Introduction to R, Day 1: “Getting oriented”

Michael Chimenti and Diana Kolbe
Dec 5 & 6 2019
Iowa Institute of Human Genetics

Slides adapted from HPC Bio at Univ. of Illinois:
https://wiki.illinois.edu/wiki/pages/viewpage.action?pageId=705021292

Distributed under a CC Attribution ShareAlike v4.0 license (adapted work):
https://creativecommons.org/licenses/by-sa/4.0/
Learning objectives:

1. Be able to describe **what R is** and how the programming environment works.

2. Be able to **use RStudio to install add-on packages** on your own computer.

3. Be able to **read, understand** and write simple R code

4. Know how to get **help**.

5. Be able to describe the **differences** between R, RStudio and Bioconductor
What is R? (www.r-project.org)

• "... a system for statistical computation and graphics" consisting of:
  1. A simple and effective programming language
  2. A run-time environment with graphics

• Many statistical procedures available for R; currently 15,111 additional add-on packages in CRAN

• Completely free, open source, and available for Windows, Unix/Linux, and Mac
What is Bioconductor? (www.bioconductor.org)

- “… open source, open development software project to provide tools for the analysis and comprehension of **high-throughput genomic data**”

- Primarily based on R language (functions can be in other languages), and run in R environment

- Current release consists of 1741 **software packages** (sets of functions) for specific tasks

- Also maintains 948 **annotation packages** for many commercial arrays and model organisms plus 371 **experiment data packages** and 27 **workflow packages**
Pros of “R” and “Bioconductor”

- FREE
- Open source, not “black box”
- Continual improvements available, cutting-edge statistical methods
- Excellent graphic capabilities
- Available for Windows, Mac OS X and Linux
Cons of “R” and “Bioconductor”

- Difficult to learn
  - Mostly command-line interface
  - Not as intuitive as point-and-click
  - No professional-quality guides and tutorials
- Too many choices sometimes…
- Support not from paid sources, but from community of users
R uses a 3-place versioning system:

3.6.1
Major.minor.revision/patch

Major: only incremented upon substantial code changes or added functionality that could cause incompatibility with other software designed for previous versions.

Minor: smaller functionality additions and/or substantial bug fixes

Revision/patch: only minor bug fixes

Bioconductor uses the Major.minor 2-place system;
all add-on packages use the 3-place system.
Installing packages in R

Main sources / repositories:

1. Comprehensive R Archive Network (CRAN)
2. Bioconductor

Easiest to do from within R; main methods:

install.packages()

- Gets proper version and any dependency packages
- Default is to only look in CRAN

BiocManager::install() (for R >= 3.5.0)

1. Wrapper around install.packages() to also look in BioC and check for updates to current packages
2. Must install BiocManager package from CRAN first
3. install_github() in devtools package
Quiz: If you need Package ‘helloWorld’ version > 2.43.16, which of below would be OK?

A) 2.8.19
B) 2.45.0
C) 2.43.5
D) More than 1 of the above
E) All of the above
Why do we worry about versions?

- New developments/methods are constantly being added
- Discovered bugs get fixed; no support for problems with older versions
- If you don’t have R 3.6.1, you’re out of date already!
- Note: if you upgrade to new version of R, you will have to reinstall your packages from scratch, so make a list first
R development process

• An annual x.y.0 release every Spring* (usually April)

• "Patch" (revision) releases as-needed the rest of the year.

• Each major/minor/revision release is actually a stand-alone installation separate from the others (but not by default on Macs), with interesting names ("Kite-Eating Tree", "Short Summer ", "Another Canoe", "Sincere Pumpkin Patch", "Bug in Your Hair", "Fire Safety", etc.)

• Most recent release version: 3.6.1 (out 2019-07-05)

• Upcoming patch release: 3.6.2 (schedule 12.02.19)

• Upcoming annual release: 3.7.0 (2020-04-??)
Bioconductor development schedule

- 2 scheduled releases per year: April (a few days after R’s) and October, usually minor.

- Individual packages in BioC have their own versioning and patch/revisions are allowed in between.

- Compatibility between BioC packages and R major.minor versions is crucial; `BiocManager::install()` function available to automatically get appropriate package versions!

