your working directory under Session > Set Working Directory > Choose Directory.
- From the console you can use the following command:

```
> setwd("/path/to/my/directory")
```

3. Getting Started

R is case-sensitive, interpreted language. You can enter commands one at a time at the command prompt (>) or run a set of commands from a source file. There are a wide variety of data types, including vectors, matrices, data frames (similar to datasets), and lists (collections of objects).

Basic Computations in R. To get familiar with R coding environment, start with some basic calculations. R console can be used as an interactive calculator too.

Exercise 1. Type the following in your console and observe the results:

```
> 8 + 23
> (11*2)/(6/3)
> sqrt(121)
> sum(1:5)
> print("hello world")
> x <- 7+5
> y <- 16
> x <= 5
> y == 16
> x != 7
```

Objects. Most functionality is provided through built-in and user-created functions and the creation and manipulation of objects. An object (a variable) is basically anything that can be assigned a value (e.g., `object_name <- expression`). For R, that is just about everything (data, functions, graphs, analytic results, and more). Every object has a class attribute telling R how to handle it.

All objects are kept in memory during an interactive session. Basic functions are available by default. Other functions are contained in packages that can be attached to a current session as needed.

The object name can have letters, numbers, underscores, and dots. All the following names are valid in R: *var_name*, *VarName*, *var.name*, *var2*. Names can never start with a number, so *2var* is not a valid name. We suggest selecting one naming style and using it consistently. In this tutorial, we will use the underscore-separated naming style, e.g. *var_name*.

Statements consist of functions and assignments. R uses the symbol `<-` for assignments, rather than the typical `=` sign. For example, the statement

```
X <- rnorm(5)
```

creates a vector object named *x* containing five random deviates from a standard normal distribution. R allows the `=` sign to be used for object assignments. But you won't find many programs written that way in R, because it's not standard syntax.

Exercise 2. Assign multiple numbers to one object, use the function `c()` and observe the results:

```
> my_variable <- 3
> my_variable^2
> 5 -> a
> b <- 4
> y <- c(a,b)
> k <- c(.5, 1)
> k <- c(k, 1.5)
> k <- c(k, c(2, 2.5))
> k <- c(k, k)
> x <- c(2,8,3)
> y <- c(6,4,1)
> x+y
> x>y
```

Note: To display the contents of an object, type the object name in the command line or `print(name)`. For example, type *k* in the command line or `print(k)`.

Data Types. some of the basic datatypes on which the R-objects are built are: Numeric, Integer, Character (text), Factor, and Logical.

- Numeric: Numbers that have a decimal value or are a fraction in nature have a data type as numeric. Example: `num <- 1.2`
- Integer: Numbers that do not contain decimal values have a data type as an integer. However, to create an integer data type, you explicitly use `as.integer()` and pass the variable as an argument. Example: `int <- as.integer(2.2)`
- Character: As the name suggests, it can be a letter or a combination of letters enclosed by quotes is considered as a character data type by R. It can be alphabets or numbers. Example: `char1 <- "datacapture"` or `char2 <- "12345"`

- Logical: A variable that can have a value of True and False like a boolean is called a logical variable. Example: `log_true <- TRUE` or `log_false <- FALSE`
- We can use the `==` to test for equality in R, example: `my_variable == 3` or `my_variable == 9`
- Factor: They are a data type that is used to refer to a qualitative relationship like colors, good & bad, course or movie ratings, etc. They are useful in statistical modeling. Example: `fac <- factor(c("good", "bad", "ugly", "good", "bad", "ugly"))`
- You can check the data type of an object using `class()`.
- you can check all the variables that have been defined by you in the working environment by using keyword `ls()`.
- If you want to remove a variable for the working environment, use the command `rm(name_of_variable)`.
- If you want to remove all the variables in the environment, use the command `rm(list = ls())`

Commenting. Comments are preceded by the `#` symbol. Any text appearing after the `#` is ignored by the R interpreter.

Exercise 3. Write a R program to take input from the user (name and age) and display the values. Also print the version of R installation.

```
> name = readline(prompt="Input your name: ")
> age = readline(prompt="Input your age: ")
> print(paste("My name is",name, "and I am",age , "years old."))
> print(R.version.string)
```

Vectors. We can create 1D data structures called vectors.

Exercise 4. Try the following examples and observe the results.

```
> 1:10
> 2*(1:10)
> seq(0, 10, 2)
> myvector <- 1:10
> 2^myvector
> b <- c(3,4,5)
> b^2
> disorders <- c("autism", "ocd", "depression", "ocd", "anxiety", "autism")
```

Exercise 5. Say that you're studying physical development and you've collected the ages and weights of 10 infants in their first year of life (see table 1.1). You're interested in the distribution of the weights and their relationship to age.

Table 1.1 The ages and weights of 10 infants

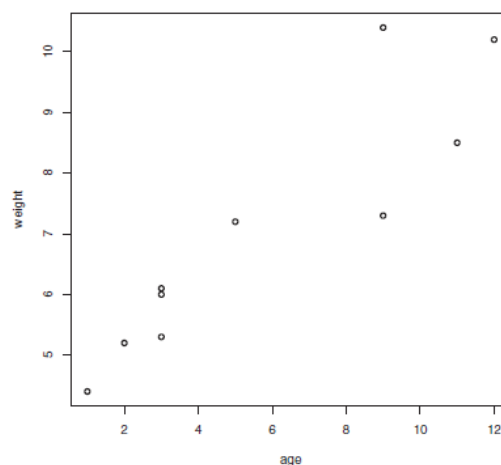
Age (mo.)	Weight (kg.)	Age (mo.)	Weight (kg.)
01	4.4	09	7.3
03	5.3	03	6.0
05	7.2	09	10.4
02	5.2	12	10.2
11	8.5	03	6.1

Note: These are fictional data.

Create a new script, and do the following:

- Create a new script (ctrl+shift+n), name it “phy_development.R”, and save it (ctrl+s) into the current directory.
- Add a commented line to describe the objective of your script, e.g., practising on variables in R.
- Create two vectors for the age and weight of the infants, using the function `c()`.
- Compute the mean and standard deviation of the weights, along with the correlation between age and weight, are provided by the functions `mean()`, `sd()`, and `cor(v1,v2)`, respectively.
- Plot the data, age is plotted against weight using the `plot(x,y)` function, allowing you to visually inspect the trend.
- Finally, save the plot using the `dev.print(pdf, 'filename.pdf')` function.
- Run your script and observe the results.
- What can you tell about the relationship of age and weight in our sample population?

Expected output:



Solution to Exercise 5:

```
> age <- c(1,3,5,2,11,9,3,9,12,3)
> weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1)
> mean(weight)
[1] 7.06
> sd(weight)
[1] 2.077498
> cor(age,weight)
[1] 0.9075655
> plot(age,weight)
> dev.print(pdf, "my_first_plot.pdf")
```