IBiS Computational Biology Bootcamp

September 12: Cook 3118, 9 AM - 1 PM
September 13: Cook 3118, 9 AM - 1 PM

September 15: Annenberg G02, 9 AM - 1 PM

Unsolicited advice about computational work

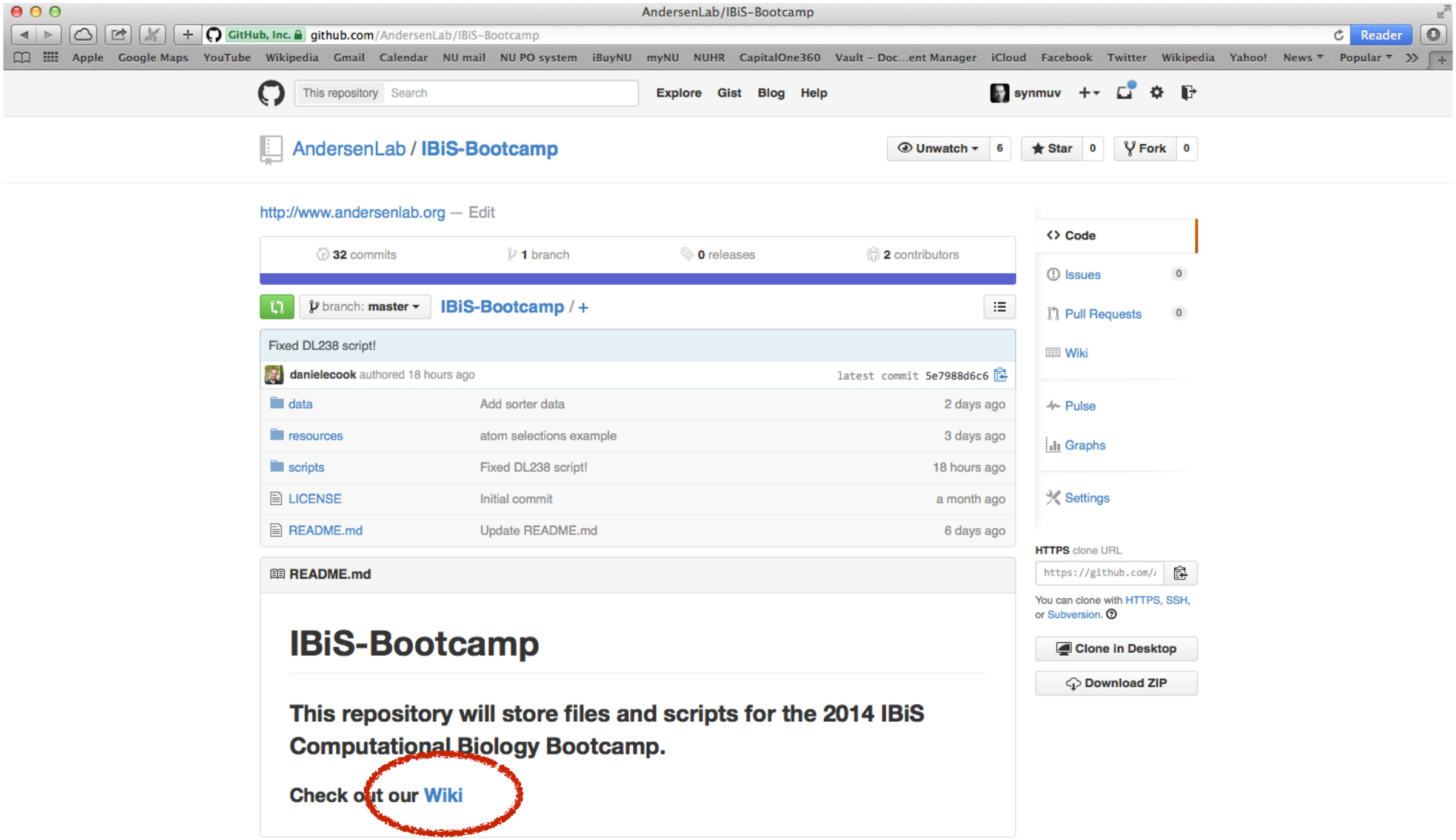
Don’t be afraid

It will be rewarding

It works

Stop using Excel

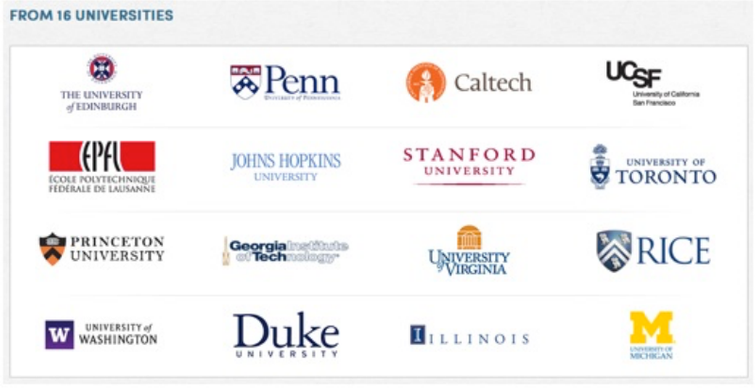
Use our wiki



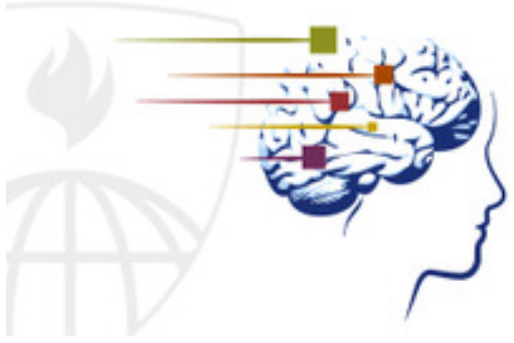
www.github.com/AndersenLab/IBiS-Bootcamp

The web is filled with free books, instructional tools, courses, etc.

