Introduction to R for Data Analysis in the Health Sciences: Lecture 1

Slides on Canvas and github.com/adw96/biost509

Amy Willis, Biostatistics, UW

27 September, 2019

Land acknowledgement

Welcome to BIOST 509

Welcome!

- "Introduction to R for Data Analysis in the Health Sciences"
- 2 credits, CR/NC grading

More on the scope and content of this course in a few slides. . . .

Who am I?

I'm Amy Willis, PhD; I am a tenure-track Assistant Professor in Biostatistics

- Principal Investigator, Statistical Diversity Lab
  - Methods for microbiome data analysis
- PhD in Statistics, Cornell University
- I <3 statistics and data analysis; I've worked for Google, Australian Government, macroeconomic forecasting consulting...
- Most important qualification: 10+ years of `R` experience
  - I'm a methods developer; most of my methods are coded in `R`
  - packages: `breakaway`, `DivNet`, `corncob`, `paramedic`...

Please call me *Amy*, *Professor Willis* or *Dr. Willis*; I use she/her pronouns

Who are they?

Two fabulous TAs: Serge and Thayer

Who are they?

Serge Aleshin-Guendel; I am a third year PhD student in Biostatistics

- Working on record linkage and capture-recapture (methods used in the the analysis of human rights data)
- BS in Math, BA in Computer Science, Boston College
- Have been programming for ~7 years, using R for ~4 years

Please call me *Serge*; I use he/him pronouns

Who are they?

Thayer Fisher

- I am a 3rd-year PhD Student in Biostatistics
- Member of the SLAB Lab, a group which studies the ways in which machine learning methods can help solve biological problems
- I am also a member of the Matsen Group at Fred Hutchinson Cancer Research Center
  - Working on understanding the mechanisms that drive mutations in the immune system
- I am currently developing my first `R` package, `gspam`, and the majority of the work I do is in `R`!

Who are you?

We have a wonderfully diverse set of majors here...

- Global health, pharmacy, nursing, pathology, public-health genetics, epidemiology, rehabilitation science, international studies, psychology, health services, nutrition, public health, health informatics, bioengineering, environmental & occupational health, urban design, speech & hearing science, *definitely others*...

Diversity of thought and experience are at the heart of university education and lifelong learning – **welcome!**

In the in-class exercise today, you will tell us more about your interest in the course and your statistics and programming background...

Learning goal

The objective of this course is to give you the **confidence** and **skills** to perform data analysis in `R`.

Learning process

To achieve this goal, we will

- Learn the syntax of `R`
- Gain experience loading, transforming, summarising and plotting data
- Develop a repertoire of strategies to troubleshoot and expand your understanding of `R`
  - Both *within* and *beyond* the scope of this class

What are we going to learn?

We will cover

- loading, transforming, summarising, and plotting data
- fitting regression models
- writing custom functions

We will not cover

- introductory statistics
- model interpretation
- model building/selection
- hypothesis testing and statistical inference
- statistical machine learning or prediction

I hope this is the right course for you, but please reach out to me if you're not sure!

Achieving our goals

This class has homeworks and in-class exercises. They are intended to help you with your learning process.

In order to receive credit (CR) for this course, you must obtain at least 28 points out of a possible 32 points

- 8 homeworks, 2 points each
- 8 in-class exercises, 2 points each

No grades will be assigned in this course: CR/NC only

Evaluating our progress: in class exercises

There will be in-class exercises made available at the beginning of every class and due at the end of every class

- For 2 points: All responses are correct **or** show a thoughtful attempt at a solution; and the response is submitted on time
  - "On time"
    - The exercises are *designed* to be completed in-class (before 3:20pm)
    - They are *due* at 5:00pm

They are *open* until Monday 5pm; 1 point is available for a late submission

Responses are due via Canvas

Evaluating our progress: homeworks

There will be a homework set made available at the end of every class.

- For 2 points: All responses are correct **or** show a thoughtful attempt at a solution; and the response is submitted on time
  - "On time": 1 p.m. on the Friday one week after the lecture

Responses due via Canvas

How are we going to learn?

