Reproducible (and collaborative) science through RStudio

A whirlwind tour with R, RMarkdown, Python, LaTeX, and more

Jenny Rieck & Derek Beaton

May 22 2019

The big outline

- Part 0: Background
- Part 1: A bit about R
- Part 2: RStudio & Project setup
- Part 3: R
- Part 4: RMarkdown & more
- Part 5: “Advanced” topics

Part 0: Background

To dive right in

If you want to skip over the background & RStudio, go straight to Part 2: RStudio & Project setup

Background

- This is a taste and to bring you into a bigger world
  - R, Python, SQL, and JavaScript are critical data science tools/languages
- R (language and community) strongly emphasizes
  - Centralization & standards
  - Rigor & reproducibility (packages, RMarkdown)
  - An interesting language
- Functional
  - With a sublanguage (or dialect?): the `tidyverse`

R is a community (actually many communities!)

- Help and resources
- Package development and distribution
- An ideal example
  - Not quite always that way
  - Strong communal presence

R: Help!

- So many websites e.g., https://www.statmethods.net/
- Online forums (Stack Exchange, r-lists)
- SpringerLink
  - All R books for free (pdf format) or for minimal cost (printed)
- Vignettes
  - step-by-step instruction guides for packages
- Git
  - With open books (via bookdown)
- Twitter #rstats
- RStudio (website)
  - Videos, cheat sheets

R Packages

- Packages are bundles of code made by someone (or many people) for everyone to use
- There are packages for everything
- We'll cover some of the diversity throughout
- Comprehensive & Reproducible
  - Available primarily on CRAN
  - But also github (less so: r-forge)

Part 1: A bit about R

R Background

- Created in 1992 by Gentleman & Ihaka

[we] considered the problem of obtaining decent statistical software for our undergraduate Macintosh lab. After considering the options, we decided that the most satisfactory alternative was to write our own. [. . .] Finally we added some syntactic sugar to make it look somewhat like S. We call the result "R".

What is R?

- R is general purpose programming
  - Design around & for statistics
  - "for and by statisticians"
- R is a collection of tools
  - Pre-packaged software at your disposal
- R is free (as in beer and speech)
  - No cost, no restrictions
  - E.g., Microsoft (nee Revolution) R
- R is a functional language
  - Turing complete
  - Mathematical functions
  - Pass expressions and functions to and from functions

Some syntax notes

Assignment

```
# allowed but not preferred
a_variable = 10 + 1
# preferred
a_variable <- 10 + 1
# a bonus
10 + 1 -> a_variable
```

Dots

```
# allowed but not preferred
a.variable = 10 + 1
  ## dots have 2 meanings in R,
    ## with a 3rd in the tidyverse

# preferred
a_variable <- 10 + 1
```

“Reserved” characters

- `c`, `q`, `t`, `C`, `D`, `I`, `F`, and `T` (via https://www.johndcook.com/blog/r_language_for_programmers/)
- Except that these can be redefined
  - With great power comes great responsibility (and danger)

R: Data types

- Stored as *vectors*
  - see `class()`
- numeric
  - real or decimal
  - Includes `NaN`, `Inf`, `-Inf`
- integer
- complex
- character
- logical
  - includes `NA`, `TRUE`, `FALSE`
- factor
  - factors are usually not your friends
  - with `read.csv()`: `stringsAsFactors = F` or convert these
  - or use `tibbles` in the tidyverse

R: factor disasters

```
a_numeric_vector <- c(3, 0, 1, -2, 2, 5, 5, 2, 1)
(a_numeric_vector + 1)
```

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

```
(a_numeric2factor_vector <- as.factor(a_numeric_vector))
```

```
## [1] 3  0  1  -2 2  5  5  2  1
## Levels: -2 0 1 2 3 5
```

```
(as.numeric(a_numeric2factor_vector))
```

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

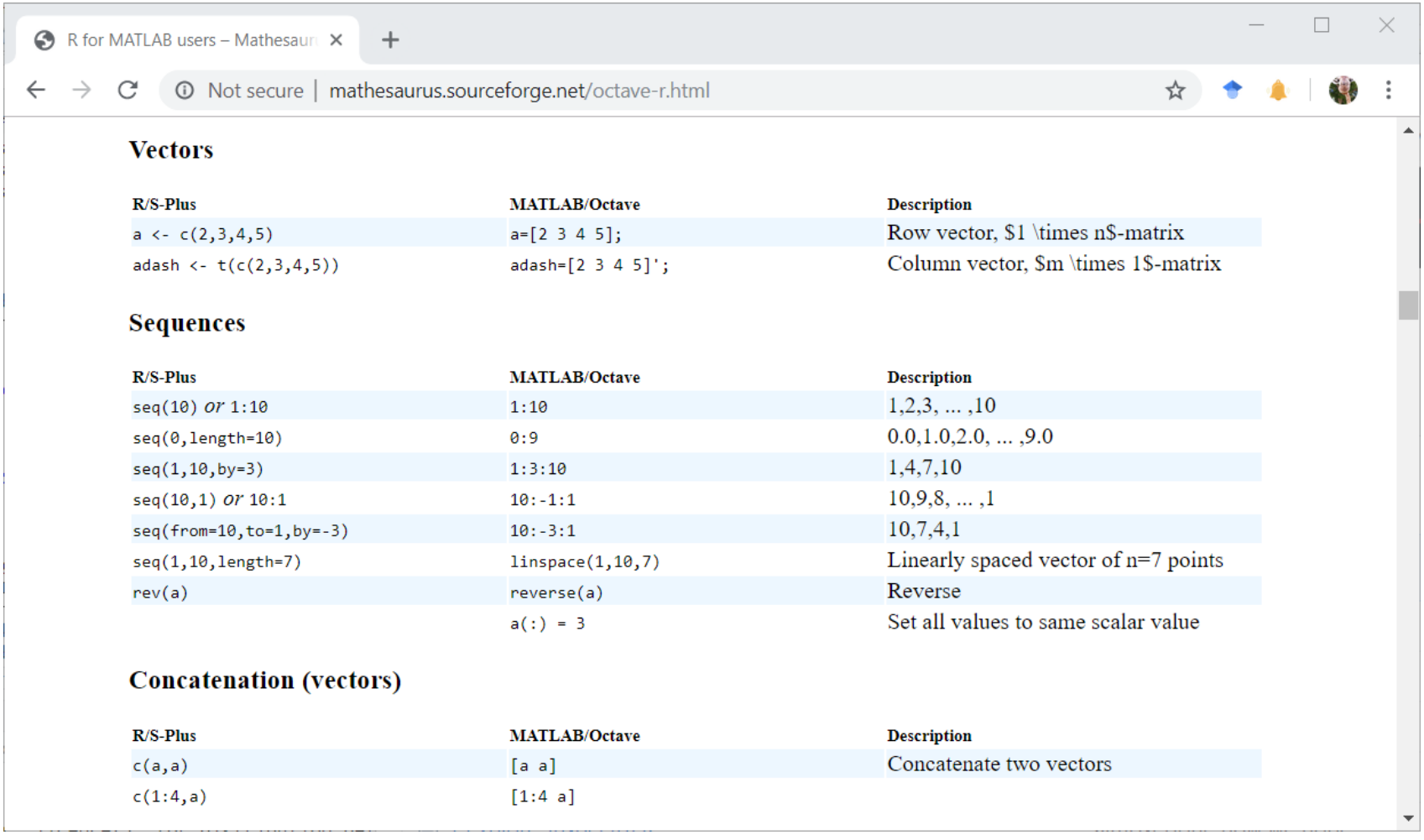
```
(as.numeric(as.character(a_numeric2factor_vector)))
```

```
## [1]  3  0  1 -2  2  5  5  2  1
```

R: Data structures

- Starts counting from 1
  - Not 0
- vector[1]
- matrix[1,1]
- array[1,1,1]
- list[[1]]
  - Can contain mixtures of types
  - or list$`name`
- data.frame:
  - Is technically a list but access in three ways
  - data.frame[[1]][1]
  - data.frame[1,1]
  - data.frame$`name`
  - tibbles: tidyverse data.frames

Cheatsheet for going from MatLab to R



Vectors

R/S-Plus	MATLAB/Octave	Description
`a <- c(2,3,4,5)`	`a=[2 3 4 5];`	Row vector, $1 \times n$-matrix
`adash <- t(c(2,3,4,5))`	`adash=[2 3 4 5]';`	Column vector, $m \times 1$-matrix

Sequences

R/S-Plus	MATLAB/Octave	Description
`seq(10)` *or* `1:10`	`1:10`	1,2,3, ... ,10
`seq(0,length=10)`	`0:9`	0.0,1.0,2.0, ... ,9.0
`seq(1,10,by=3)`	`1:3:10`	1,4,7,10
`seq(10,1)` *or* `10:1`	`10:-1:1`	10,9,8, ... ,1
`seq(from=10,to=1,by=-3)`	`10:-3:1`	10,7,4,1
`seq(1,10,length=7)`	`linspace(1,10,7)`	Linearly spaced vector of n=7 points
`rev(a)`	`reverse(a)`	Reverse
	`a(:) = 3`	Set all values to same scalar value

Concatenation (vectors)

R/S-Plus	MATLAB/Octave	Description
`c(a,a)`	`[a a]`	Concatenate two vectors
`c(1:4,a)`	`[1:4 a]`	

http://mathesaurus.sourceforge.net/octave-r.html

Cheatsheet for base R

Base R
Cheat Sheet

Getting Help

Accessing the help files

```
?mean
```
Get help of a particular function.
```
help.search('weighted mean')
```
Search the help files for a word or phrase.
```
help(package = 'dplyr')
```
Find help for a package.

More about an object

```
str(iris)
```
Get a summary of an object's structure.
```
class(iris)
```
Find the class an object belongs to.

Using Libraries

```
install.packages('dplyr')
```
Download and install a package from CRAN.

```
library(dplyr)
```
Load the package into the session, making all its functions available to use.

```
dplyr::select
```
Use a particular function from a package.

```
data(iris)
```
Load a built-in dataset into the environment.

Working Directory

```
getwd()
```
Find the current working directory (where inputs are found and outputs are sent).

```
setwd('C://file/path')
```
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

`c(2, 4, 6)`	2 4 6	Join elements into a vector
`2:6`	2 3 4 5 6	An integer sequence
`seq(2, 3, by=0.5)`	2.0 2.5 3.0	A complex sequence
`rep(1:2, times=3)`	1 2 1 2 1 2	Repeat a vector
`rep(1:2, each=3)`	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

`sort(x)` Return x sorted.
`rev(x)` Return x reversed.
`table(x)` See counts of values.
`unique(x)` See unique values.

Selecting Vector Elements

By Position

`x[4]`	The fourth element.
`x[-4]`	All but the fourth.
`x[2:4]`	Elements two to four.
`x[-(2:4)]`	All elements except two to four.
`x[c(1, 5)]`	Elements one and five.

By Value

`x[x == 10]`	Elements which are equal to 10.
`x[x < 0]`	All elements less than zero.
`x[x %in% c(1, 2, 5)]`	Elements in the set 1, 2, 5.

Named Vectors

`x['apple']`	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

Reading and Writing Data

Input	Ouput	Description
`df <- read.table('file.txt')`	`write.table(df, 'file.txt')`	Read and write a delimited text file.
`df <- read.csv('file.csv')`	`write.csv(df, 'file.csv')`	Read and write a comma separated value file. This is a special case of read.table/ write.table.
`load('file.RData')`	`save(df, file = 'file.Rdata')`	Read and write an R data file, a file type special for R.

Conditions

`a == b`	Are equal	`a > b`	Greater than	`a >= b`	Greater than or equal to	`is.na(a)`	is missing
`a != b`	Not equal	`a < b`	Less than	`a <= b`	Less than or equal to	`is.null(a)`	is null

Tidyverse

- R is "base R"
- `tidyverse`: "an opinionated collection of R packages designed for data science [that] share an underlying design philosophy, **grammar, and data structures.**"
  - Started(?) with/because of `ggplot2`
  - A sublanguage or dialect
  - Can do so because R is functional language
  - For full tidy immersion see: https://style.tidyverse.org/
- Strongly built around a style:
  - objects are nouns
  - functions are verbs
- Core packages:
  - `ggplot2`, `dplyr`, `tidyr`, `readr`, `tibble`, `stringr`
- Get them all with `install.packages("tidyverse")`
- Learn it!
  - But don't learn *only* the tidyverse, you'll be lost in base R

tidyverse cheatsheet

R For Data Science *Cheat Sheet*

Tidyverse for Beginners

Learn More R for Data Science Interactively at www.datacamp.com

Tidyverse

The tidyverse is a powerful collection of R packages that are actually data tools for transforming and visualizing data. All packages of the tidyverse share an underlying philosophy and common APIs.