Coursera



Data Science Specialization

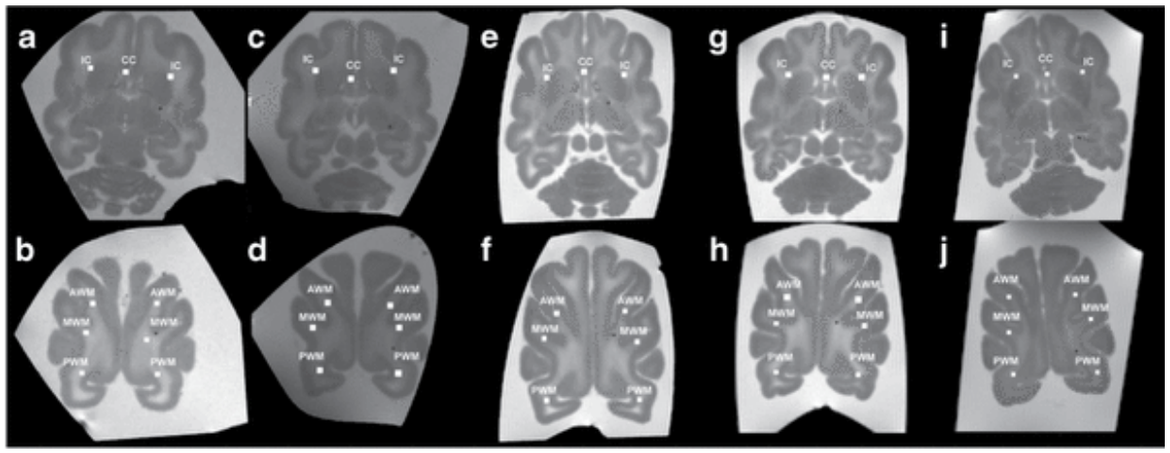


codecademy

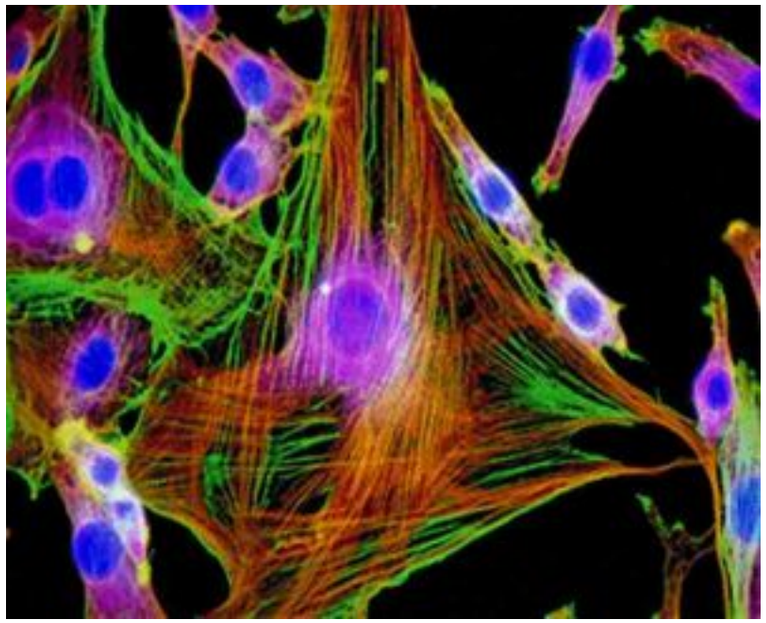
R SALIND

software carpentry