Goal: give you the **confidence** and **skills** to perform data analysis in R

How do we create a learning environment where we can achieve this goal together?

How are we going to learn? Class norms

What **class norms** and **ground rules** would you like to set for **yourselves/each other**?

Submit your responses at PollEv.com/adwillis

Feel free to discuss in pairs/small groups. *We will regroup to discuss in 3 minutes.*

How are we going to learn? Class norms

What **class norms** and **ground rules** would you like to set for **me and the TAs**?

Submit your responses at PollEv.com/adwillis

Feel free to discuss in pairs/small groups. *We will regroup to discuss in 3 minutes.*

How are we going to learn?

Our norms are available at:

https://docs.google.com/document/d/1CpXRAzzDBKZ2HivX5okY2LqZ-KRPY2mbT0SmqqLnT3o/edit?usp=sharing

R: Lost in translation

What is R?

R is a "programming environment for statistics and graphics"

- Does *basically* everything, can also be extended
- It's the default when statisticians implement new methods
- Free, open-source

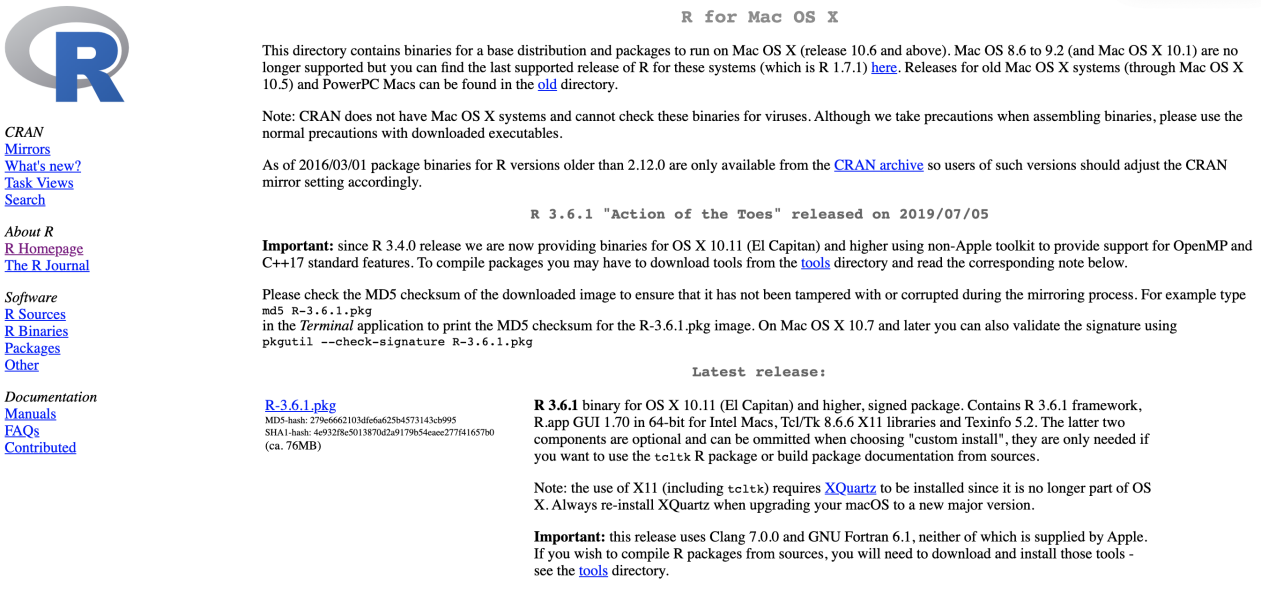
But

- Steeper learning curve than e.g. Excel, Stata
- Command-line driven – you *program*, not click

R is **the** environment for statistical analysis

Installing R

You can download R from https://ftp.osuosl.org/pub/cran/



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

As of 2016/03/01 package binaries for R versions older than 2.12.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting accordingly.

