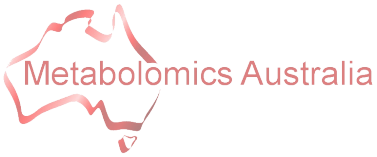
R Tutorial



The Basics

The R CLI

The command line interface (CLI) is the basis of most functionality in R. This can be used to run scripts, or simple commands. For example, one can type

```
> 4+5
```

which will print on the screen

```
[1] 9
```

You can also type in

```
> if (length(grep("^X[\\d]",colnames(pca_data),perl=TRUE)) != 0)
{gsub("^X([\\d].*)","\\1",colnames(pca_data),perl=TRUE)} else {colnames(pca_data)}
```

which will have some effect in context, however looks somewhat disturbing at first glance, but this will be covered later.

Using scripts

There are two main methods of running scripts in R. The first is to open the file containing the script in a text editor, and copy and paste the lines into the CLI, either line-by-line or as a block. The second is to use the command '`source`' (provided the script is present in the working directory – see "Setting the working directory" below)

```
> source("name_of_script.r")
```

which will run the script. The script may contain a function, which is a block of code that has a particular purpose, which is then executed by using the command that is specified by the function (see below).

Functions

These are a blocks of code that execute many lines of code at once. It is possible to have a script that defines a number of functions that are then executed within the script. This can prevent re-typing the whole code block by using a short command instead. For example, if you needed to make a number of plots of the same type, you might have a function that generated the plot which takes an argument (that which is in the brackets) of which data matrix to use, e.g.

```
> plot_image(density)
> plot_image(mass)
```

rather than

```
> plot(density,main="Density vs Size",ylab="Size (km)",xlab="Density (kg/m³)",cex=3)
> plot(mass,main="Mass vs Gravity",ylab="Gravity (g)",xlab="Mass (kg)",cex=3)
```

etc.

R Tutorial

R basics

Variable assignment

Assigning a result to an object is one of the basic steps in using R. To do this, you indicate that you wish to put a result into a variable by using the syntax `variable <- <R command>`, for example

```
data <- read.csv("input.csv", header=TRUE, row.names=1)
```

which reads the file "`input.csv`" and stores it as the variable `data`.

Setting the working directory

To see where you are, you can type

```
getwd()
```

which will show you the path to the directory that you are working in (where you will be able to access files). To change the directory, you can use the command (for a GNU/Linux machine)

```
setwd("/home/user/folder/with/data/or/scripts/")
```

or for a Windows machine:

```
setwd("C:\\Documents and Settings\\user\\My Documents\\folder\\with\\data\\or\\scripts")
```

(e.g. copy from Windows Explorer and double the backslashes), or

```
setwd("C:/Documents and Settings/user/My Documents/folder/with/data/or/scripts")
```

(e.g. replace the backslashes with forward slashes), or select `File > Change dir..` from the menu. All of these variants (on a Windows machine) perform the same function, so choose whichever method works best for you. After using `Change dir..` from the file menu, it is a good practice to use `getwd()` to verify your working directory is what you expect it to be.

Getting help

For all in-built functions in R, you can use `help("command")` or `?command` to show a help file. Under GNU/Linux, this usually comes up in the current terminal window, while in Windows it appears in a browser window.