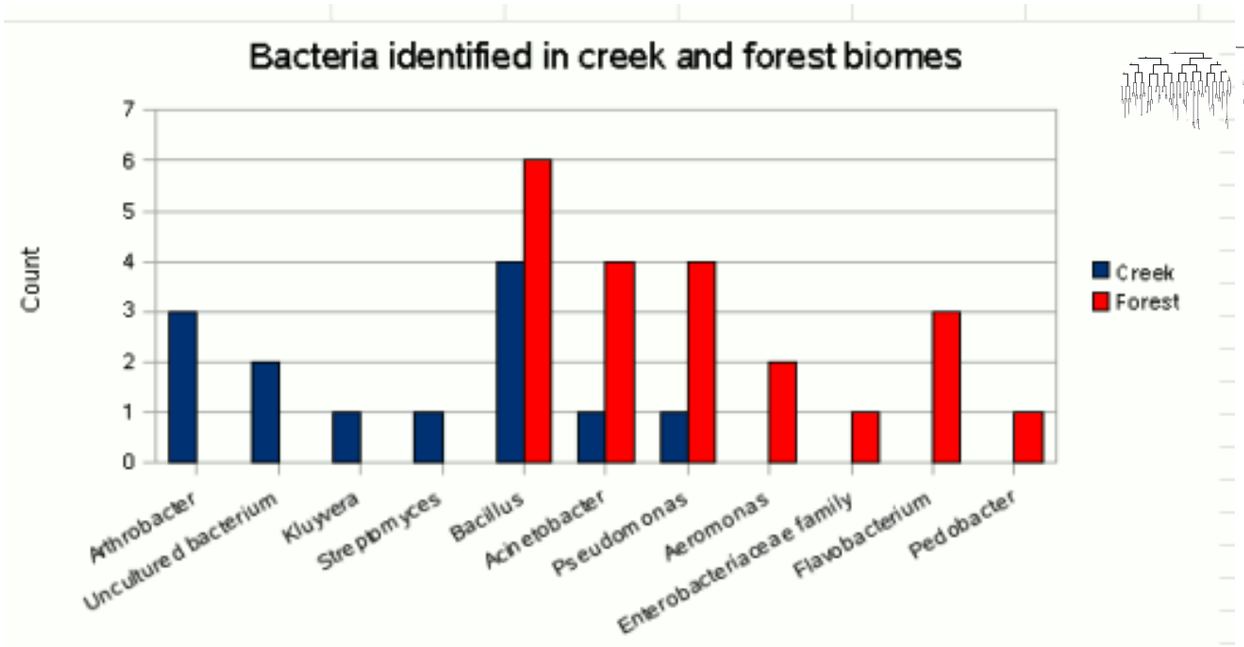
The days of descriptive biology are [almost] over



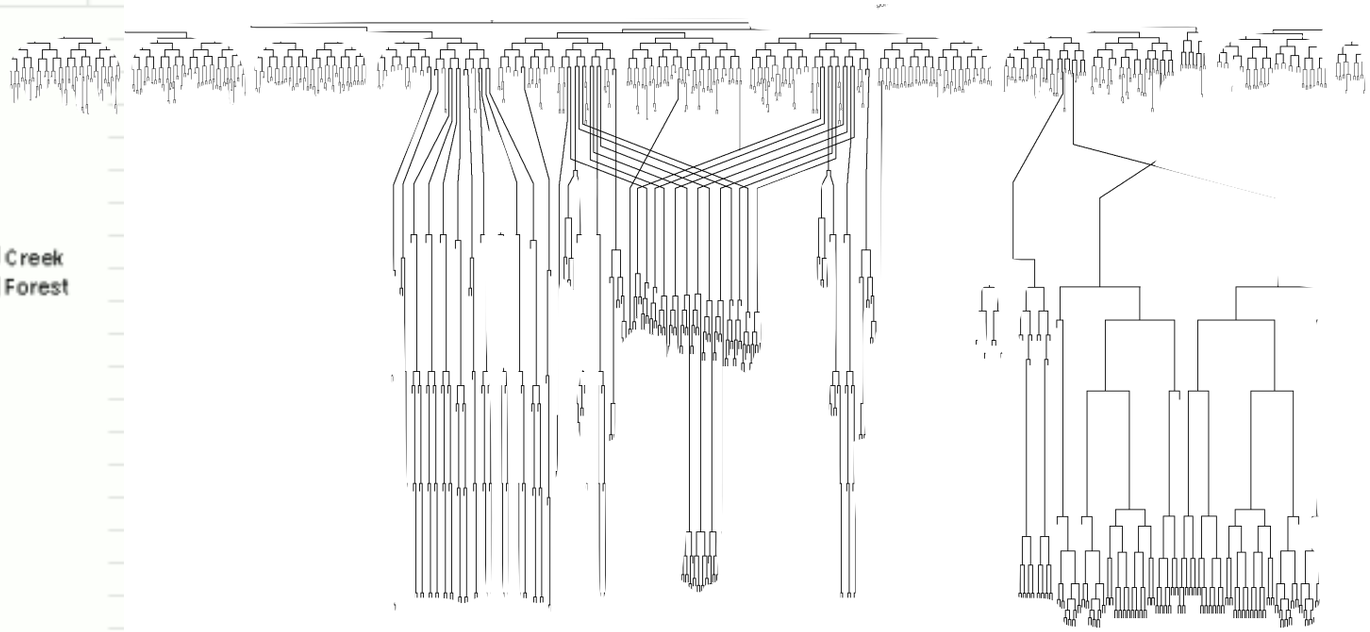
Reproducibility?



