WELCOME! Please sign in and pick up:

1. iClicker
2. Name tag for back of computer
3. Red and green sticky notes
4. Handouts

BOOKMARK the link to workshop wiki:
https://go.Illinois.edu/introR
Learning Objectives:

1. Be able to describe **what R is** and how the programming environment works.
2. Be able to **install R and 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 do you know about R?

A) I’ve never heard of R before signing up for this workshop.

B) I’ve heard of R before, but don’t know what it is.

C) I know what R is, but I’ve never used it before.

D) I’ve used R a little bit before this workshop.

E) I’ve used R extensively *(wait, why am I even here?)*
What software do you use for statistics?

A. SAS / SPSS / JMP
B. Matlab
C. Excel
D. R / Python
E. I don't do statistics
Did you download R and attempt the swirl pre-homework?

A. No
B. I just downloaded R
C. I started swirl but couldn't figure out how to use it
D. I got through a lesson or two
E. I got through many lessons
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 12,223 additional **add-on packages** in [CRAN](http://cran.r-project.org)
- Completely free, open source, and available for Windows, Unix/Linux, and Mac
Programming tools: Adventures with R

A guide to the popular, free statistics and visualization software that gives scientists control of their own data analysis.

Sylvia Tippmann

29 December 2014 | Clarified: 13 February 2015

http://www.nature.com/news/programming-tools-adventures-with-r-1.16609
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 1477 software packages (sets of functions) for specific tasks

• Also maintains 909 annotation packages for many commercial arrays and model organisms plus 326 experiment data packages
Recent Nature Methods article

Orchestrating high-throughput genomic analysis with Bioconductor


Affiliations | Corresponding author

Received 30 July 2014 | Accepted 09 December 2014 | Published online 29 January 2015

http://www.nature.com/nmeth/journal/v12/n2/abs/nmeth.3252.html
Pros of R & 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 & Bioconductor

• STEEP learning curve
  - 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
A quick note on software versioning...

R uses a 3-place versioning system:

3.4.3

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.
If you need package X version \( \geq 2.43.16 \), which versions 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
Wait, why do I need to worry about versioning? My lab computer already has R on it!

- New developments/methods are constantly being added
- Discovered bugs get fixed; no support for problems with older versions

If you don’t have R 3.4.3 right now, you are out of date!!!!!
R’s continuing development

• 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: 3.4.3 (out Nov 30, 2017)

• Upcoming patch release: 3.4.4 (out Mar 15, 2018)

• Upcoming annual release: 3.5.0 (likely end April 2018)

*used to be biannual release every Spring and Fall, which is why...
Bioconductor’s continuing development

- 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; `biocLite()` function available to automatically get appropriate package versions!
- Most recent release version: 3.6 (Oct 31, 2017)
- Upcoming biannual release: 3.7 (May 1, 2018)
WARNING!!!

- From April to Oct 2016, they happened to have the same major.minor: R 3.3.x and BioC 3.3
- In Oct 2016, R stayed 3.3.1 but BioC incremented to 3.4
- In Apr 2017, both incremented: R to 3.4.0 and BioC to 3.5
- In Apr 2018, both will increment: R to 3.5.0 and BioC to 3.7
- Use biocLite() to get correct package versions!
Installing R

- R: www.r-project.org
- See my R installation guide handout:
- I will demonstrate how to install R on Windows and Macs…

Warning – do not try to install on lab computers!!
Installing add-on packages

- Main sources / repositories:
  1. Comprehensive R Archive Network (CRAN)
  2. Bioconductor
  3. Omegahat
  4. github / SVN / other internet

- Easiest to do from within R; main methods:
  1. `install.packages()`
     - Gets proper version and any dependency packages
     - Default is to only look in CRAN
  2. `biocLite()`
     1. Wrapper around install.packages() to also look in BioC and check for updates to current packages
     2. Must download first; I still recommend for either CRAN or BioC packages
     3. `install_github()` in devtools package
How to install both ways

• `biocLite()`
  - example in R installation guide
  - also will demo in intro BioC workshop

• `install.packages()`
  - example in R swirl guide
  - will demo this now...
Windows Security Control

- On Windows >= 7 machines, there are controls so that even if the user logged in has Administrator privileges, individual pieces of software are not automatically granted admin status.

- During the R installation, the base packages are typically installed at C:/Program Files/R/R-3.4.3/library.

- However, when R is opened it does not have admin privileges, so it cannot install into the main library location.

- R will now detect this and create a local library for you, by default in your home directory's Documents: C:/Users/igb/Documents/R/win-library/3.4 (note this is versioned only to the 2nd place!)