- Most recent release version: 3.10 (2019-10-30)
Introduction to R Studio

The only way you’ll want to use R if possible
• “Integrated development environment for R”

• Integrates R console with an excellent editor, workspace viewer and graphics manager

• Make using R *easier*
"Map" of RStudio

EDITOR

CONSOLE

ENVIRONMENT

HISTORY

PLOTS

FILE BROWSER

HELP FILES
“Reproducible Research”

- Very important to document how you manipulate / select data, particularly for large data sets

- **R code** is an easy way to track what you have done and instantly reproduce it, even months or years later! (unlike Excel)

- Many tools in the R/Bioconductor community for easy integration of codes and html output that document both the codes and the results (ReportingTools, RStudio/Shiny, RStudio/Rmarkdown/knitr/git)

https://www.rstudio.com/resources/cheatsheets/
RStudio Text Editor

- Works just like notepad or any text editor you’re used to
- Creates an exact, reproducible copy of the commands used
- Text can be saved, edited and copied
- Comments can be added for later reference
R Console is where you execute live commands

- R has a command-line driven interface; entered commands or expressions are evaluated and the proper output is returned

  > 2+2
  4
  > 3*3
  9
  > log(100)
  2
  > mean(x)
  ...

Key concepts in R coding

1. **Objects**
   1. Hold information in a structured way
   2. “class” of the object depends on type and arrangement of information

2. **Functions**
   1. Pre-written code to perform specific commands
   2. Typically used on objects
Common object “classes”

vector – a series of data, all of the same type

matrix – multiple columns of same length, all must have the same type of data (usually numeric)

data.frame – multiple columns of same length, can be mix of data types, headers allowed

list – a collection of other objects; each item in the list can be a separate type of object

function – a command in R
Naming Objects

- In R, use “<-” to create objects; what’s on the left side is the object name and can be *almost* anything.
  
  ```r
  x <- 4
  ```

- Object names can consist of letters, numbers, periods* and underscores.
  - Cannot start with a number; best to start with letter.
  - e.g., x, mydata, mydata_normalized, TrtRep2

- Check to make sure desired object name is not already a function
  
  ```r
  ?objectname
  ```

*best practice is to not use ‘.’ because it means something very different in Python
Object attributes

- Standard attributes include **dim**, **class** and **names**
- A matrix is actually a type of vector with ‘dim’ attribute set
- A data.frame is actually a type of list with every item having the same length
- Generic functions like plot() can have methods defined for a particular object class
How to use functions in R

- Functions are indicated by parentheses – ()
  
  \( \text{sqrt}(81) \)

- "Arguments" are the input to functions within () and are separated by commas
  
  \( \text{ls()} \) 0 arguments
  
  \( \text{rm(myobject)} \) 1 argument
  
  \( \text{cbind(x1, x1 + 1)} \) 2 arguments

- Most functions have > 1 argument; input can either be listed in order, or associated by name.
  
  \( \text{write.table(object, "outputname.txt", FALSE)} \)
  
  \( \text{write.table(object, append = FALSE, file = "outputname.txt"}) \)
Getting help to understand a function

- type in ?rownames
- Anatomy of a help page:
  - very top: main.function (package)
  - Title
  - Sections:
    - Description
    - Usage: names arguments in order with (usually) default values
    - Arguments: description and possible input
    - Details: further information
    - Value: the output of the function
    - ... possibly other sections
    - Note: any other useful information
    - References: see for more information, what to cite
    - See Also: related functions
    - Examples: how can be used
How to use functions in R, cont.

- R add-on packages - sets of functions that do particular things.

- ONCE only per R version: packages need to be **installed**

- EVERY time you open R: packages need to be **loaded again!**
  
  ```
  library(edgeR)
  
  "Error in library(edgeR) : there is no package called 'edgeR' – package has not been installed yet"
  ```

- "Error: could not find function "xxxx" " – package has probably not been loaded from library.
Functions for exploring objects

str() – overall structure of the object

class() – gives the "class" of the object

length() – gives the number of entries in 1D objects (vector, list)

dim(), nrow(), ncol() – gives number of rows/columns for 2D objects (matrix, data.frame)

names() – gives/sets names of list items or data.frame columns

rownames(), colnames() – gives/sets row & column names of a matrix or data.frame
How R syntax works

• R has **strict formats** for entering commands and referring to objects; commands that are not properly formatted will not be evaluated.

• () {} and “” must come in pairs; if not done correctly, R will indicate command is not finished, or will return error

• R is **case-sensitive**, except for file names on Windows/Mac
  
  Plot != plot but “myfile.txt” == “MyFile.txt”

• Spaces generally do not matter, except within quotes
  
  temp<-c(1,2) == temp <- c ( 1 , 2 )

• To use \ must use \\\n
Types of R variables

**Numeric** – 1, 2, 426, 4500, etc.

**Character** – “a”, “B”, “data”, “cell line”, etc.

**Factor** – reads as character, but treated as categorical

**Logical** – TRUE or FALSE

  T  F
  1  0

**Missing** - NA to indicate missing values
**Base R vs. RStudio / tidyverse (more tomorrow)**

- **Hadley Wickham** wrote a series of packages (the **tidyverse**) that updates the way R handles large data to be more data-science friendly.