Sampling?

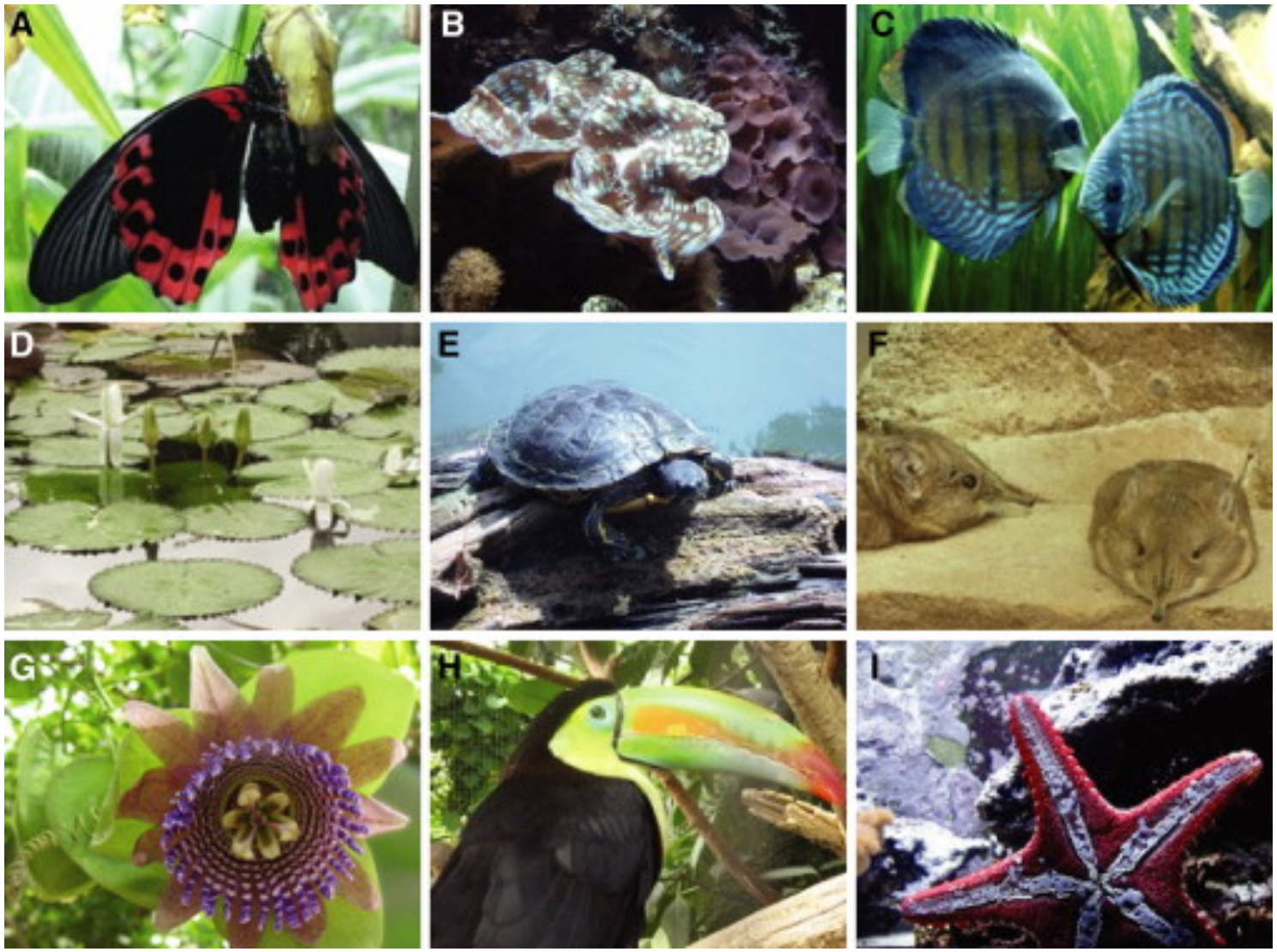


Error?