- This also occurs on systems where you do not have admin status, like these workshop computers and biocluster.
Windows Security Control, cont.

- Using a local library + the main library is not really a problem…
- EXCEPT 1: when packages in the main library need to be updated!
  - If you do not have admin privileges, using `update.packages()` or `biocLite()` should ask you about installing in your local library, which will be used over the version in your main library. (Note – the pop-up window can sometimes be hidden – will need to look for it if R seems to freeze after warning about "not writable")
  - If you do have admin privileges, you can still do the above to make your life simpler, OR you can right-click on the R shortcut and select “Run as administrator” for that session.
- EXCEPT 2: during the Fall BioC release, because the local library is only versioned to the 2nd place you can have more than one R/BioC version pointing to the same library!
How to upgrade Bioconductor in the Fall:

1. Source biocLite as normal
2. Follow the message
How to upgrade Bioconductor in the Fall:

1. source("https://bioconductor.org/biocLite.R")

2. biocLite("Bioconductor")
How to upgrade Bioconductor in the Fall:

1. Source biocLite as normal
2. Follow the message
3. Tell biocLite to upgrade
4. Type y to upgrade
Practice swirl interactive lessons

- swirl: [http://swirlstats.com](http://swirlstats.com)
- See my "R swirl guide" for directions on how to get started with swirl

Double-click on the R x64 3.4.3 on the desktop (not RStudio).

- Work at your own pace until XX:XX. Ideally you will get through several lessons but I will go over all the core concepts again. **TAKE A BREAK DURING THIS TIME!!!**

- **Note** - **SKIP lesson #2 Workspace and Files** (while useful, I used R for 8 years without needing to know most of this!)
“Reproducible Research”

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

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

- Large push 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/
Tips:

• A text editor makes working with R much easier!
  - Notepad++ or RStudio

• Always save the commands you’ve entered, at least one of these three ways:
  - **Work through an editor, and save the file**
  - In R, savehistory(“myhistory.Rhistory”)
  - Copy the contents of the entire console to Word & save

• Always document the output of sessionInfo() to record the versions of R + packages you used.
RStudio
http://www.rstudio.com/products/rstudio-desktop/

• “Integrated development environment for R”
• Integrates R console with an excellent **editor, workspace viewer and graphics manager**
• Make using R *easier*
• Beware of too much reliance on RStudio because it’s not available everywhere R is!!
• We will start using this, but in a limited fashion…
Base R vs. RStudio / tidyverse

- **Hadley Wickham** wrote a series of packages (the **tidyverse**) that changes the fundamentals of R

- Syntax is very different between the two:
  ```r
  mtcars$pounds <- mtc$wt * 1000
  mtcars <- mtcars %>% mutate(pounds = wt / 1000)
  ```

- Google yields lots of **good discussions**

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

- **Genomic data analyses**: learn base R

- Other data analyses: learn the tidyverse
Open RStudio

Double-click on RStudio icon on desktop to open (if you get a pop-up about allowing the program to make changes on your computer, select “Yes”)

See "Continued use of R guide" for how to download for your own computer:

http://www.rstudio.com/products/rstudio/download/

Note: RStudio is overly helpful with auto code completions. To turn off/modify see Tools -> Global Options -> Code -> Completion -> Show code completions
Create folder to save in

- In bottom-left console, type `getwd()` and note the location
- Use File Explorer/Finder to navigate to this directory and create a folder named “introR_sp18”
- Download `introR_06mar18.zip` file from the wiki, copy & paste all the internal files to introR_sp18:

  https://go.Illinois.edu/introR
Let’s try some practice codes together…

First, hover over the divider between the two panes on the right side, then drag the divider most of the way to the top. *We will mostly ignore the top pane and only use the “Help” on the bottom for now.*

At the top tool bar, click on:
File -> Open File -> introR_sp18 -> introR_06mar18R

Copy the line of code starting with “options(…)”) and paste into the Console pane on the bottom right (or hit Ctrl + Enter)
Copy and Paste or Type by Hand?