R 3.6.1 "Action of the Toes" released on 2019/07/05

Important: since R 3.4.0 release we are now providing binaries for OS X 10.11 (El Capitan) and higher using non-Apple toolkit to provide support for OpenMP and C++17 standard features. To compile packages you may have to download tools from the tools directory and read the corresponding note below.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`md5 R-3.6.1.pkg`
in the *Terminal* application to print the MD5 checksum for the R-3.6.1.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-3.6.1.pkg`

Latest release:

R-3.6.1.pkg
MD5-hash: 279e6662103dfe6a625b4573143cb995
SHA1-hash: 4e932f8e5013870d2a9179b54eaee277f41657b0
(ca. 76MB)

R 3.6.1 binary for OS X 10.11 (El Capitan) and higher, signed package. Contains R 3.6.1 framework, R.app GUI 1.70 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 5.2. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires XQuartz to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

Important: this release uses Clang 7.0.0 and GNU Fortran 6.1, neither of which is supplied by Apple. If you wish to compile R packages from sources, you will need to download and install those tools - see the tools directory.

No rush/stress – we will do this during the in-class exercise time

Installing R

During the in-class exercise time

1. Go to `https://ftp.osuosl.org/pub/cran/`
2. Click "Download R for (Mac) OS X" or "Download R for Windows"
3. Download the latest release

Working in pairs *highly* recommended

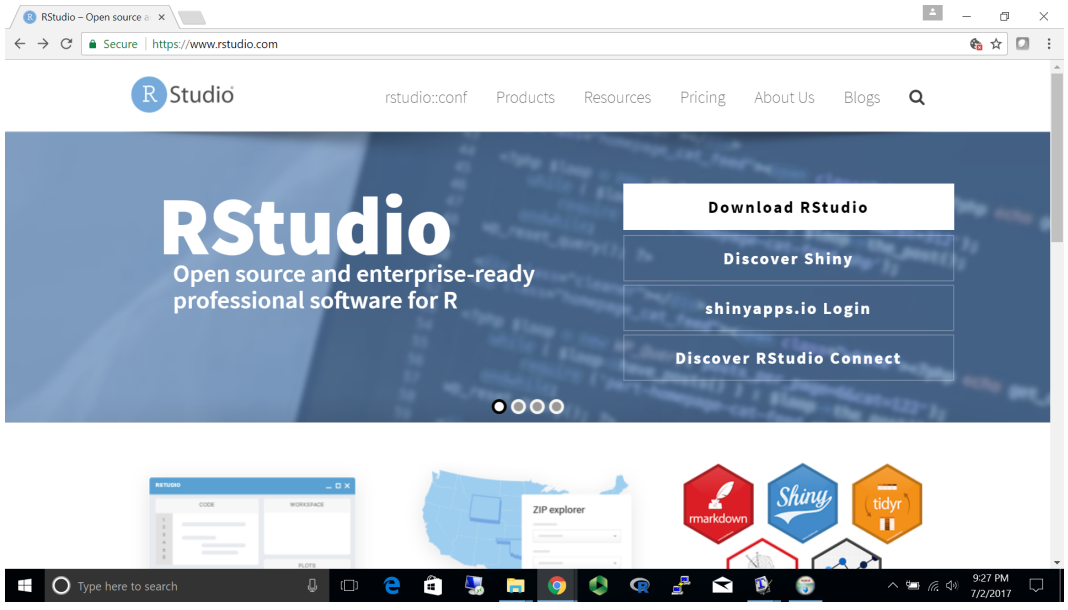
Making life easier with RStudio

We will use RStudio, a front-end for `R`!