Automation?

NOW is the most exciting time to be a biologist



Any biologist can collect data.

Few biologists can process and analyze them.

Bootcamp outline

Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

Day #2: Data cleaning, manipulation, processing

Day #3: R and data plotting, five-minute presentations

Bootcamp outline

Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

- **Intro to reproducible research**
- **Git and github**
- **Directory structures**
- **Basic command line**
- **Markdown**
- **More command-tool utilities**
- **R and Studio**
- **Data input**

Working with biological data requires the ability to clean data.

The New York Times

For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights

By STEVE LOHR AUG. 17, 2014

Monica Rogati, Jawbone's vice president for data science, with Brian Wilt, a senior data scientist.
Peter DaSilva for The New York Times

FORTUNE

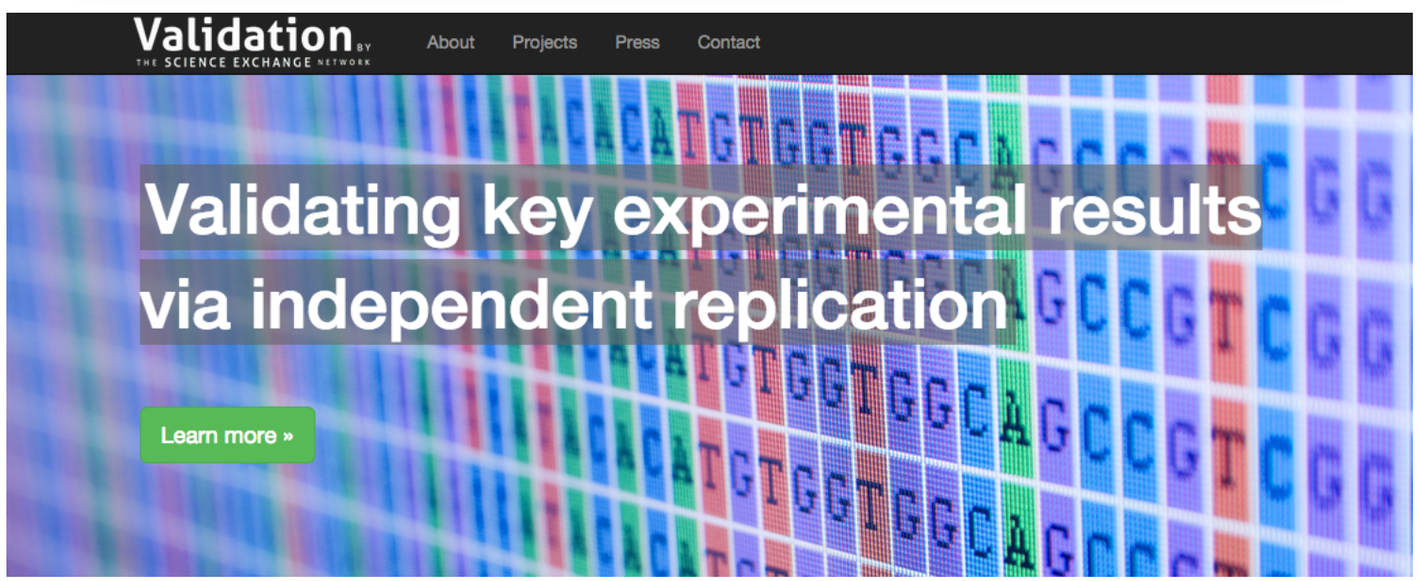
Big data's dirty problem

by Verne Kopytoff @vkopytoff JUNE 30, 2014, 10:58 AM EDT

PHYS.ORG

Science is in a reproducibility crisis: How do we resolve it?

Sep 20, 2013 by Fiona Fidler & Ascelin Gordon, The Conversation



Reproducibility initiatives will invite increased scrutiny into data cleaning methods.