- Way of working/thinking is vastly different:
  - Bioconductor pre-dates the tidyverse and many packages don’t work well in the tidyverse (although this is changing: **biobroom**, **Organism.dplyr**, **plyranges**)

- Tidyverse can always be used to clean up raw data, even in the downstream packages don’t “play nice” with it.

```r
mtcars$pounds <- mtcars$wt * 1000
mtcars <- mtcars %>% mutate(pounds = wt / 1000)
```
Subsetting objects in base R

[ and $ are the main base R ways to subset:

• use [ ] to subset 1D objects (vector, list)
• use [ , ] to subset 2D objects (matrix, data.frame)
  – rows first, then columns
• inside [ ] or [ , ] can be positions, names in quotes or TRUE/FALSE values.
  – Can also be used to re-order objects
• $ can pull out a named column from a data.frame or a named item in a list
• a $ must be followed by the name
Subsetting lists

Given list \( x \):

- \( x[4:6] \) is items 4-6
- \( x[5] \) is item 5
- \( x[[5]] \) is the object in slot 5 (could be another list, dataframe etc…)
- \( X[-1] \) is \( x \) without the first entry
Workspace vs. working directory

The **Workspace** is the internal R memory where it stores the objects you create during a session. The objects can be saved to/loaded from an external file for a more permanent copy.

The **Working Directory** is an external directory (folder) where R will look to import files or export files when told to given a relative file name (e.g., “myfile.txt”). Many functions automatically read from/export to the working directory without you having to specify it.
Saving work

- Matrix/data.frame objects can be written out to individual files using `write.table()` and `write.csv()`
- All of the R objects can be saved to an .RData file using `save.image()` or `saveRDS()`
  - These can only be read by R to re-load objects using `load()` or `readRDS()`
  - If no filename given to `save.image()`, will be saved as unnamed .RData file in current working directory - DO NOT USE
  - **Note:** only objects saved, not how they came to be
- To save all commands entered, can use `savehistory()`, or...

- **Commands more important than objects, so strongly recommend using RStudio or other text editor to save final, correct version of commands!**
Saving work and objects, cont.

- Objects already in the workspace can be overwritten **without warning**!

- Files in the working directory can also be overwritten **without warning**!

- There is no "undo" in R other than to rerun the code!
  - in RStudio, the "undo" only works on the code editor, not the objects in the R workspace
Reading in data from tables and sheets

If have Excel-type spreadsheet:

1. Use short column names, no spaces or special characters; do not start with a number
2. No merged cells, fancy formatting or thousand separators in numbers
3. Save as tab-delimited text file (.txt) or comma separated values file (.csv) for ease of importing:
   - `read.delim()`
   - `read.csv()`
R tips and tricks

1. You don't have to understand everything that code does in order to modify it - just be able to recognize the part that does need modification.

2. Search for a package or code snippet that does what you want. Do not “re-invent the wheel” when you are starting out unless you are doing a learning exercise.

   – Example: DO NOT write your own data frame structure, your own statistics package, your own plotting functions.

3. R and Bioconductor packages written by others will often have function(s) that contain the main computations that you want to do.

4. When a line of code contains multiple computations/functions, run each computation separately to get a better understanding of what it does or why it may be throwing an error.
How to get help

• Help: ?function or help(function)
  for example: ?read.table or ??read.table

• Html help:
  1) type help.start()
  2) Menu: help -> html help

• To see the code of many functions, simply type the function name. For example: apply


• Longer reference card: http://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf

• Understanding R code cheat sheet:
  http://go.Illinois.edu/introR
How to get help, cont.

R help mailing list

- https://stat.ethz.ch/mailman/listinfo/r-help

Bioconductor support site

- https://support.bioconductor.org/
- Be sure to read the posting guide before posting!

Google!

Stack overflow
Quitting R

• Before you quit, first save your script in the editor and give it a descriptive name

• Default prompt asking whether you want to save the workspace image
  – If pick "Yes", will save objects in workspace as unnamed .RData file and commands as unnamed .Rhistory in current working directory; DO NOT GET IN THE HABIT OF USING THIS!
  – If pick "No", will lose objects and codes unless you have saved them elsewhere; despite risk, this is best for reproducible research unless the R data objects took a VERY long time to compute, then see above.
  – If pick "Cancel", return to R
**Additional Resources**

https://www.nature.com/news/programming-tools-adventures-with-r-1.16609

https://ropensci.org/packages/


https://www.nature.com/articles/nmeth.3252
Base R ‘SWIRL’ lessons

type:

> install.packages('swirl')

> library('swirl')

> swirl()

Work lessons 1 through 6 at your own pace...

We are available to answer questions!