The core packages are:

- ggplot2, which implements the grammar of graphics. You can use it to visualize your data.
- dplyr is a grammar of data manipulation. You can use it to solve the most common data manipulation challenges.
- tidyr helps you to create tidy data or data where each variable is in a column, each observation is a row end each value is a cell.
- readr is a fast and friendly way to read rectangular data.
- purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors.
- tibble is a modern re-imagining of the data frame.
- stringr provides a cohesive set of functions designed to make working with strings as easy as posssible
- forcats provide a suite of useful tools that solve common problems with factors.

You can install the complete tidyverse with:

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

Then, load the core tidyverse and make it available in your current R session by running:

```
> library(tidyverse)
```

Note: there are many other tidyverse packages with more specialised usage. They are not loaded automatically with library(tidyverse), so you'll need to load each one with its own call to `library()`.

Useful Functions

```
> tidyverse_conflicts()
> tidyverse_deps()
> tidyverse_logo()
> tidyverse_packages()
> tidyverse_update()
```

Conflicts between tidyverse and other packages
List all tidyverse dependencies
Get tidyverse logo, using ASCII or unicode characters
List all tidyverse packages
Update tidyverse packages

Loading in the data

```
> library(datasets)
> library(gapminder)
> attach(iris)
```

Load the datasets package
Load the gapminder package
Attach iris data to the R search path

dplyr

Filter

`filter()` allows you to select a subset of rows in a data frame.

```
> iris %>%
    filter(Species=="virginica")
> iris %>%
    filter(Species=="virginica",
           Sepal.Length > 6)
```

Select iris data of species "virginica"
Select iris data of species "virginica" and sepal length greater than 6.

Arrange

`arrange()` sorts the observations in a dataset in ascending or descending order based on one of its variables.

```
> iris %>%
    arrange(Sepal.Length)
> iris %>%
    arrange(desc(Sepal.Length))
```

Sort in ascending order of sepal length
Sort in descending order of sepal length

Combine multiple `dplyr` verbs in a row with the pipe operator `%>%`:

```
> iris %>%
    filter(Species=="virginica") %>%
    arrange(desc(Sepal.Length))
```

Filter for species "virginica" then arrange in descending order of sepal length

Mutate

`mutate()` allows you to update or create new columns of a data frame.

```
> iris %>%
    mutate(Sepal.Length=Sepal.Length*10)
> iris %>%
    mutate(SLMm=Sepal.Length*10)
```

Change Sepal.Length to be in millimeters
Create a new column called SLMm

Combine the verbs `filter()`, `arrange()`, and `mutate()`:

```
> iris %>%
    filter(Species=="Virginica") %>%
    mutate(SLMm=Sepal.Length*10) %>%
    arrange(desc(SLMm))
```

Summarize

`summarize()` allows you to turn many observations into a single data point.

```
> iris %>%
    summarise(medianSL=median(Sepal.Length))
> iris %>%
    filter(Species=="virginica") %>%
    summarise(medianSL=median(Sepal.Length))
```

Summarize to find the median sepal length
Filter for virginica then summarize the median sepal length

You can also summarize multiple variables at once:

```
> iris %>%
    filter(Species=="virginica") %>%
    summarise(medianSL=median(Sepal.Length),
              maxSL=max(Sepal.Length))
```

`group_by()` allows you to summarize within groups instead of summarizing the entire dataset:

```
> iris %>%
    group_by(Species) %>%
    summarise(medianSL=median(Sepal.Length),
              maxSL=max(Sepal.Length))
> iris %>%
    filter(Sepal.Length>6) %>%
    group_by(Species) %>%
    summarise(medianPL=median(Petal.Length),
              maxPL=max(Petal.Length))
```

Find median and max sepal length of each species

Find median and max petal length of each species with sepal length > 6

ggplot2

Scatter plot

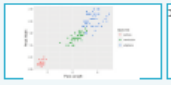
Scatter plots allow you to compare two variables within your data. To do this with ggplot2, you use `geom_point()`

```
> iris_small <- iris %>%
    filter(Sepal.Length > 5)
> ggplot(iris_small, aes(x=Petal.Length,
                         y=Petal.Width)) +
    geom_point()
```

Compare petal width and length

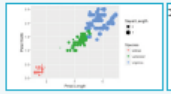
Additional Aesthetics

- Color



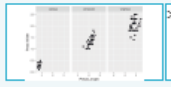
```
> ggplot(iris_small, aes(x=Petal.Length,
                         y=Petal.Width,
                         color=Species)) +
    geom_point()
```

- Size



```
> ggplot(iris_small, aes(x=Petal.Length,
                         y=Petal.Width,
                         color=Species,
                         size=Sepal.Length)) +
    geom_point()
```

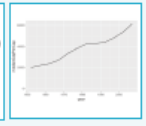
Faceting



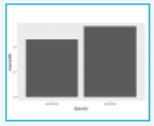
```
> ggplot(iris_small, aes(x=Petal.Length,
                         y=Petal.Width)) +
    geom_point()+
    facet_wrap(~Species)
```

Line Plots

```
> by_year <- gapminder %>%
    group_by(year) %>%
    summarise(medianGdpPerCap=median(gdpPercap))
> ggplot(by_year, aes(x=year,
                      y=medianGdpPerCap))+
    geom_line()+
    expand_limits(y=0)
```



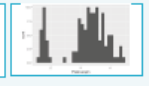
Bar Plots



```
> by_species <- iris %>%
    filter(Sepal.Length>6) %>%
    group_by(Species) %>%
    summarise(medianPL=median(Petal.Length))
> ggplot(by_species, aes(x=Species,
                         y=medianPL)) +
    geom_col()
```

Histograms

```
> ggplot(iris_small, aes(x=Petal.Length))+
    geom_histogram()
```



Box Plots

```
> ggplot(iris_small, aes(x=Species,
                         y=Sepal.Width))+
    geom_boxplot()
```

DataCamp
Learn R for Data Science Interactively

Part 2: RStudio & Project setup

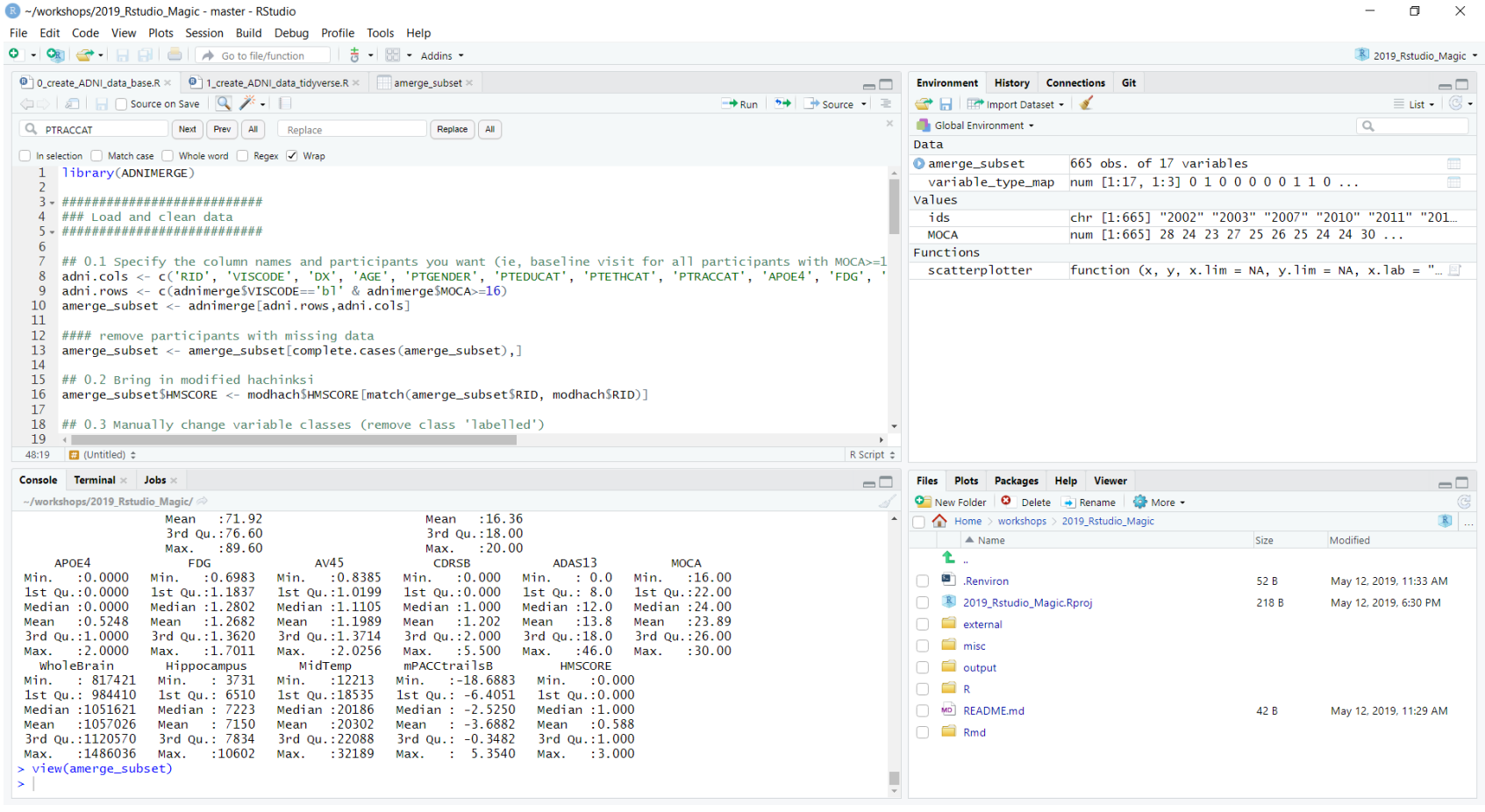
RStudio

- IDE: Integrated development environment
- RStudio: Does so much
  - We scratch the surface here
- Quick walk through
  - Followed by specific set up
  - Generally, but
  - Also for this workshop

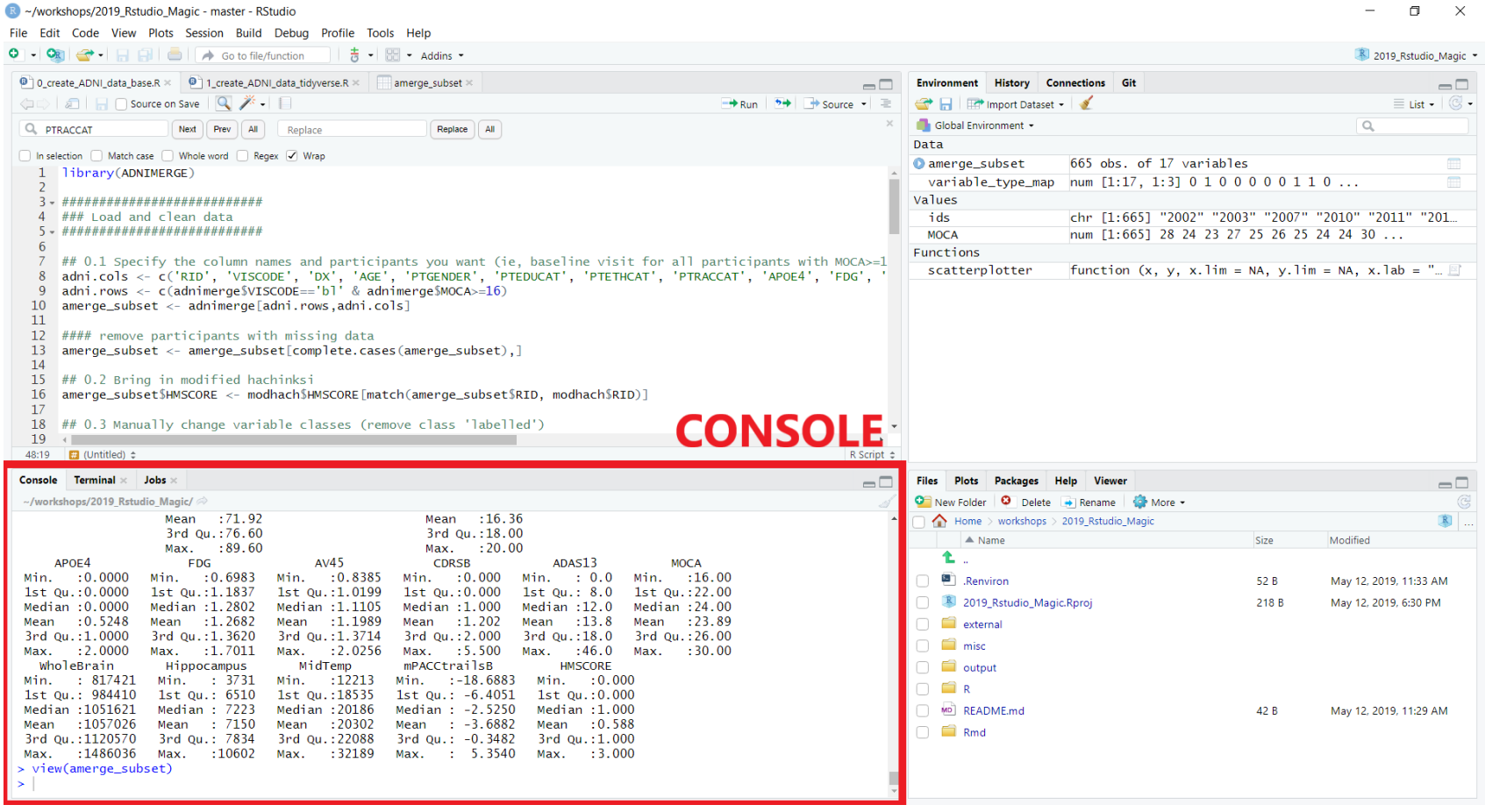
RStudio Setup

- Download R and Rstudio
  - Strongly recommend Microsoft R (https://mran.microsoft.com/open)
  - Comes with Intel MKL
- Plain R is fine (https://cran.r-project.org/)
  - Can relink to faster libraries
- Download RStudio (https://www.rstudio.com/)