1. **First**, installing the latest version of `R`
2. **Then**, install RStudio

RStudio

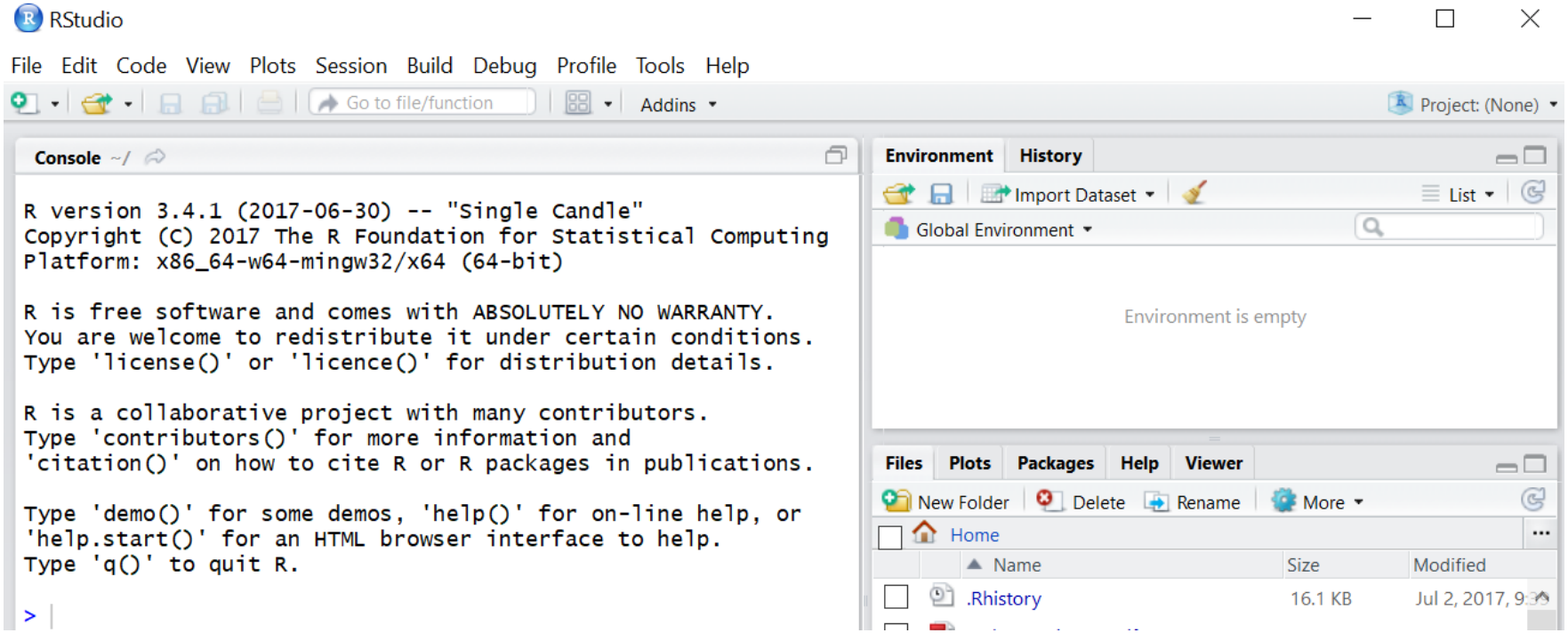
You can download RStudio from rstudio.com/products/rstudio/download/



- Select & download the FREE installer for *your* system
- Choose "RStudio Desktop 1.2.5001"

RStudio

On first startup, RStudio should look like this; (up to version and Mac/PC differences)



If you've used it before, RStudio defaults to remembering what you were doing.

RStudio

Tabs: Console, Environment, Plots, Packages, Help, Files...

RStudio

We'll use the "Console" window first – as a (fancy!) calculator

```
2+2*4
```

```
## [1] 10
```

```
2^5+7
```

```
## [1] 39
```

```
2^(5+7)
```

```
## [1] 4096
```

RStudio

```
0.05/1E6    # a comment; note 1E6 = 1,000,000
```

```
## [1] 5e-08
```

R

R can store data as *objects*. New objects are created when we *assign* them values using "<-"

```
x <- 3
y <- 2 # now check the Environment window
x + y
```

```
## [1] 5
```

This is not the only way to do assignment, but it's how I want you to do assignment

R

Assigning new values to existing objects over-writes the old version. *There is no "undo"*

```
y <- 17.4 # check the Environment window again
x+y
```

```
## [1] 20.4
```