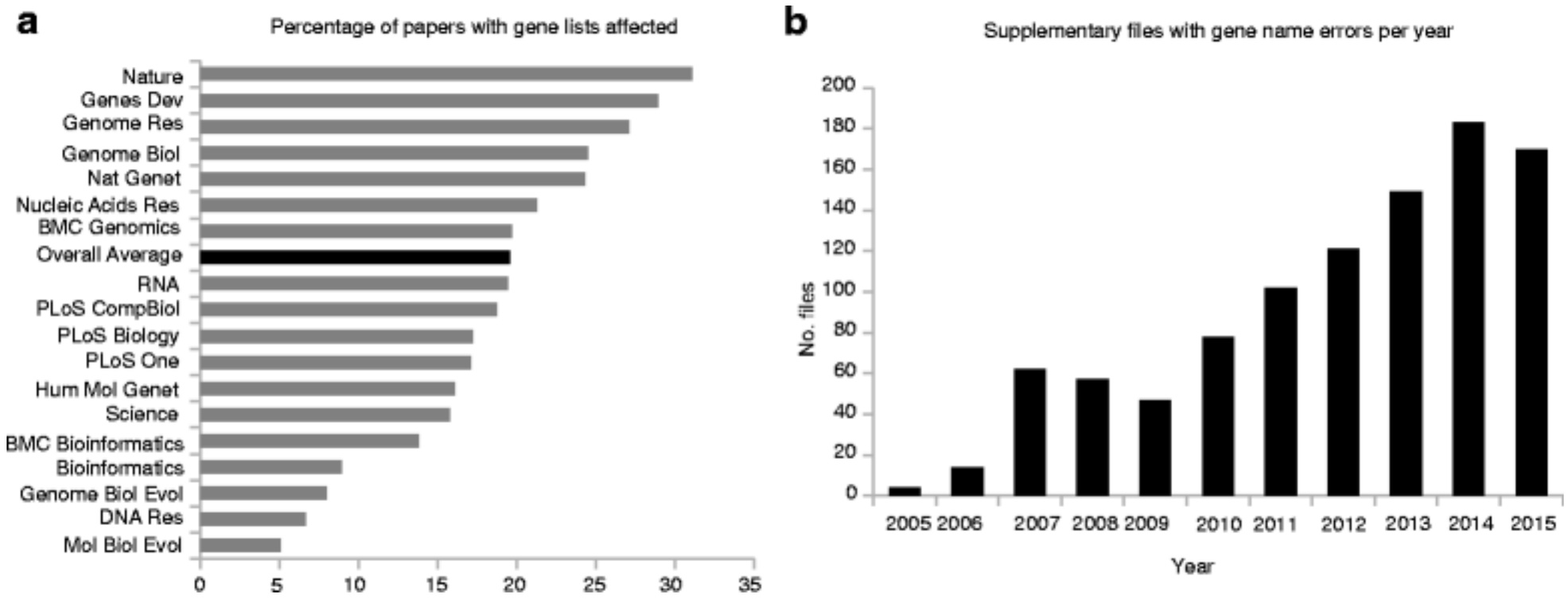
What is reproducible research?

The concept that scientific claims are published with data and code so that others can verify and build on those findings.

Please experience more at:

www.coursera.org Reproducible research course
GitHub.com Help and training pages

Microsoft Excel is destroying biology



Gene name errors are widespread in the scientific literature

Ziemann, Eren, and El-Osta. Genome Biology August 2016

Bootcamp outline

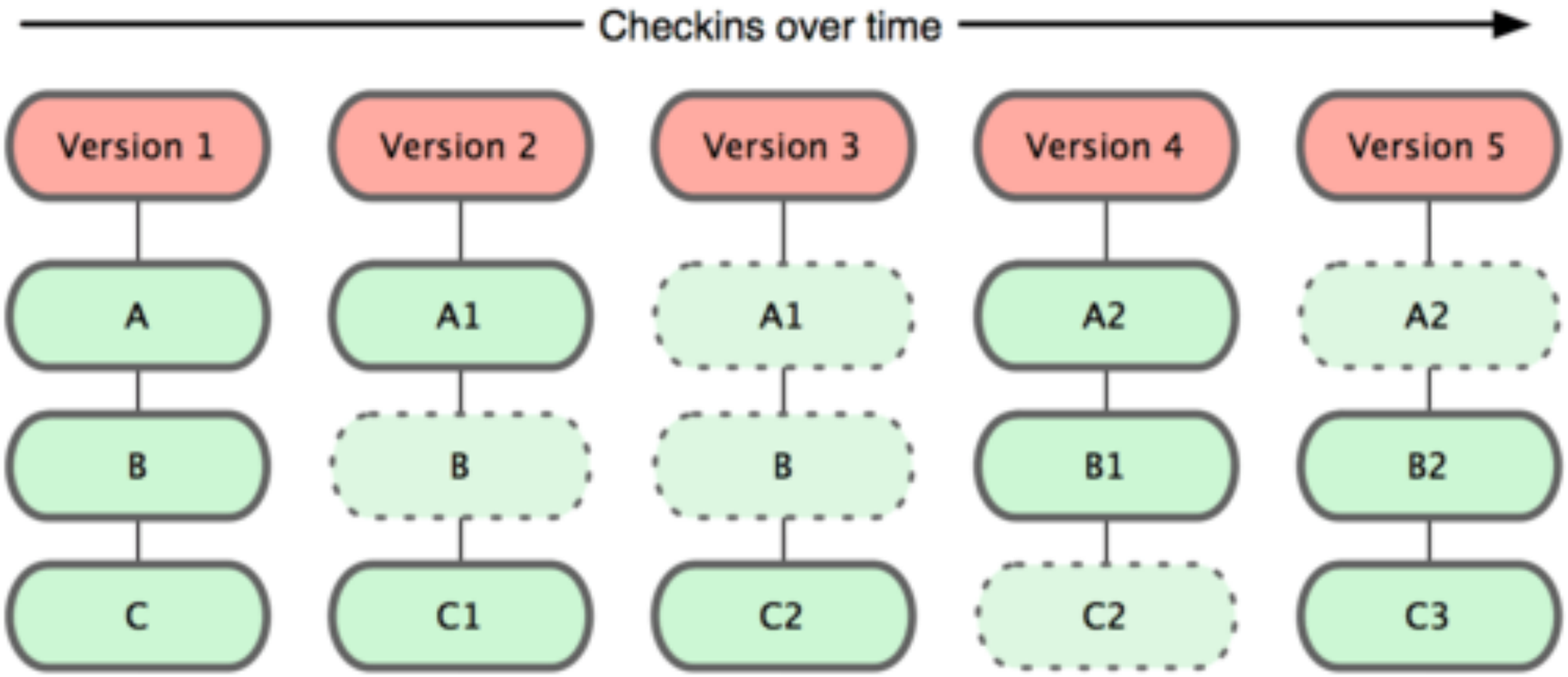
Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