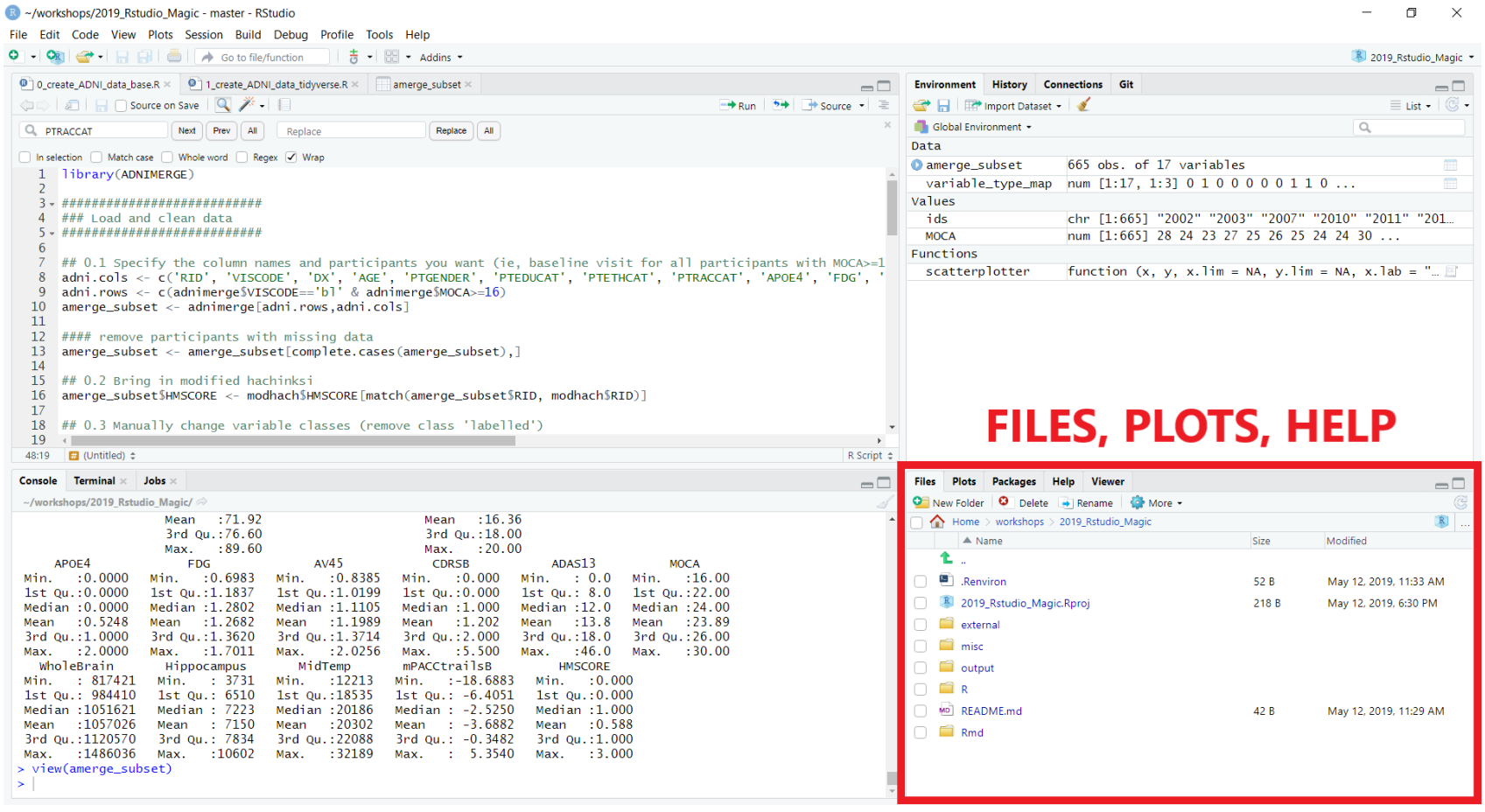
RStudio Environment



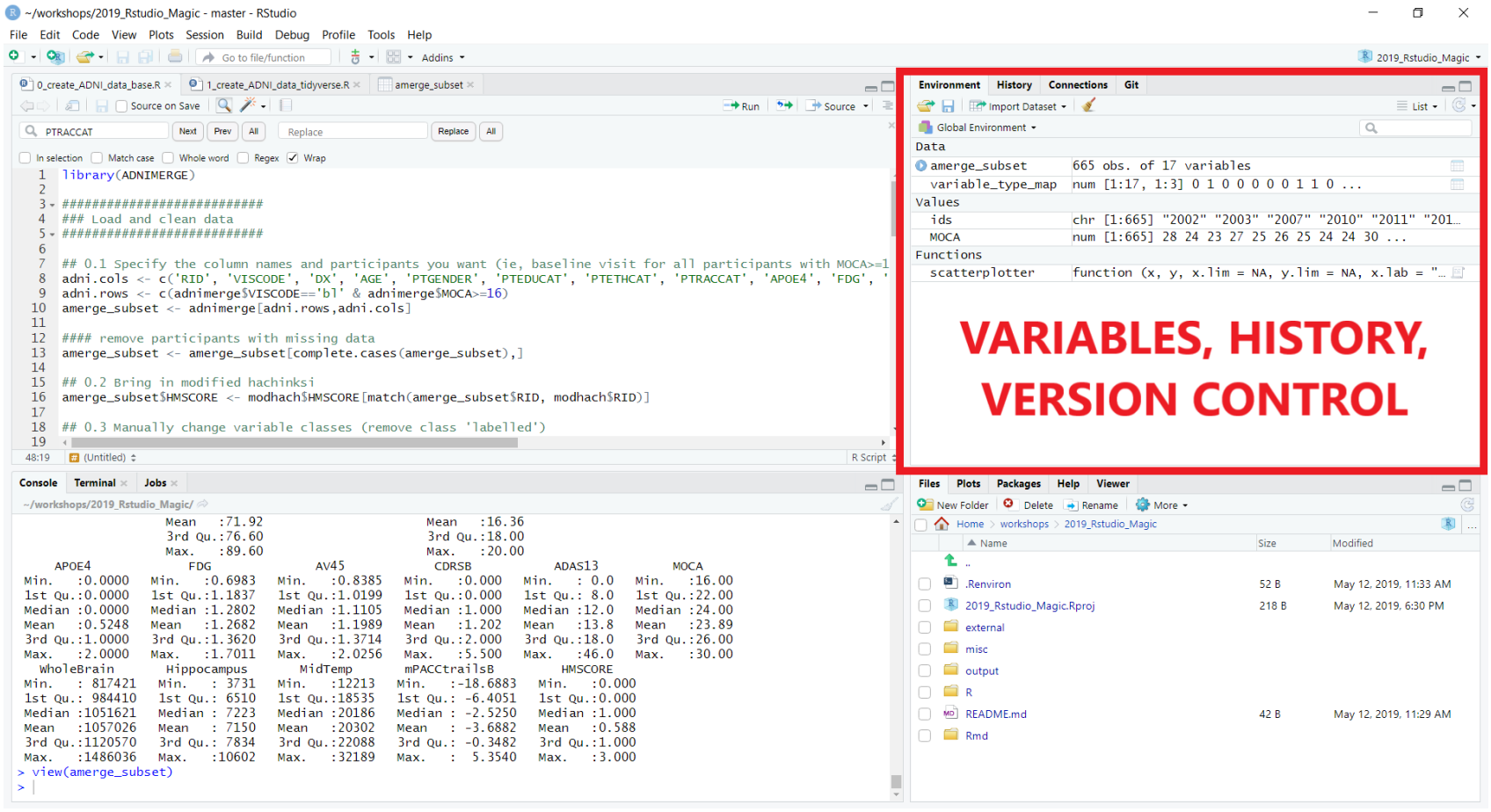
RStudio Environment



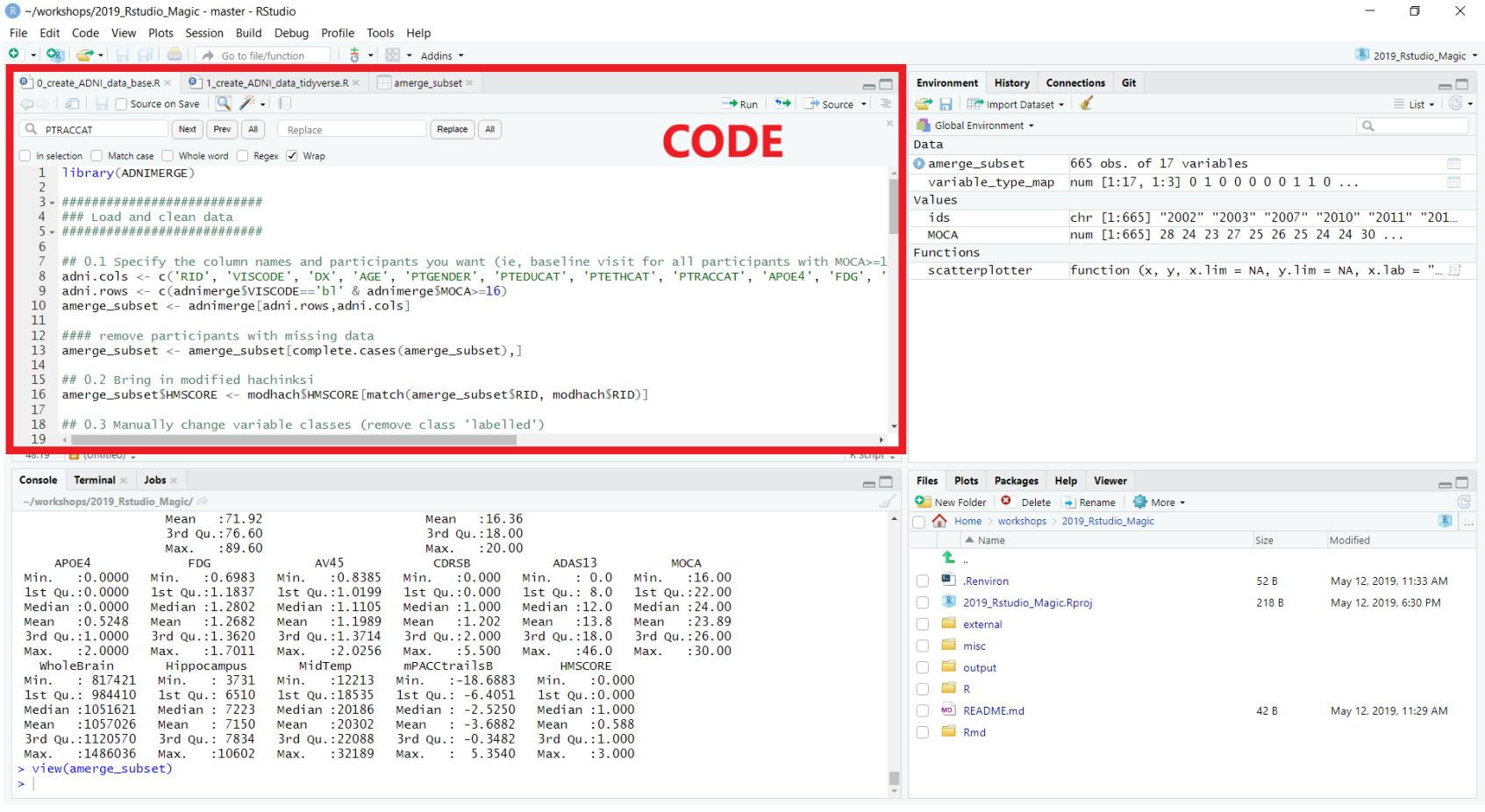
RStudio Environment



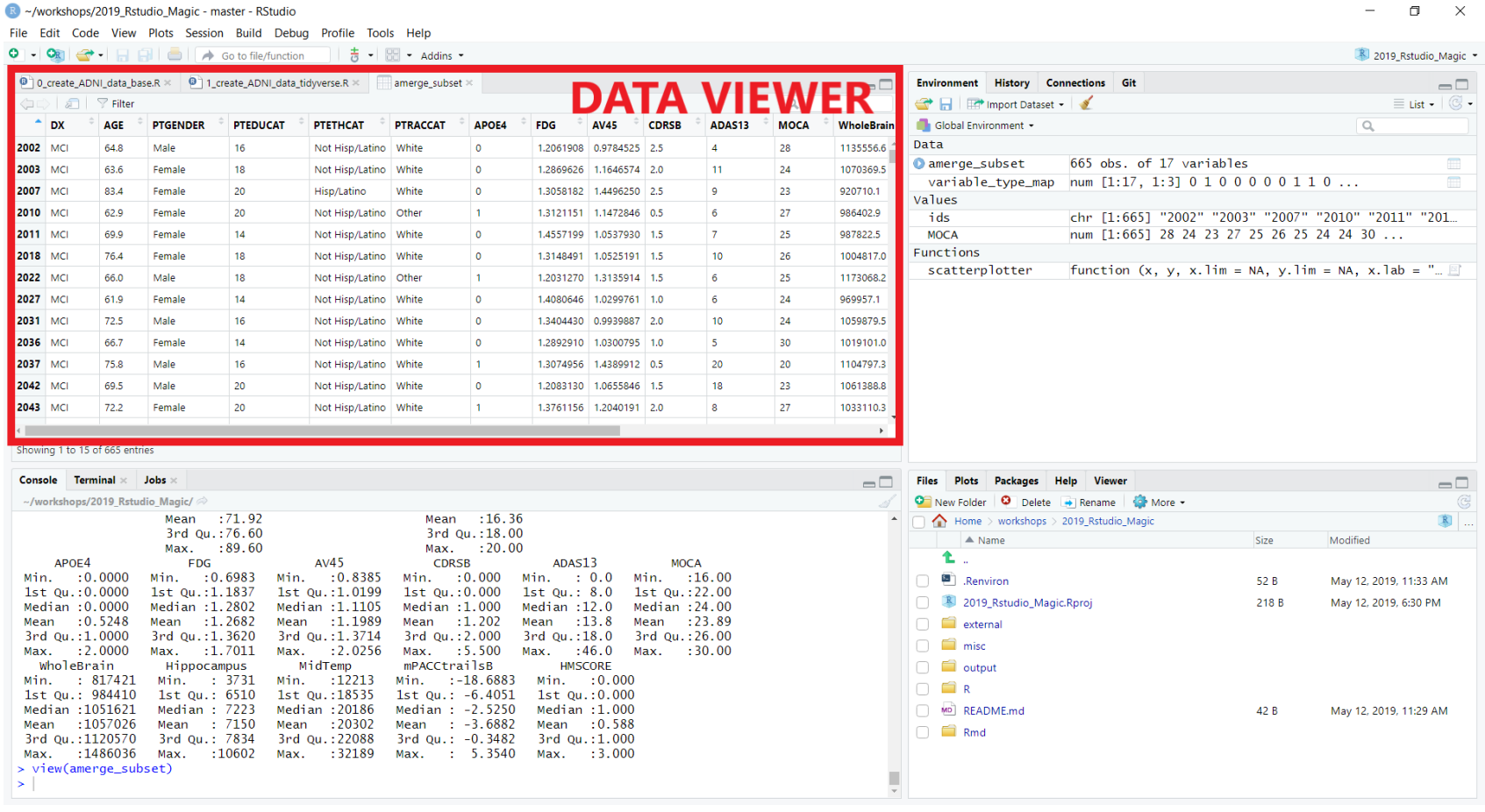
RStudio Environment



RStudio Environment



RStudio Environment



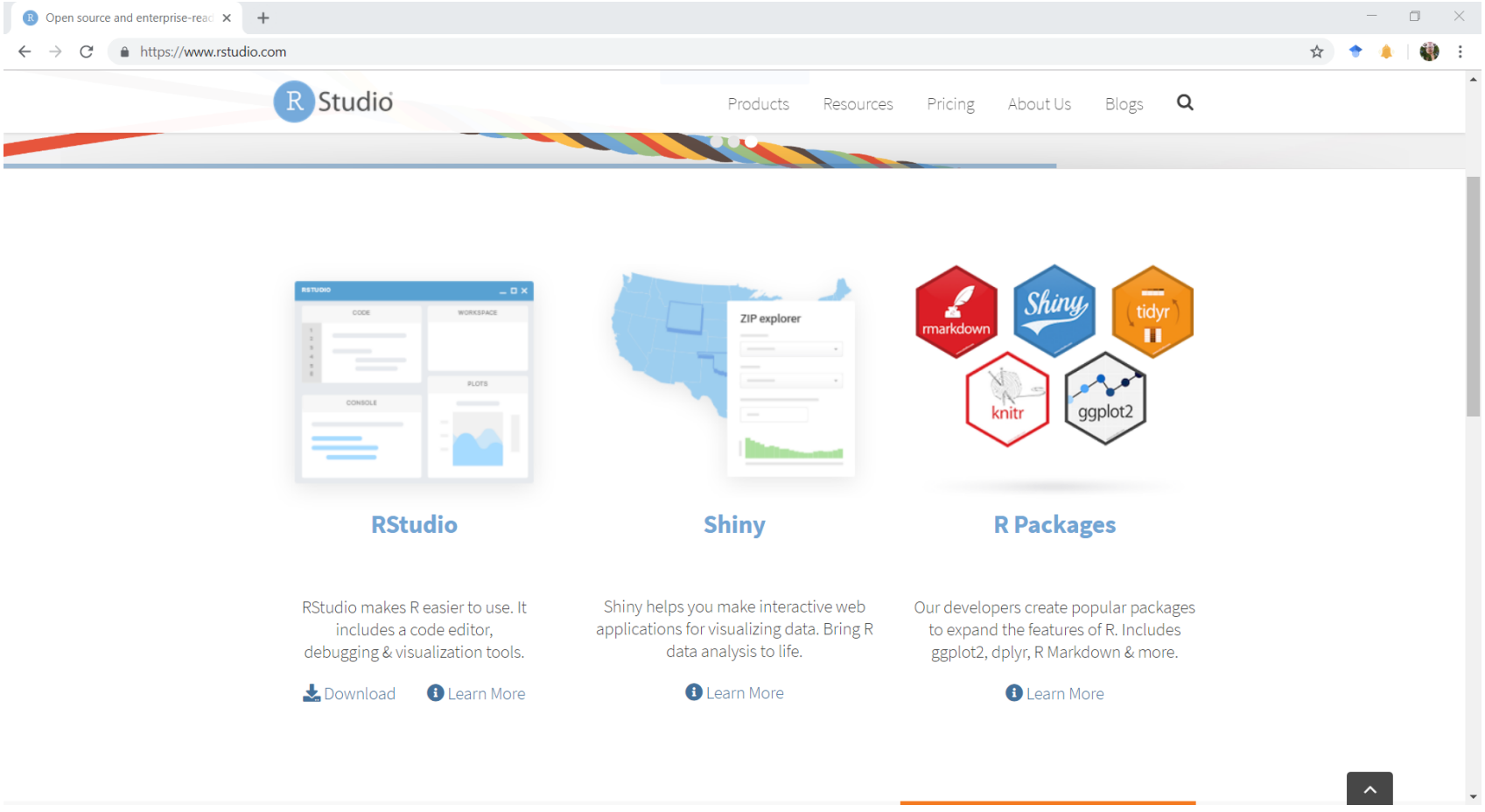
Some benefits of RStudio

- Built-in integration with version control (git or SVN)
- R Markdown
  - Save and execute code