- Anything after a # is ignored – use this to make comments

R: packages

Packages extend the functionality of "base R"

- Packages are officially distributed via CRAN: the Comprehensive R Archive Network

1. If you know the name of the package and it's available from CRAN, use `install.packages("packagename")` to install it onto your system
2. Once the package is installed, you need to load it using `library(packagename)`

R: packages

```
install.packages("tidyverse")
```

R: packages

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------- tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

`tidyverse` is a special package – it's actually a collection of packages, including `ggplot2`, `readr`, and `dplyr`

R: using functions

Once you have `tidyverse` installed and loaded, you can call *functions* on objects in two different ways

```
sqrt(2)
```

```
## [1] 1.414214
```

```
2 %>% sqrt
```

```
## [1] 1.414214
```

R: using functions

- A "+" prompt instead of the usual ">" means the line isn't finished
  - If piping (using %>%), describe the next function
  - If not piping, hit Escape to get out, then try again
- Common math functions are available
  - Remember rules for parentheses: `log(20+5)` and `log(20)+5` are different

RStudio: Reading in data

To read in a comma-separated value (csv) file, type the file path, then a "%>%" then `read_csv`:

```
"/Users/adwillis/teaching/19-509/datasets/fev.csv" %>%
    read_csv
```

```
## Parsed with column specification:
## cols(
##   seqnbr = col_double(),
##   subjid = col_double(),
##   age = col_double(),
##   fev = col_double(),
##   height = col_double(),
##   sex = col_double(),
##   smoke = col_double()
## )
```

```
## # A tibble: 654 x 7
##    seqnbr subjid   age   fev height   sex smoke
##     <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
##  1      1    301     9  1.71   57        0     0
##  2      2    451     8  1.72   67.5      0     0
```

RStudio: Reading in data

Don't forget to store it!

```
fev_data <- "/Users/adwillis/teaching/19-509/datasets/fev.csv" %>%
    read_csv
```

```
## Parsed with column specification:
## cols(
##   seqnbr = col_double(),
##   subjid = col_double(),
##   age = col_double(),
##   fev = col_double(),
##   height = col_double(),
##   sex = col_double(),
##   smoke = col_double()
## )
```

RStudio: Reading in data

To see what it looks like, type the name of the stored dataset

```
fev_data
```

```
## # A tibble: 654 x 7
##    seqnbr subjid   age   fev height   sex smoke
##     <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
##  1      1    301     9  1.71   57       0     0
##  2      2    451     8  1.72   67.5     0     0
##  3      3    501     7  1.72   54.5     0     0
##  4      4    642     9  1.56   53       1     0
##  5      5    901     9  1.90   57       1     0
##  6      6   1701     8  2.34   61       0     0
##  7      7   1752     6  1.92   58       0     0
##  8      8   1753     6  1.42   56       0     0
##  9      9   1901     8  1.99   58.5     0     0
## 10     10   1951     9  1.94   60       0     0
## # ... with 644 more rows
```

No software can cope with every format that might be used to store data, so make sure to check it's sensible (more on this later)...

RStudio: Reading in data

What is this? It's a *tibble* – a (clever) data frame. Look at its header with `head`

```
fev_data %>% head
```

```
## # A tibble: 6 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1      1    301     9  1.71   57       0     0
## 2      2    451     8  1.72   67.5     0     0
## 3      3    501     7  1.72   54.5     0     0
## 4      4    642     9  1.56   53       1     0
## 5      5    901     9  1.90   57       1     0
## 6      6   1701     8  2.34   61       0     0
```