- Intro to reproducible research
- **Git and github**
- Directory structures
- Basic command line
- Markdown
- More command-tool utilities
- R and Studio
- Data input

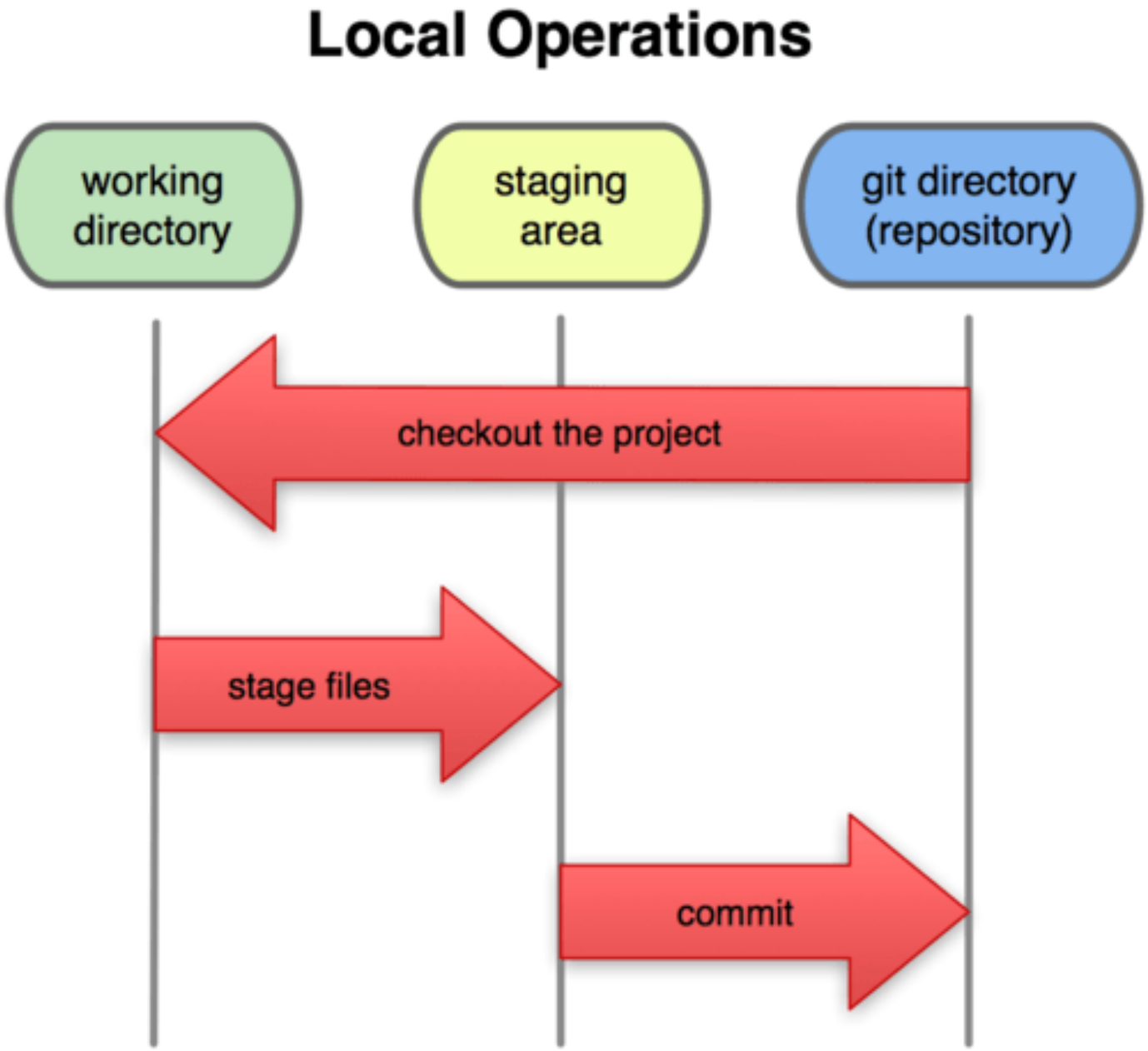