  - Generate high quality reports that can be shared
  - Create presentations (like this one!)
  - Even write papers
  - This workshop
    - See https://github.com/jennyrieck/workshops/tree/master/2019_Rstudio_Magic
- Python, D3 (JavaScript), SQL, Shiny, LaTeX, Git/SVN, HTML/CSS, and so much more.

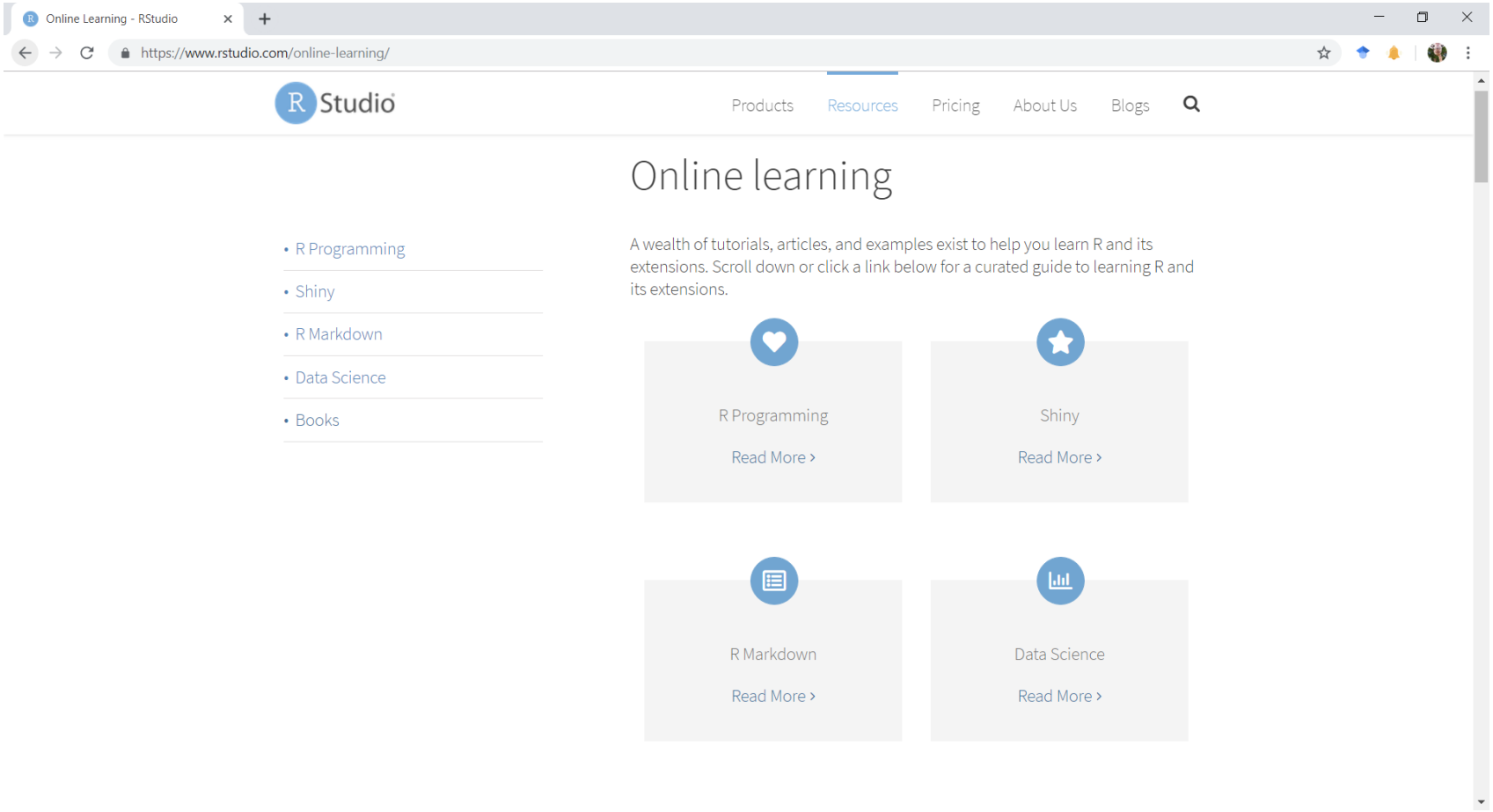
RStudio is more

- Not just an IDE
- A company
- A community
- A conference
- A centralized resource