Operating on data: columns

Individual columns are identified using the $ symbol

```
fev_data$fev
```

```
##   [1] 1.708 1.724 1.720 1.558 1.895 2.336 1.919 1.415 1.
##  [12] 1.735 2.193 2.118 2.258 1.932 1.472 1.878 2.352 2.
##  [23] 0.839 2.578 2.988 1.404 2.348 1.755 2.980 2.100 1.
##  [34] 2.093 1.612 2.175 2.725 2.071 1.547 2.004 3.135 2.
##  [45] 1.343 2.076 1.624 1.344 1.650 2.732 2.017 2.797 3.
##  [56] 2.570 3.016 2.419 1.569 1.698 2.123 2.481 1.481 1.
##  [67] 2.069 1.631 1.536 2.560 1.962 2.531 2.715 2.457 2.
##  [78] 1.452 3.842 1.719 2.111 1.695 2.211 1.794 1.917 2.
##  [89] 1.580 2.126 3.029 2.964 1.611 2.215 2.388 2.196 1.
## [100] 1.523 1.292 1.649 2.588 0.796 2.574 1.979 2.354 1.
## [111] 2.639 1.829 2.084 2.220 1.473 2.341 1.698 1.196 1.
## [122] 1.827 1.461 1.338 2.090 1.697 1.562 2.040 1.609 2.
## [133] 1.675 1.947 2.069 1.572 1.348 2.288 1.773 0.791 1.
## [144] 2.631 3.114 2.135 1.527 2.293 3.042 2.927 2.665 2.
```

Operating on data: columns

No need to print out the whole thing! Just grab the `header` (first 6 elements), or a 6-number `summary`

```
fev_data$fev %>% head
```

```
## [1] 1.708 1.724 1.720 1.558 1.895 2.336
```

```
fev_data$fev %>% summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.791   1.981   2.547   2.637   3.119   5.793
```

```
fev_data$fev %>% length
```

```
## [1] 654
```

Operating on data: columns

Some other functions useful for summarizing tibbles/data frames. . .

```
fev_data %>% names # column names
```

```
## [1] "seqnbr" "subjid" "age"    "fev"    "height" "sex"
```

```
names(fev_data) # a different way to do the same thing!
```

```
## [1] "seqnbr" "subjid" "age"    "fev"    "height" "sex"
```

```
dim(fev_data)       # dimension
```

```
## [1] 654   7
```

Operating on data: columns

Some other functions useful for columns. . .

```
max(fev_data$height)
```

```
## [1] 74
```

```
mean(fev_data$fev)
```

```
## [1] 2.63678
```

```
sd(fev_data$fev) # standard deviation
```

```
## [1] 0.8670591
```

You can usually guess these names!

RStudio: the Script window

While fine for occasional use, entering *every* command *every time* is error-prone and tedious.

Solution: Use a Script! Open one with Ctrl-Shift-N, or the drop-down menus

[Live demo]

- Save your commands as `.R` files for easy re-running
- Run current line (or selected lines) with `Ctrl-Enter`, or `Ctrl-R`

RStudio: the Script window

Always save your code as a script!

- You can submit your homework as a `.R` file
  - To encourage you, the homework template is a `.R` file
- Exercise "solutions" given as .R files
- Scripts make it easy to run slightly modified code, without re-typing everything
- Save your work as you go!

The ability to save and run scripts is one of the main advantages of `R`!

Operating on data: subsets

To identify general subsets – not just the columns selected by `$` – R uses square brackets. Selecting individuals elements

```
fev_data$fev[32] # 32nd element of fev_data$fev
```

```
## [1] 3
```

```
fev_data$height[32]
```

```
## [1] 65.5
```

Operating on data: subsets

Can also select entire columns or entire rows this way

```
fev_data[32, 3] # subtable with just 32nd row, 3rd column
```

```
## # A tibble: 1 x 1
##     age
##   <dbl>
## 1     9
```

```
fev_data[32, "age"] # same thing
```

```
## # A tibble: 1 x 1
##     age
##   <dbl>
## 1     9
```

I encourage you to use the 2nd approach – it's more readable and robust!

