R Starter Kit: R Basics

Welcome to R Open-Labs!

PART I.

What is R?

R is a open-source programming language and environment for statistical computing and graphics. You can use R as it is, but most people prefer to use R in combination with the Rstudio (GUI-based IDE for R), which has an organized layout, and several extra options (RMarkdown, Shiny, etc).

Some helpful resources:

1. **https://goo.gl/RWV89v (A (very) short introduction to R)**

An interesting 10 page reading that guides you through setting up R and covers all the basics. This is **highly** recommended for first timers.

2. **https://goo.gl/FTT9yZ (R Language Fundamentals)**

For the peeps who love learning through lecture slides, these are just the right resource for understanding the different data types and their manipulation.

PART II.

R Studio Interface

Console:

Where commands are actually executed.

```
> 5 + 6

> x <- c(5, 29, 13, 87)
```

Script

Where commands are edited and saved. Click Run or press CTRL+ENTER(Windows)/command+return(MacOSX) to send it to the console. There are different types of files you can generate: R script, R Markdown, etc.

Environment/history window

This window is your workspace. Environment shows data and values R has defined in its memory. The history window shows what command has been typed before.

Files / plots / packages / helps

Where you can open files, view plots (also previous plots), install and load packages or use the help function.

PART III.

Some Data types with Examples

Basic Calculations

```
> 6 / 2
> 10 * 5 * 3
> 2 + 3 * 6
> (2 + 3) * 6
> 3 ^ 2
> sqrt(4)
> log(10)
```

Defining

The arrow points to the name; on the other side, it is the value you assign to that name. Can also use `=`.

```
> a <- 10
> a / 2
```

Vectors

1-D data; all elements of the same type.

```
> x <- c(5, 29, 13, 87)
> x <- 1:50
> u <- rep(1, 18)
> # Subsetting Vectors ('#' creates a comment)
> y[3:5]
> summary(u)
```

Matrices

matrix: 2-D data, all values of the same type.

```
> mat <- matrix(1:9, nrow = 3, ncol = 3)
> mat2 <- matrix(nrow = 2, ncol = 2)
> mat2[1, 1] = 5
> # Getting the internal structure of an object
> str(mat2)
```

Character

```
> txt <- "Hello World"
```

Logical Value

```
> y <- TRUE
```

List

list: one-D data, elements of different categories.

```
> p <- list(10, "Hello", c(1, 2, 3), FALSE)
```

PART IV.

Loops and conditionals

A program's control flow is the order in which the program's code executes. The control flow of an R program is regulated by conditional statements, loops, and function calls. Examples of conditional and loop statements are given below. Function calls will be dealt with in later classes.

1. Conditional statements - Often, you need to execute some statements only if some condition holds, or choose statements to execute depending on several mutually exclusive conditions. The Python compound statement if, which uses if, elif, and else clauses, lets you conditionally execute blocks of statements.

   Example:

   ```
   > x = 6
   > if(x > 0) print("Non-negative number") else
   print("Negative number")
   ```

2. Loops - Loops supports repeated execution of a statement or block of statements that is controlled by an iterable expression.

   for loop:

   ```
   for (i in vec) {
        action
   }

   1st iteration: i = vec[1]
   2nd iteration: i = vec[2]
   so on...
   ```

   Example:

   ```
   > x <- c(1, 2, 3)
   > for (i in x) {
        print(i^2)
   }
   ```

   while loop:

   ```
   while (condition) {
        action
   }
   ```

   Example:

   ```
   > t <- 10
   > while (t > 5) {
        t <- t-1
   }
   ```