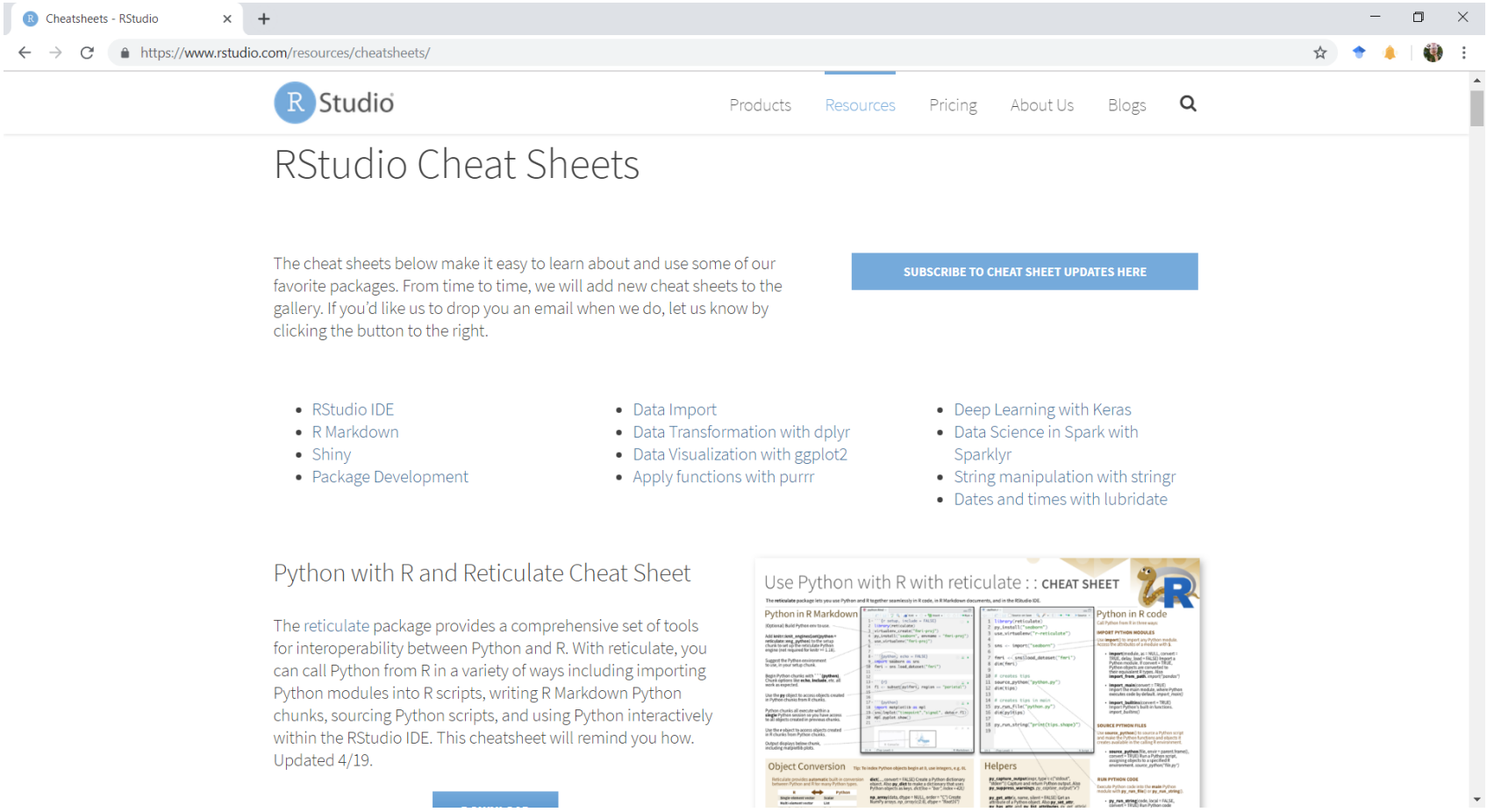
RStudio Resources



RStudio Resources



RStudio Resources



Project and Environment Setup

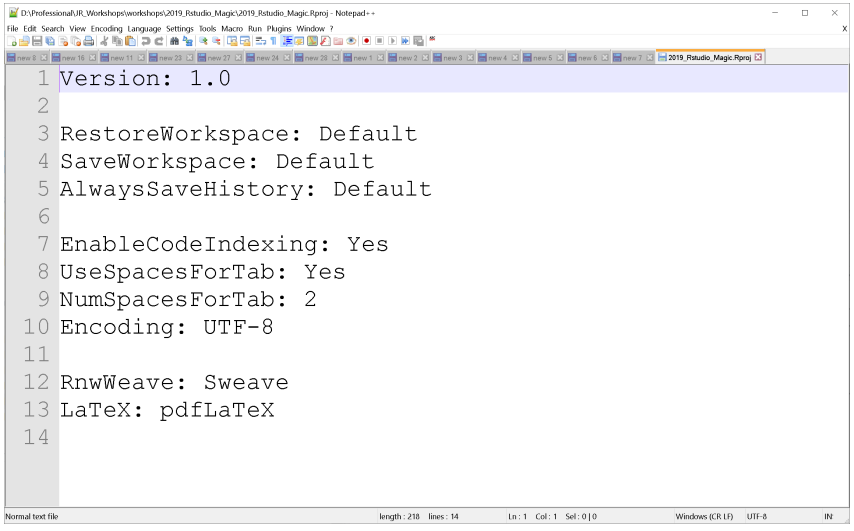
- Special & hidden files
- Having a structure

RStudio Setup

- See https://jennybc.github.io/2014-05-12-ubc/r-setup.html for a detailed guide

For safety & collaboration

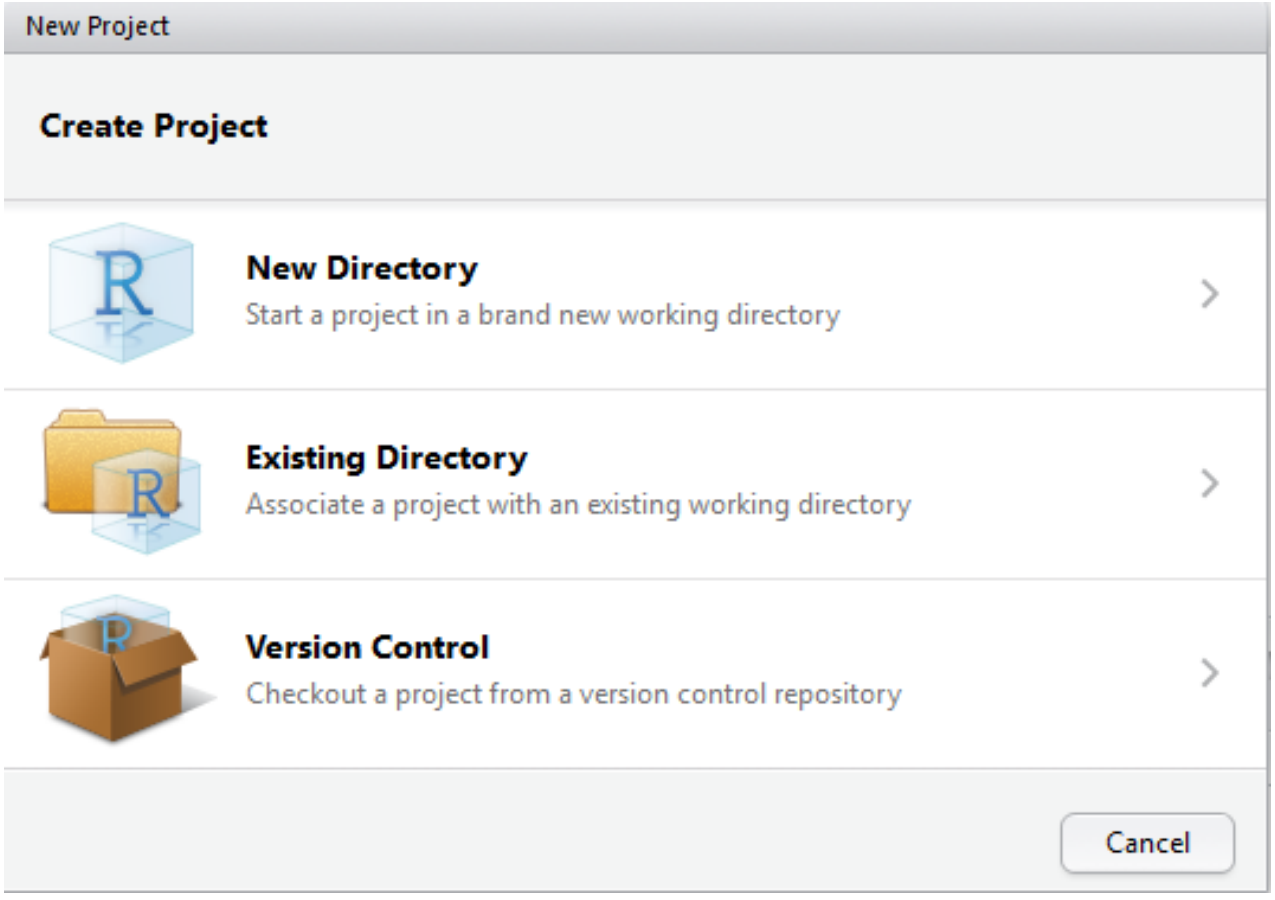
- RStudio projects
  - "RStudio projects make it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents."
  - Allows for return to key states
- .Rproj files
  - Basically a text file with some parameters for start up



Projects

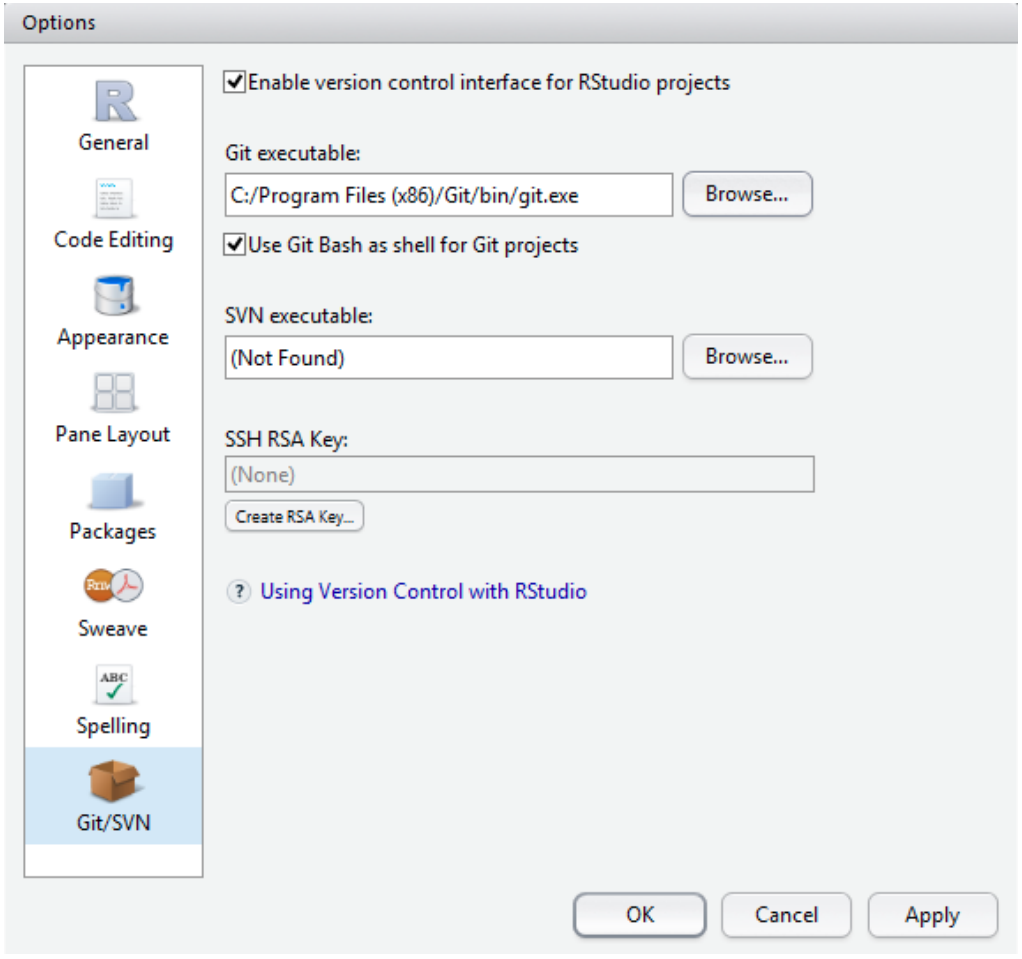
Create a new one for:

- a folder
- packages
- (and from) git repos:



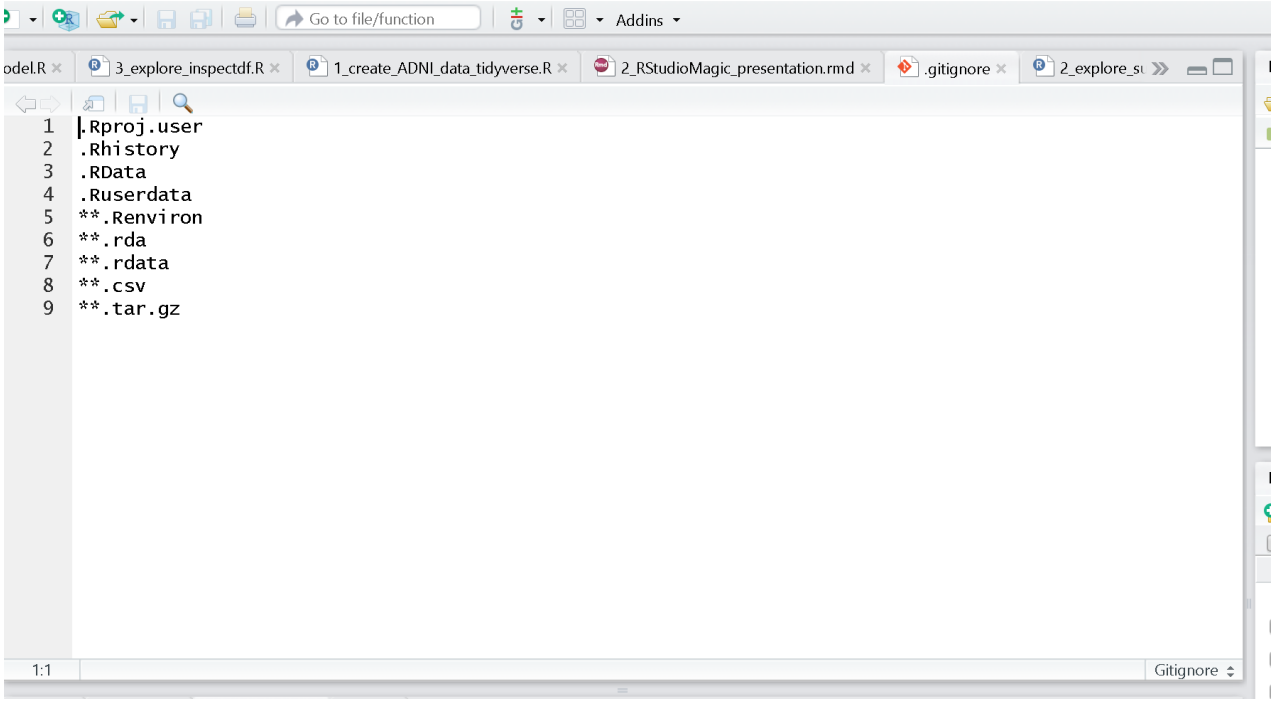
Git & Projects

- Git
- Download git and link executable within RStudio

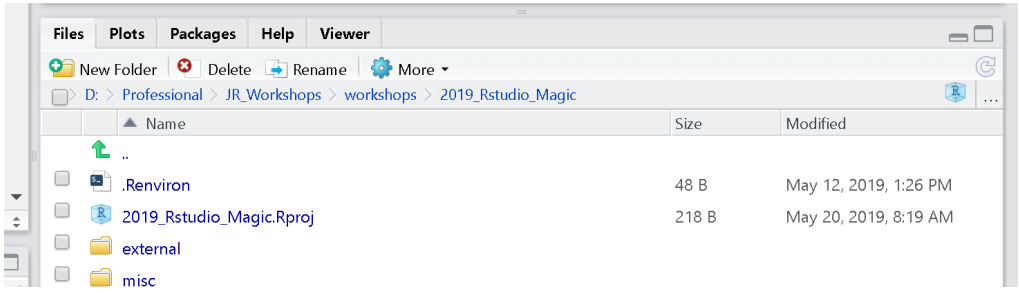


Format .gitignore

- File types to ignore via version control
- ** before each extentions will match directories anywhere in the repo

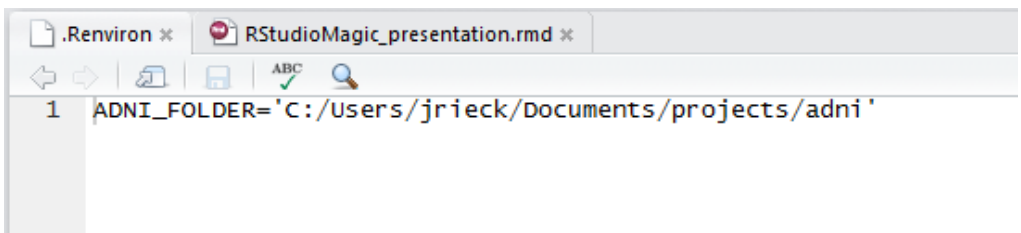


Environmental variables

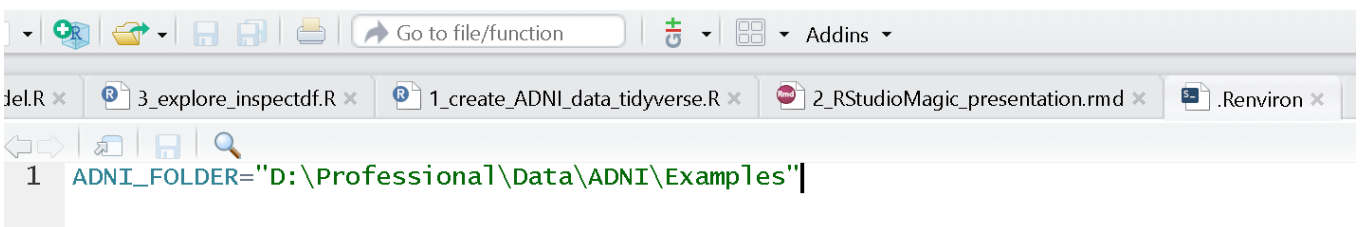


Format environmental variables

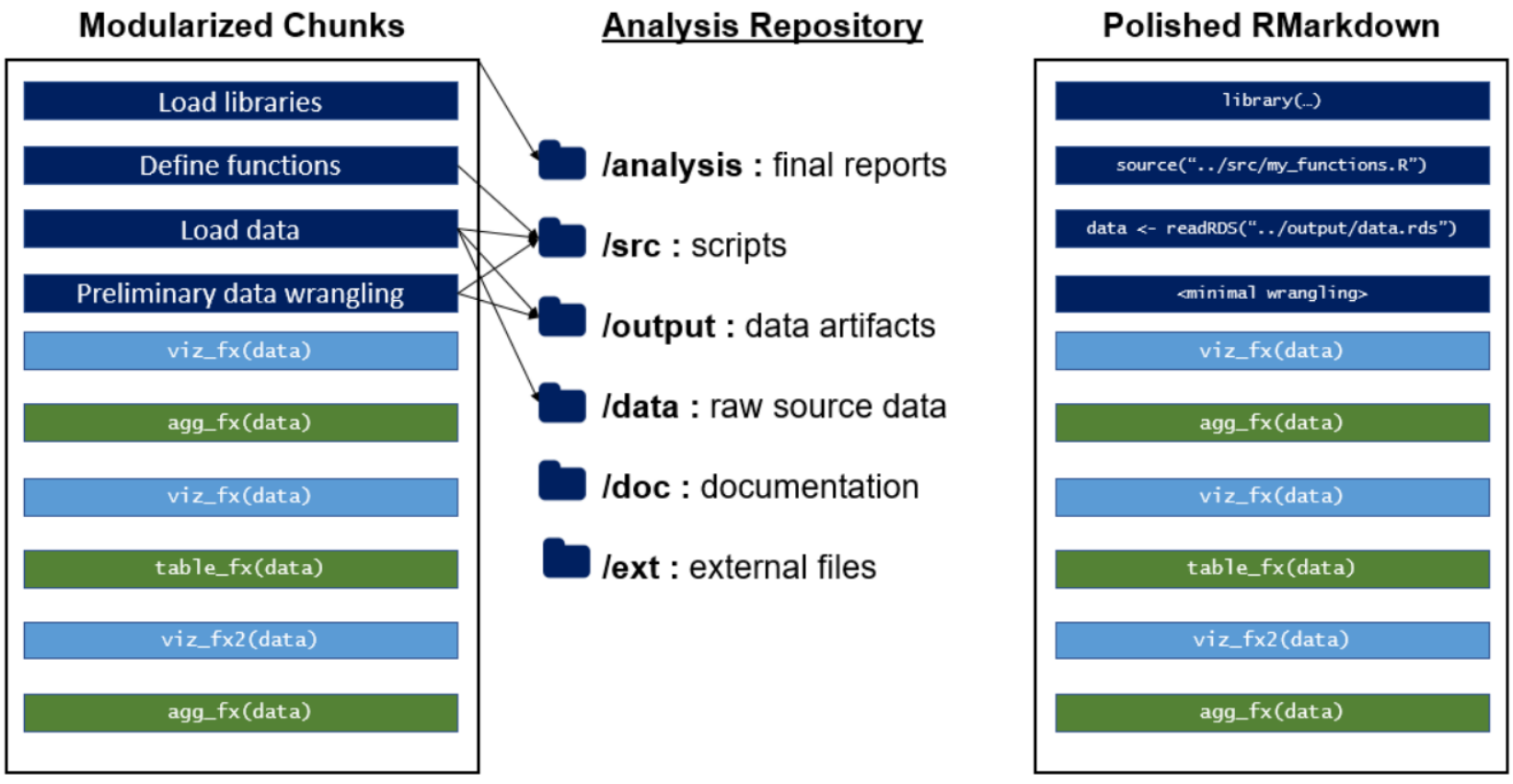
- Set environmental variables (ie, directory location of data) to make code generalizable across computers
  - Don't commit or share these
- In **your** project folder create a `.Renviron` file and define variables
  - Jenny's:



  - Derek's:



Organize your project folders and markdown

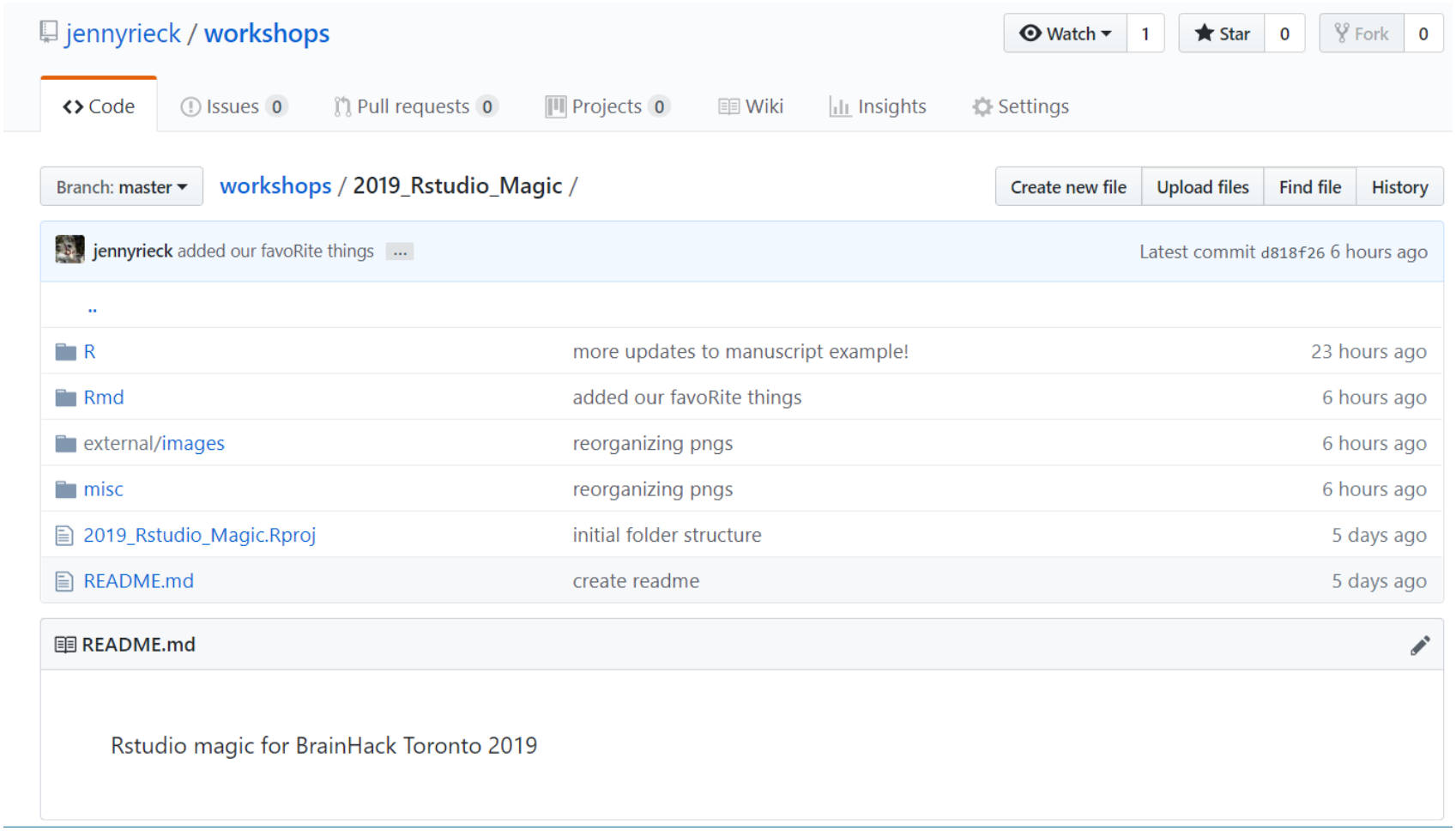


https://emilyriederer.netlify.com/post/rmarkdown-driven-development/

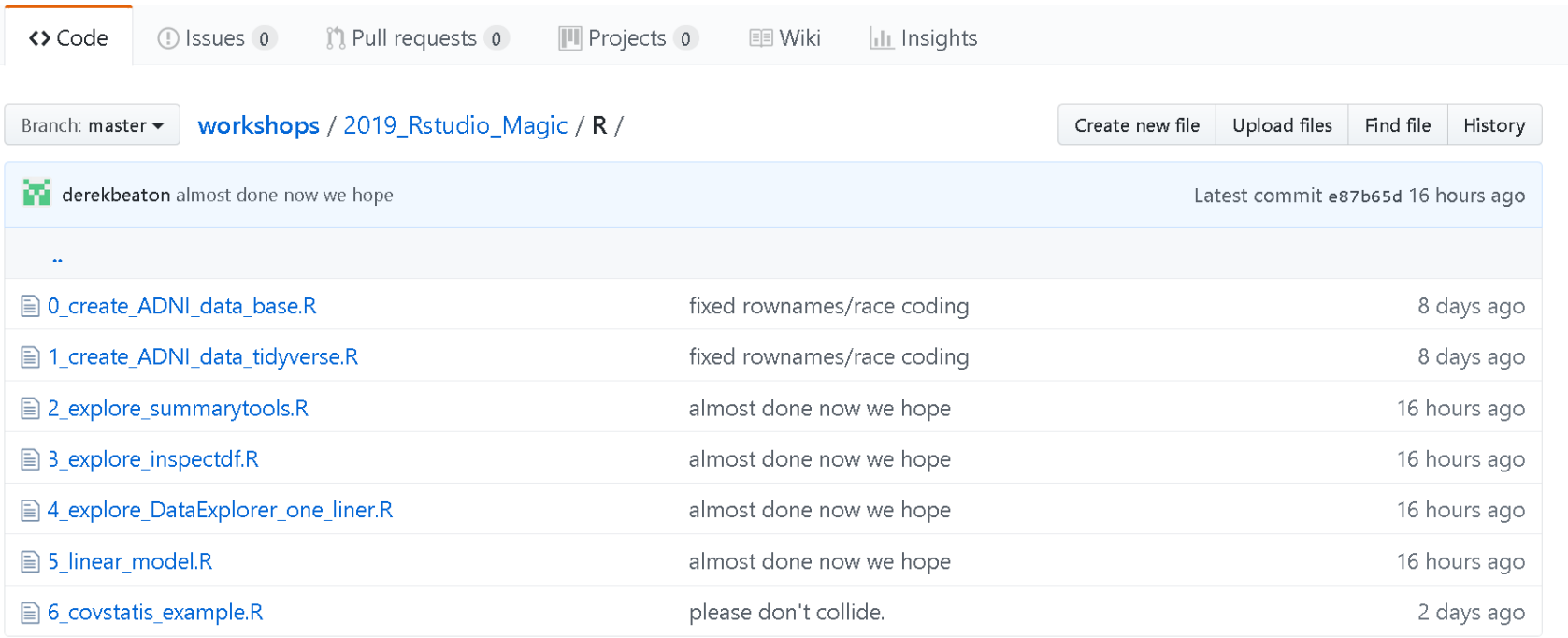
Organize your project folders and markdown

- What works for you?
- What works for your organization or team?
- Maximize utility, minimize complexity

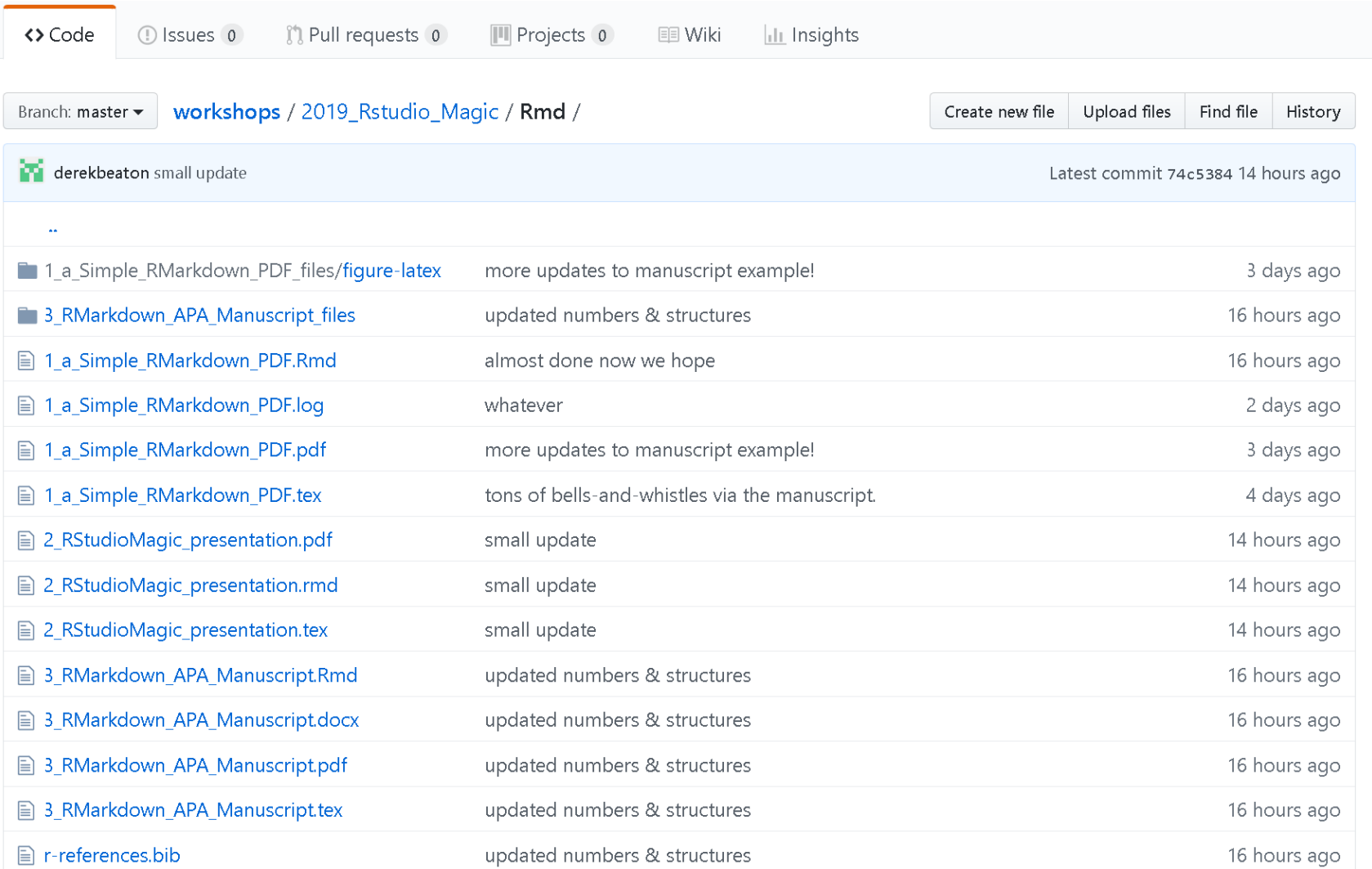
This works for us



This works for us



This works for us



Get the packages you need

e.g.,

```
#to install from CRAN
install.pacakges('devtools', depenedencies = TRUE)

#to install from a git  (requires the devtools package)
devtools::install_github(Gibbsdavidl/CatterPlots)

#to install from a file
install.packages('/mypath/to/package/ADNIMERGE.tar.gz',
                 type='source', repos=NULL)
```

(hint: you'll need more than this!)

Part 3: R

Let's take a closer look at R

- The Alzheimer's Disease Neuroimaging Initiative (ADNI).
- We use the `ADNIMERGE` R package
  - Lots and lots of data prepared *for you*
- We use this for all of our examples

The R examples

- Making & prepping data
  - `0_create_ADNI_data_base.R`
  - `1_create_ADNI_data_tidyverse.R`
- Exploring data
  - `2_explore_summarytools.R`
  - `3_explore_inspectdf.R`
  - `4_explore_DataExplorer_one_liner.R`
  - With a small rant from Derek
- Some stats: Linear models
  - `5_linear_model.R`
- Experimental packages (covSTATIS via Github)
  - `6_covstatis_example.R`

Derek's rant break

- DataExplorer committed a crime.
  - "coding categorical variables with the indicator matrix of dummy variables and considering them as Gaussian, for instance, is almost a crime."
- See Derek's PCA & multiple correspondence analysis workshop
  - Same data as this workshop
  - All in R & RMarkdown

Part 4: RMarkdown & more

RMarkdown

- Yuhui Xie:
  - https://bookdown.org/yihui/rmarkdown/
- What it is & why to use it
- Fancy helpers:
  - `kable` & `kableExtra`
  - `grid` & `gridExtra`
- Deviations for:
  - `LaTeX`
  - `Python`
- Tying it all together through here

RMarkdown Don'ts

- Don't hardcode values or absolute file paths
  - see `here::here()`
  - Use projects (`.Rproj`)
- Don't do complicated or expensive stuff
  - Database queries
  - Resampling
- avoid `eval=FALSE`
  - except to help illustrate code...
- Reduce repeated code
  - One time: Script
  - Two times: Function
  - Three times: Package

RMarkdown dos & don'ts

- Suggestions & More:
  - https://emilyriederer.netlify.com/post/rmarkdown-driven-development/
- We're using this here

Let's take a look

- `1_a_Simple_RMarkdown_PDF.Rmd`
- `2_RStudioMagic_presentation.rmd`
  - This presentation!
- `3_RMarkdown_APA_Manuscript.Rmd`

Part 5: “Advanced” topics

Some advanced/other things we're not covering

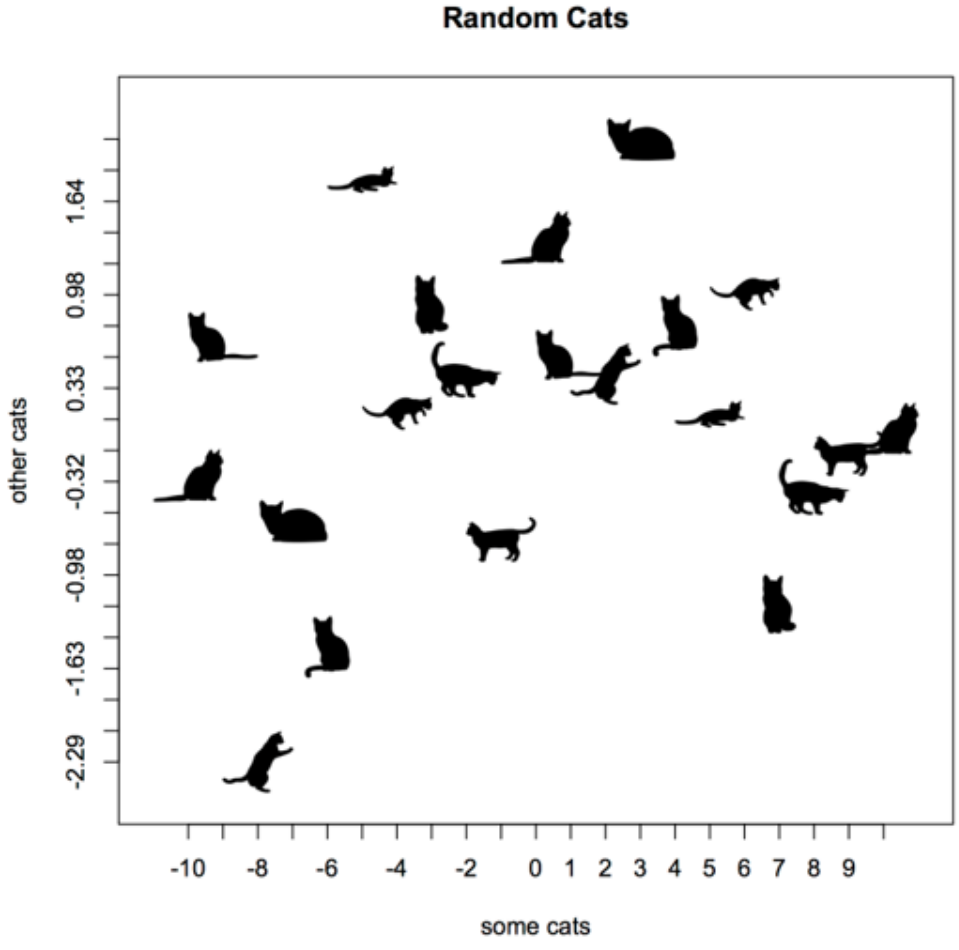
- package development
- Shiny
- SQL
- C/C++
- R2D3 (JavaScript)

A few of our favorite things

- Fun R do-dads

CatterPlot for feline based graphics:

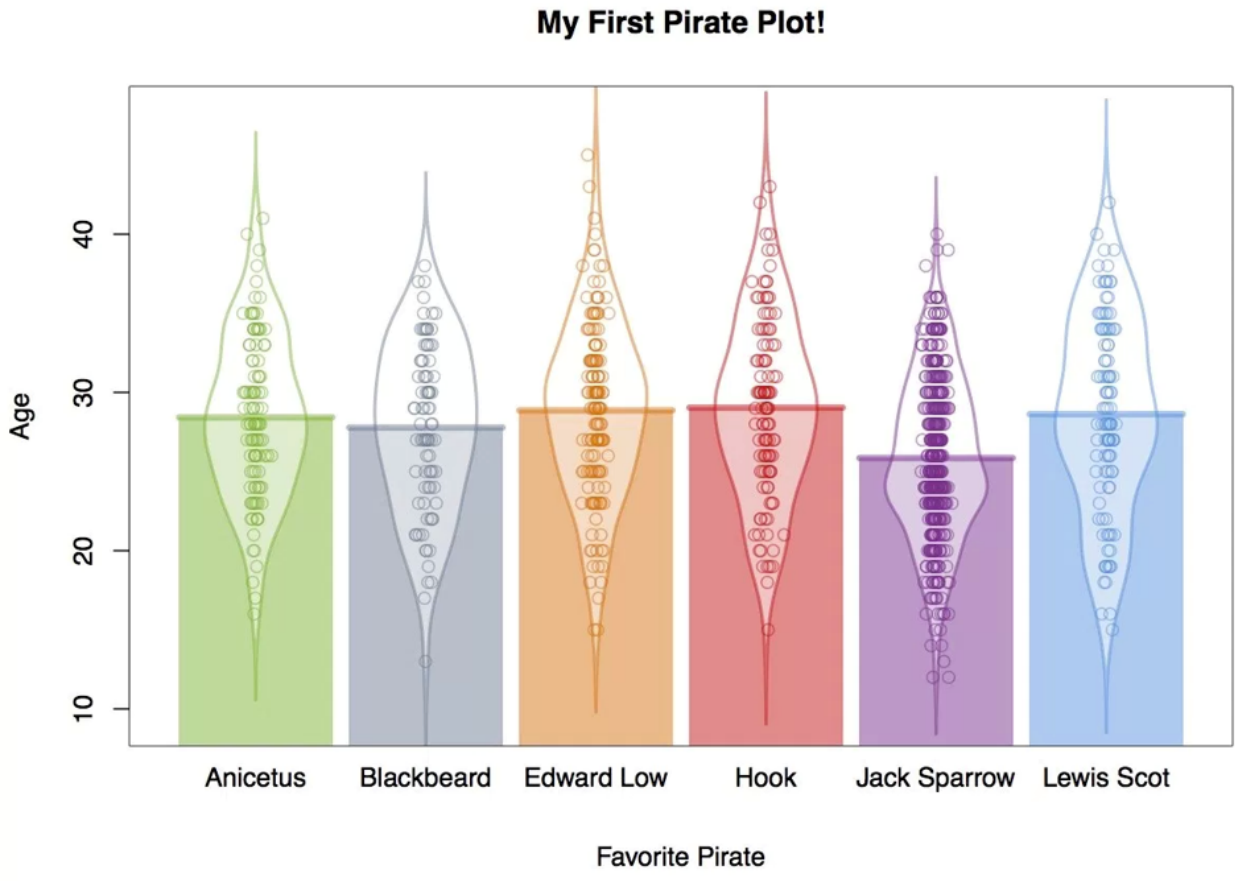
```
devtools::install_github(Gibbsdavidl/CatterPlots)
```



https://github.com/Gibbsdavidl/CatterPlots

What's a pirate's favorite programming language?

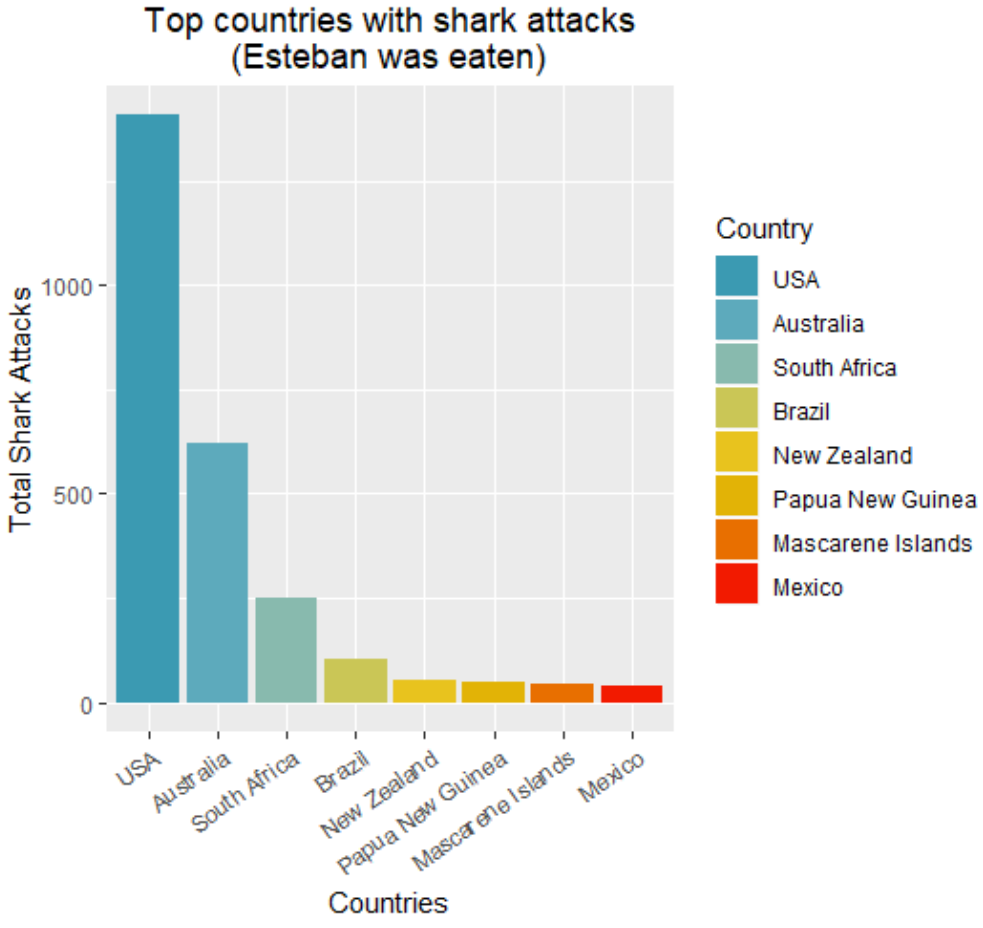
```
install.packages('yarrr')
```



https://cran.r-project.org/web/packages/yarrr/vignettes/pirateplot.html

Color palettes to fit your mood

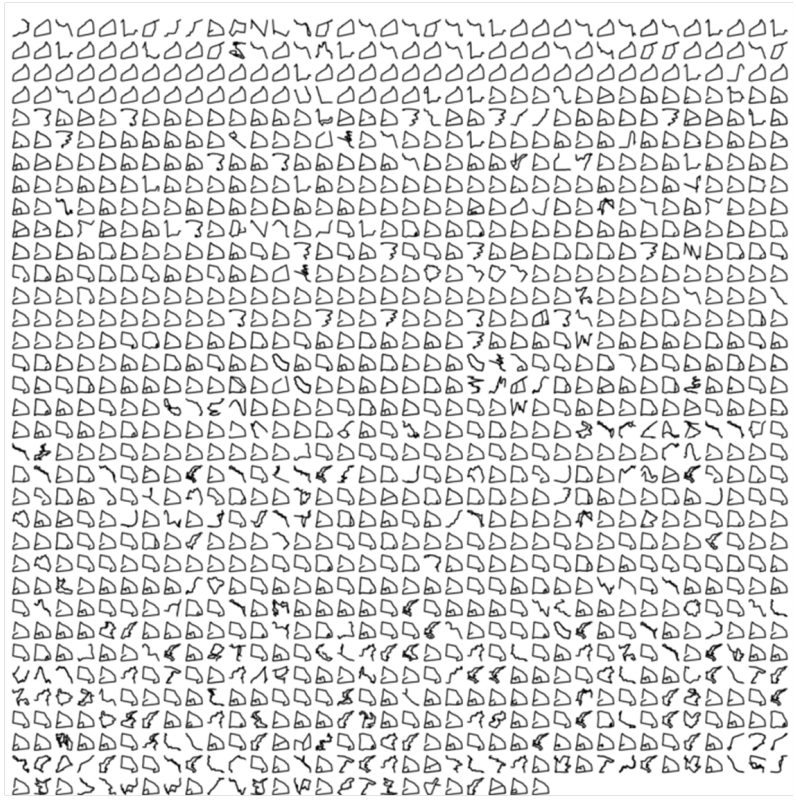
```
devtools::install_github(karthik/wesanderson)
```



https://github.com/karthik/wesanderson

Mapping your Strava routes

```
devtools::install_github(marcusvolz/strava)
```

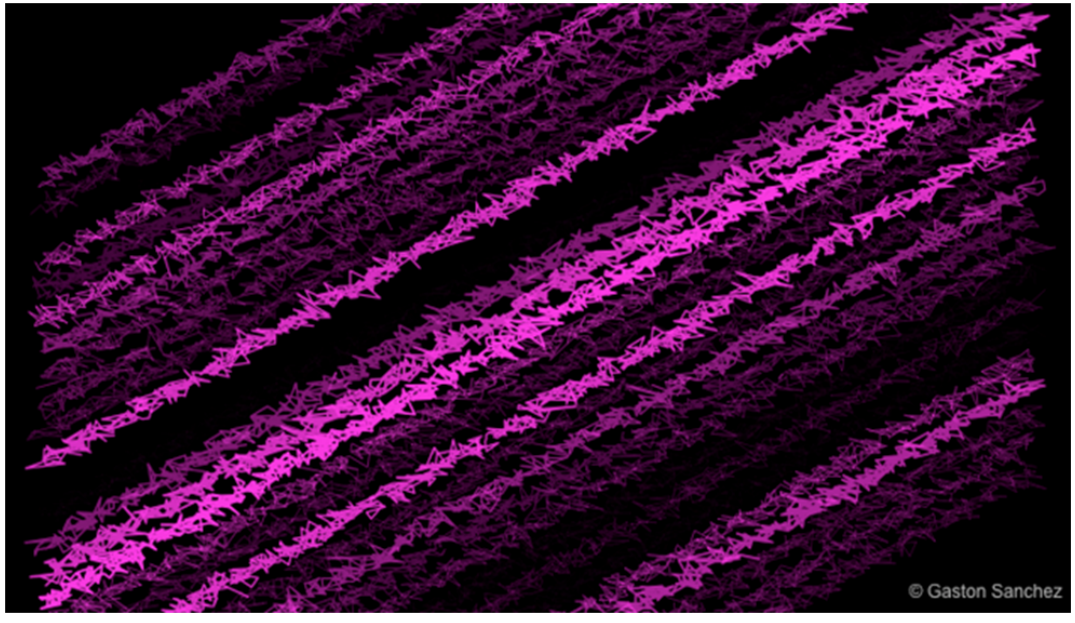


https://www.r-bloggers.com/strava-rides-map-in-r/

ALSO https://marcusvolz.com/?p=4068

Make aRt!

- R Graph Gallery
  - http://www.r-graph-gallery.com/
- Rtist: Gaston Sanchez
  - http://gastonsanchez.com/Rtist/




```
# ============================================================================
# Pink Barbs
# ============================================================================
# generate pairs of x-y values
x <- seq(1, 100, length = 1000)
y <- x + rnorm(1000)


png("pink_barbs.png", width = 700, height = 400)
# set graphical paramters
op <- par(bg = "black", mar = rep(0, 4))
# plot
plot(x, y, type = 'n')
for (i in seq(-80, 70, by = 5))
{
  lines(x + rnorm(1000), x + i + rnorm(1000, 2), pch = 19,
        col = hsv(0.85, 1, 1, runif(1000)),
        lwd = sample(seq(0.3, 2, length = 20), 1))
}
# signature
legend("bottomright", legend = "© Gaston Sanchez", bty = "n",
       text.col = "gray70")
# reset par
par(op)
dev.off()
```

Wouldn't be a brain hack without brains

R & Brain Imaging

- Slow to take hold
  - Getting there
- Neuroimaging: one of the last fields in R
  - Virtually every other field has an R community
  - Bioinformatics especially (see Bioconductor)

Reading & manipulating

- The neuroimaging ecosystem is growing in R
  - `neuroim`
  - `ANTsR`
  - `oro.nifti` - been around a while
  - `RMINC`

More brains

- MRI in Shiny
- brainGraph
- ggSeg

Neuroconductor

- Conceptually based on Bioconductor
- https://neuroconductor.org/