Introduction to R

R Basics

Arun Frey

Welcome!

Welcome to your first `R` session! Today we'll cover some of the basics of how `R` works. This includes understanding how to assign values to objects, generate vectors, and load and inspect dataframes.

This worksheet provides you with exercises to complete during the lab sessions, and some important background information. If you don't get through everything in class, make sure you finish the rest of the material at home.

Introduction

What is `R`?

`R` is a free statistical programming language. You can download the latest version of `R` by visiting: www.r-project.org. For this course, we will be running `R` in the integrated development environment (IDE) "RStudio". To download and setup the most recent version of RStudio, go to: www.rstudio.com/products/rstudio/.

Why use `R`?

`R` is fantastic because, unlike some of the other statistical software you may have used, it is **completely free** and **open-source**! This means you won't have to pay for the program or for any subsequent updates. The `R` community is very inclusive, and you will likely find the right answer to any question you may have with a bit of googling.

Introducing RStudio

RStudio is a nice interface that lets you write code, run scripts, and view the results all in one.

- **Source**: This is the code editor, where you write and save your code. To open a new R script, go to `File --> NewFile --> RScript`
- **Console**: This is where the output of your code will be printed.
- **Environment/History**: This is where any objects, such as vectors, matrices, or dataframes, will be stored.
- **Viewer**: This viewer previews any plots you create. You can also check your folder files and call for help here.

Note: You should always write code into the code editor instead of writing it directly into the console. This allows you to save and revisit the code at a later stage. By contrast, if you run code directly from the console, you will have to write it out all again after closing and re-opening RStudio, which isn't a lot of fun in the long run.

To execute the line of source code where the cursor currently resides you press the **Ctrl + Enter/Cmd + Enter key**, or use the Run toolbar button (located on the top right corner of the editor pane).

Basic commands

Command	What it does
`class(x)`	See the class of an object
`cor.test(x,y)`	Conduct a correlation test between variables between x and y
`head(data)`	Shows first six rows of the data
`hist(x)`	Create a histogram of a variable
`length(x)`	Compute the length of a vector
`ls()`	Lists all the objects that are stored in your environment
`mean(x)`	Compute the mean of a variable
`median(x)`	Compute the median of a variable
`quantile(x)`	Compute the quantiles of the variable
`read.csv(data)`	Load data into R (remember to assign this to an object, by using `<-`!)
`plot(y, x)`	Create a scatterplot that show the relationship between variables x and y
`sum(x)`	Calculate the sum of the vector's values
`summary(data)`	Compute summary statistics of the data/variable (mean, standard deviation, etc.)
`sd(x)`	Compute the standard deviation of a variable
`tail(data)`	Shows last six rows of the data

- To look up how to use commands/options, use the help function in R: `help(command)` or `?` command.
- Use `$` to call a variable. For example, if I have a dataset called data that contains a variable called `economic_growth`, use `dataset$economic_growth`. (Hint: you can use auto-complete by using the tab button on your keyboard once you have entered the first letter of the variable name after `$`.)
- Use `<-` to assign a value or name to an object. For example, to assign the value 5 to an object called a, use `a <- 5`.
- When you want to calculate the mean/median/standard deviation of a variable that has missing values, use the `na.rm=TRUE` option. E.g. `mean(dataset$economic_growth,na.rm=TRUE)`.
- To combine multiple plots, use the `par(mfrow=c(number_of_rows,number_of_columns))`.

Exercise 1

1. Complete lessons 1, 3, 4, and 5 in `swirl()`. To do that, follow the steps below:

```
install.packages("swirl") #you only need to do this one time
library(swirl) # loads the package

# if you don't see the R Programming course
# run the following command
install_from_swirl("R Programming", mirror = "bitbucket")

swirl() # starts swirl and allows you to select from installed courses.
```

2. Create a vector called `x` that contains the numbers 1 to 50.
3. Create a logical vector `y` that is `TRUE` if `x` is smaller than 25.
4. Create the vector `c(0, 5, 10, 15)` in two ways: using `c()` and using `seq()`.
5. Create a vector of length 4 that reads `c("Hello", "my", "name", is")`, and save the vector under the name `my_name`.
    - Using `my_name` and the function `c()`, create a new vector `my_name2` that ends with your name.

Exercise 2

1. Generate a vector that includes all even numbers between 2 and 100. Store the number in `even`.
2. Generate summary statistics for the `even` vector.
3. Check the length of `my_name2`. Modify the vector and save it under `my_name3` in a way that reads `"My name is [your name]"`
4. What happens when you run, and how can you fix it?

```
x <- c(1, 2, "a", 4, 5)
mean(x)
```

5. Convert x into a numeric vector, replacing `"a"` with `NA`
6. Complete lesson 5 in `swirl()`.

Exercise 3

1. In the data folder of this course's Github repository, I have uploaded a dataset of World Bank country-year population estimates. I have uploaded the same dataset in `pop.csv`, `pop.Rda`, and `pop.dta` format. Load each of the datasets.
2. Load the `pop2.csv` dataset into `R` and inspect the data. Does it look any different to the previous datasets? If so, how can you fix this? *Hint*: If you are stuck, look at the documentation of the `read.csv()` function.
3. Save each of the datasets back to your workspace, under the names `pop_new.csv`, `pop_new.dta`, and `pop_new.Rda`. I haven't told you the function for saving data, but it's quite intuitive (and google is always your friend!)
4. Complete Exercise 7 and 12 in `swirl()`.
5. Generate a dataset out of the following vectors:

```
country <- c("Rwanda", "Mexico", "France", "Vietnam", "United Kingdom")
continent <- c("Africa", "Americas", "Europe", "Asia", "Europe")
life_exp <- c(46.2, 76.2, 80.7, 74.2, 79.4)
gdp_pc <- c(863, 11978, 30470, 2442, 33203)
```

6. The above data is taken from the `gapminder` dataset, which you can load by typing `data(gapminder)`. Which country-year observation has the lowest and highest life expectancy, and the lowest and highest GDP per capita?