Operating on data: subsets

Can ‘blank" entries to indicate you want everything:

```
fev_data[32, ]      # everything in the 32nd row
```

```
## # A tibble: 1 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     32   7201     9     3   65.5     1     0
```

Operating on data: subsets

Suppose we were interested in the FEV (i.e. 4th column) for rows 14, 55, & 61. How to select these multiple elements?

```
fev_data[c(14, 55, 61), 4]
```

```
## # A tibble: 3 x 1
##     fev
##   <dbl>
## 1  2.12
## 2  1.63
## 3  2.12
```

Operating on data: subsets

But what is `c(14, 55, 61)`? It's a *vector* of numbers – `c()` is for *combine*

```
c(14, 55, 61)
```

```
## [1] 14 55 61
```

```
c(14, 55, 61) %>% length
```

```
## [1] 3
```

Operating on data: subsets

We can select these rows and *all* the columns

```
fev_data[c(14, 55, 61),]
```

```
## # A tibble: 3 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     14   2401     8  2.12   60.5     1     0
## 2     55  12241     6  1.63   54       1     0
## 3     61  14101     8  2.12   60       1     0
```

Operating on data: subsets

A very useful special form of vector

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
6:2
```

```
## [1] 6 5 4 3 2
```

```
-1:-3
```

```
## [1] -1 -2 -3
```

Operating on data: subsets

For a "rectangular" selection of rows and columns

```
fev_data[20:22, 1:3]
```

```
## # A tibble: 3 x 3
##   seqnbr subjid   age
##    <dbl>  <dbl> <dbl>
## 1     20   4301     9
## 2     21   4351     5
## 3     22   5151     5
```

Operating on data: subsets

Negative values correspond to *dropping* those rows/columns;

```
fev_data[c(1,3,4), -4:-7]
```

```
## # A tibble: 3 x 3
##   seqnbr subjid   age
##    <dbl>  <dbl> <dbl>
## 1      1    301     9
## 2      3    501     7
## 3      4    642     9
```

We will see how to drop columns based on exclusion criteria next week

Operating on data: subsets

As well as storing numbers and character strings (like "Users/data/mydata.csv"), `R` can also store *logicals* – `TRUE` and `FALSE`.

To make a new vector, with elements that are TRUE if height is above 72cm and FALSE otherwise

```
is.tall <- fev_data$height > 72
```

Operating on data: subsets

```
is.tall
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [111] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [122] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
## [166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
```

Summarising data

Another useful data summary command:

```
table(is.tall)
```

```
## is.tall
## FALSE  TRUE
##   647     7
```

Operating on data: subsets

Which fev_data were these? (And what were their ages?)

```
fev_data[is.tall, ]
```

```
## # A tibble: 7 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    401  18841    14  4.27   72.5     1     0
## 2    450  32741    13  4.22   74       1     0
## 3    464  37241    13  4.88   73       1     0
## 4    517  49541    13  5.08   74       1     0
## 5    550  59941    14  4.27   72.5     1     0
## 6    632  37441    17  5.63   73       1     0
## 7    636  44241    16  3.64   73.5     1     0
```

This prints just the rows for which `is.tall` is `TRUE`

Quitting time (almost)

When you're finished with RStudio;

- Ctrl-Q, or the drop-down menus, or entering `q()` at the command line all start the exit process
- You will be asked "Save workspace image to...?"
  - No/Don't Save: Your *objects* are not saved. *This is recommended for now.*
  - Yes: Your *objects* are stored in R's internal format (`.Rdata`) and will be available when you return
  - Cancel: don't quit, go back

For now, *save your `.R` script*... but not your objects

Summary

- R is the environment, RStudio is the front end
- R is extendable with packages
  - Start by installing and loading `tidyverse`
- Store objects in R with <-
- Read in data in `csv` files using `read_csv`
- Access rows/columns of a data frame by name or index
- Many useful summary functions are available, with sensible names
- Scripts are important to avoid repeating work!

The plan

0. Take a 5 minute break
    - Perhaps while `R` and `RStudio` are downloading
1. In-class exercise available via Canvas and github.com/adw96/biost509
    - Due today 5 p.m.
    - Place a *yellow sticky note* on your computer to indicate you are stuck, or a *blue sticky note* to indicate you have a non-urgent question
2. Homework due next week by 1 p.m.
3. Office hours on the syllabus

The TAs and I will walk around to help you; we will help *yellow stickies*, then *blue stickies*