Day 3, Session 1: R basics

Jessica Williams-Nguyen and Brian D. Williamson

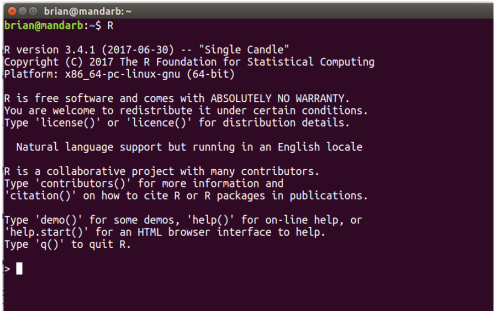
EPI/BIOST Bootcamp 2017

26 September 2017

Command line R

The most basic form of R is from the command line (Terminal in Mac/Linux, Command Prompt in Windows).

After installing R, typing R in the command line will open an R session:



```
brian@mandarb:~$ R

R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

The text gives you the version number (here, 3.4.1) and some other (not necessary for our purposes) information.

Command line R

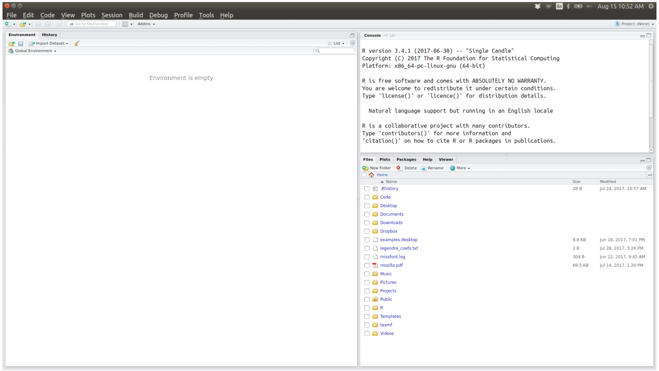
You also get a prompt, the >, where you can enter commands.

When you first open R, you have a limited number of functions available for use. These include things like `mean()`, `median()`, and `read.table()`.

The command line version of R is most useful (in my experience) for computing-intensive tasks. Data analyses are best done in RStudio.

RStudio

When you first open RStudio, you are met with a blank set of four panes:



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

Name	Size	Modified
.Rhistory	28 B	Jul 24, 2017, 10:57 AM
Code		
Desktop		
Documents		
Downloads		
Dropbox		
examples.desktop	8.8 KB	Jun 18, 2017, 7:01 PM
legendre_coefs.txt	2 B	Jul 28, 2017, 3:26 PM
missfont.log	304 B	Jun 22, 2017, 9:45 AM
mozilla.pdf	69.5 KB	Jul 14, 2017, 1:20 PM
Music		
Pictures		
Projects		
Public		
R		
Templates		
texmf		
Videos		

RStudio

You can edit pane layout under `Tools > Global Options > Pane Layout`. My preference is to have the source in the upper left, console in the upper right, environment in the lower left, and viewer in the lower right.

You'll notice that on the previous slide, there were only three panes visible: since we haven't opened or created any files to edit, the source pane is not currently open.

RStudio: the console

Is it a coincidence that the screenshot has exactly the same text as we saw on the command line? No!

The console performs identically to the command line. Any commands that we wish to enter go through the console, and any results that are output by these commands will appear in the console.

Example: the console

Typing 47 at the > symbol (from now on, called the execution line) and hitting Enter on the keyboard yields the following:

```
> 47
[1] 47
```

Since R is a functional programming language, typing 47 and hitting Enter is the same thing as using the print() function:

```
> print(47)
[1] 47
```

The output is displayed as a vector, one of the fundamental data structures in R. The [1] helps to tell which element of the vector we are looking at (which is most useful if the vector spans multiple lines in the console).

Example: heights and weights

The console is the workhorse of RStudio. If we wanted, we could work exclusively here, but then reproducibility is difficult...

Let's say we have data on the height and weight of five (imaginary) individuals. A simple way of reading the data into R is by creating two objects, `ht` and `wt`, and assigning the numerical values that we have collected for each individual; in the console, type

1. `ht <- c(72,65,84,73,68)`, followed by hitting the `Enter` key, (creates the `ht` object)
2. `wt <- c(165,120,210,180,125)`, followed by hitting the `Enter` key (creates the `wt` object)

Each of these commands assigns a value to an object, using the special <- command. Values go on the right-hand side, and objects go on the left.

Example: heights and weights

Both `ht` and `wt` are also vectors! We create vectors using the `c()` function, which concatenates scalars (single numbers, or single strings like `"Hello"`) into vectors.

In words, `ht <- c(72,65,84,73,68)` means: concatenate the numbers 72, 65, 84, 73, and 68 into a vector, and assign that vector to a variable (or object) called `ht`.

Typing these commands into the console allows us to manipulate both `ht` and `wt`; in particular, they can be used as arguments into many R functions.

Example: heights and weights

However, these objects are part of the current R environment: this means that we can only use them while the current session of R is open. If we exit RStudio, we will have to re-enter the two lines defining both `ht` and `wt`.

For more complicated analyses, this can lead to a lot of confusion. Also, just typing in the console can lead us to forget what steps we took to get a certain figure or table.

Organizing commands into scripts allows us to save exactly what we did during a given data analysis, which makes reproducible research easy. This helps both your future self and your collaborators, so that you can share what you did!

RStudio: the text editor ("Source")

The text editor (which RStudio calls the source) is the second most powerful part of the IDE. This allows us to edit and save files, and to run commands directly from the text file into the console.

To create a new R script, either type `Ctrl+Shift+N` or go to `File > New > R script`. There are many other options here (most notably R markdown, which we won't cover but is incredibly useful). This brings up the source pane.

At the top of the source pane are a set of buttons (with associated keyboard shortcuts) that will save the current file, run sections of code, or run the entire document.

R scripts

R scripts are collections of R commands and comments. These are run in the console to manipulate objects and produce output.

To create a comment, type a # symbol. Anything on a line following a # will not be run by R.

I like to include a preamble in all of my R scripts, which gives me information on when I created the file, what it does, whether or not it takes any input or produces output, and what I have changed since I started. Here is a sample:

```
###########################################################################################
##
## FILE: <insert file name here>.R
##
## CREATED: <insert date here> by <your name here>
##
## PURPOSE: <Give a brief description of what the code in this file is supposed to do>
##
## INPUTS: <What inputs? Where does the data come from?>
##
## OUTPUTS: <What outputs? Plot/table names and locations>
##
## UPDATES:
## DDMMYY INIT COMMENTS
## ------ ---- --------
## <date> <inits> <What did you change?>
###########################################################################################
```

Example: heights and weights

We can now enter

```
ht <- c(72,65,84,73,68)
wt <- c(165,120,210,180,125)
```

into a new R script. Saving this as `ht_wt_analysis.R` is a start to analyzing these data! The script now looks like:

```
###############################################################################################
##
## FILE: ht_wt_analysis.R
##
## CREATED: 15 August 2017 by Brian Williamson
##
## PURPOSE: Teach some R basics using height and weight data on 5 imaginary individuals
##
## INPUTS: None
##
## OUTPUTS: None
##
## UPDATES:
## DDMMYY INIT COMMENTS
## ------ ---- --------
###############################################################################################

ht <- c(72,65,84,73,68)
wt <- c(165,120,210,180,125)
```

RStudio: environment, file history, etc.

RStudio: files, plot viewer, packages, help

Functions

Objects

Loading data

Saving data

Manipulating data

R packages