- CaP – less likely to have typos, but get less experience writing code
- TbH – waste a lot of time on typos, but will learn R faster

General recommendation: TbH in editor either from scratch or modify previous code, then CaP into R console

But for today, let’s retype the codes in the console…
How R Works

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

  
  
  2+2
  3*3
  3+8*2
  log10(1000)
  log2(8)
  abs(-10)
  sqrt(81)
2 key concepts in R

1. **Object**
   1. Holds information
   2. “class” of the object depends on type/s and arrangement of information

2. **Function**
   1. Pre-written code to perform specific commands
   2. Typically used on objects
“types” of information
swirl 1, 3, 5, 8

Most common:

**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

\[
\begin{array}{cc}
T & F \\
1 & 0 \\
\end{array}
\]

**Missing** - NA to indicate missing values
vector – a series of data, all of the same type

matrix – multiple columns of same length, all must have the same type of data

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

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

function – a command in R
Namening Objects

swirl 1

- In R, use "<-" to create objects; what’s on the left side is the object name and can be *almost* anything.

  \[ x <- 4 \]

- 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

  \[ ?\text{objectname} \]

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

Do codes through Vector & Matrix…
How to use functions

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

- “Arguments” are the input to functions within () and are separated by commas
  \[
  \begin{align*}
  \text{ls()} & \quad 0 \text{ arguments} \\
  \text{rm(myobject)} & \quad 1 \text{ argument} \\
  \text{cbind(x1, x1 + 1)} & \quad 2 \text{ arguments}
  \end{align*}
  \]

- Most functions have > 1 argument; input can either be listed in order, or associated by name.
  \[
  \begin{align*}
  \text{write.table(object, “outputname.txt”, FALSE)} \\
  \text{write.table(object, append = FALSE, file = “outputname.txt”)}
  \end{align*}
  \]
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 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

  \[
  \text{Plot} \neq \text{plot} \quad \text{but} \quad \text{"myfile.txt"} = \text{"MyFile.txt"}
  \]

- Spaces generally do not matter, except within quotes

  \[
  \text{temp}<\text{c}(1,2) = \text{temp} <\text{c}(1,2)
  \]

- To use \ must use \\, else use /
Help for functions

- 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 II

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

• ONCE only per R version: packages need to be **installed** (see the [R Installation guide](#)).

• EVERY time you open R: packages need to be **loaded**

```
library(edgeR)
```

• "Error: could not find function "xxxx" " – package has probably not been loaded.

back to demo...
Subsetting objects

[ and $ are the main 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

If list x is a train carrying objects, then:

- x[4:6] is a train of cars 4-6
- x[5] is a train of just car 5
- x[[5]] is the object in car 5

back to demo...

Inspiration from:
http://twitter.com/#!/RLangTip/status/118339256388304896
Object attributes

• Arbitrary additional metadata can be added to an object as an attribute

• Standard attributes include \texttt{dim}, \texttt{class} and \texttt{names}

• Can set class() attribute to make up your own \texttt{S3 class} of object out of a base type; common in CRAN packages

• A matrix is actually a type of vector with dim attribute set

• A \texttt{data.frame} is actually a type of list with every item having the same length

• Generic functions like \texttt{plot()} can have methods defined for a particular object class
Workspace vs. Working Directory
swirl 2

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 your work

• Matrix/data.frame objects can be written out to individual files using write.table()/write.csv()

• All of the R objects can be saved to an .RData file using save.image()
  - These can only be read by R to re-load objects using load()
  - 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!
Important notes

- 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
A Warning about Excel!

- Excel likes to automatically & **irreversibly** change entries that look like dates to a date format.

- This is bad for gene symbols like sept1, apr1, dec1, etc.

- Beware of opening .csv files in Excel and re-saving as any type of file; symbols will be changed.

- Output data from R in .txt format; when opening in Excel, the 3\textsuperscript{rd} page of the import wizard will let you change columns to text format.
Reading in data

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()
Remember these R tips for newbies

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

2. **Copy and paste** working codes, then modify instead of typing all out from scratch

3. Codes written by others will often have a function with a computation or other function nested inside of it.

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
Getting Help and Codes

- **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`

- **Good reference card:** [http://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf](http://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf)

- **Understanding R code cheat sheet:** [http://go.Illinois.edu/introR](http://go.Illinois.edu/introR)
Additional Help II

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

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

- Google!
How to quit R

• Command line: q()
  – or in RStudio/Rgui, top right corner X of window

• 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
  – If pick "Cancel", return to R
Additional Resources

http://go.Illinois.edu/introR

Why learn R: some good articles

How to learn R: Other people’s Intro to R materials

How to learn the tidyverse: a few links

R blogs: Short posts on specific topics – very educational!
Practice questions & homework

• At the end of the intro_06mar18.R script are 12 practice questions.

• How many can you do:
  - Without looking back through the example codes?
  - Without peeking at the answers at the end?

• Before Friday:
  - Download newest version of R to your computer
  - Spend 30-60 minutes doing more swirl lessons