Notes vs `what you type`

is a version control system

git tracks changes and files over time



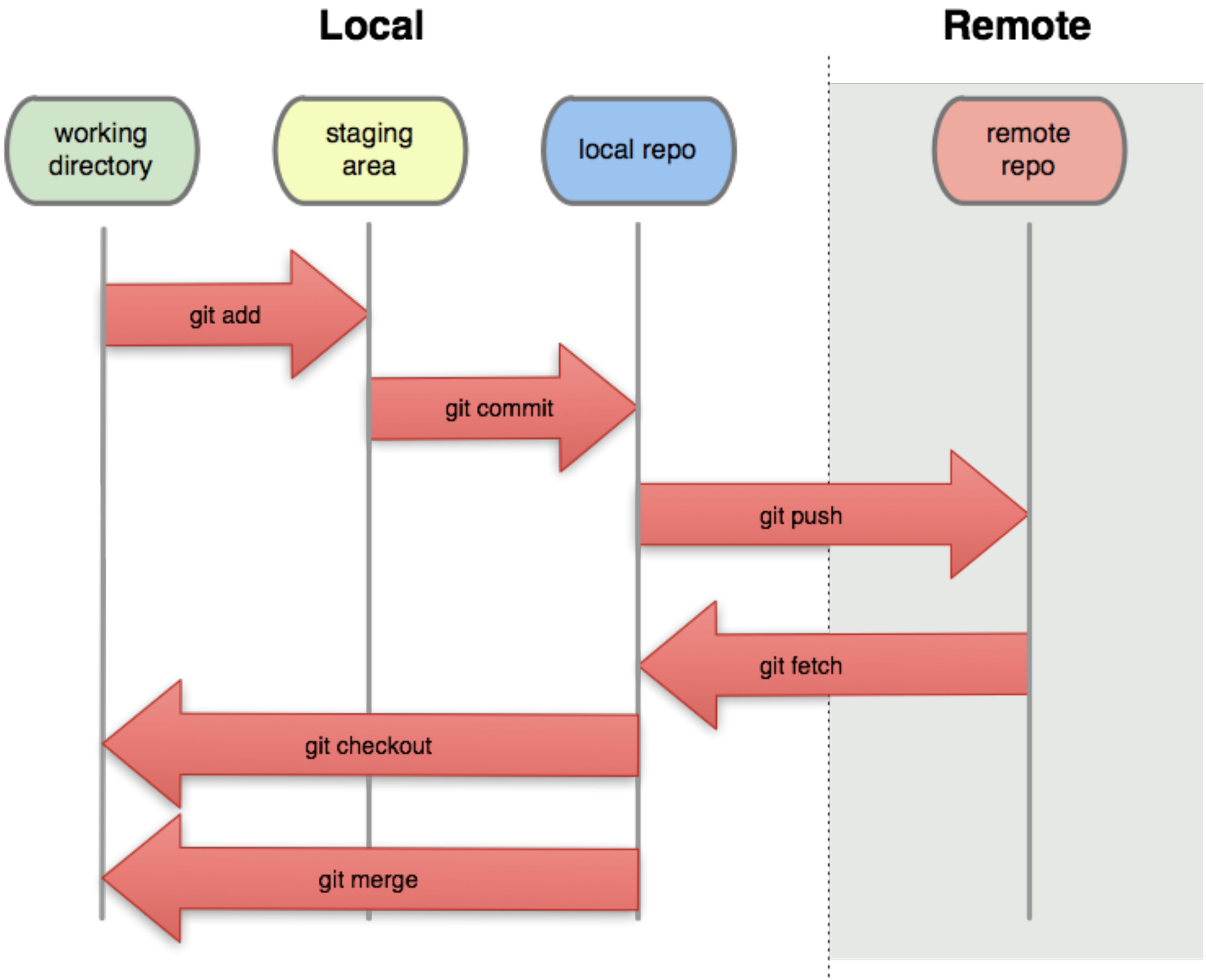
git tracks local changes



Github makes git super powerful

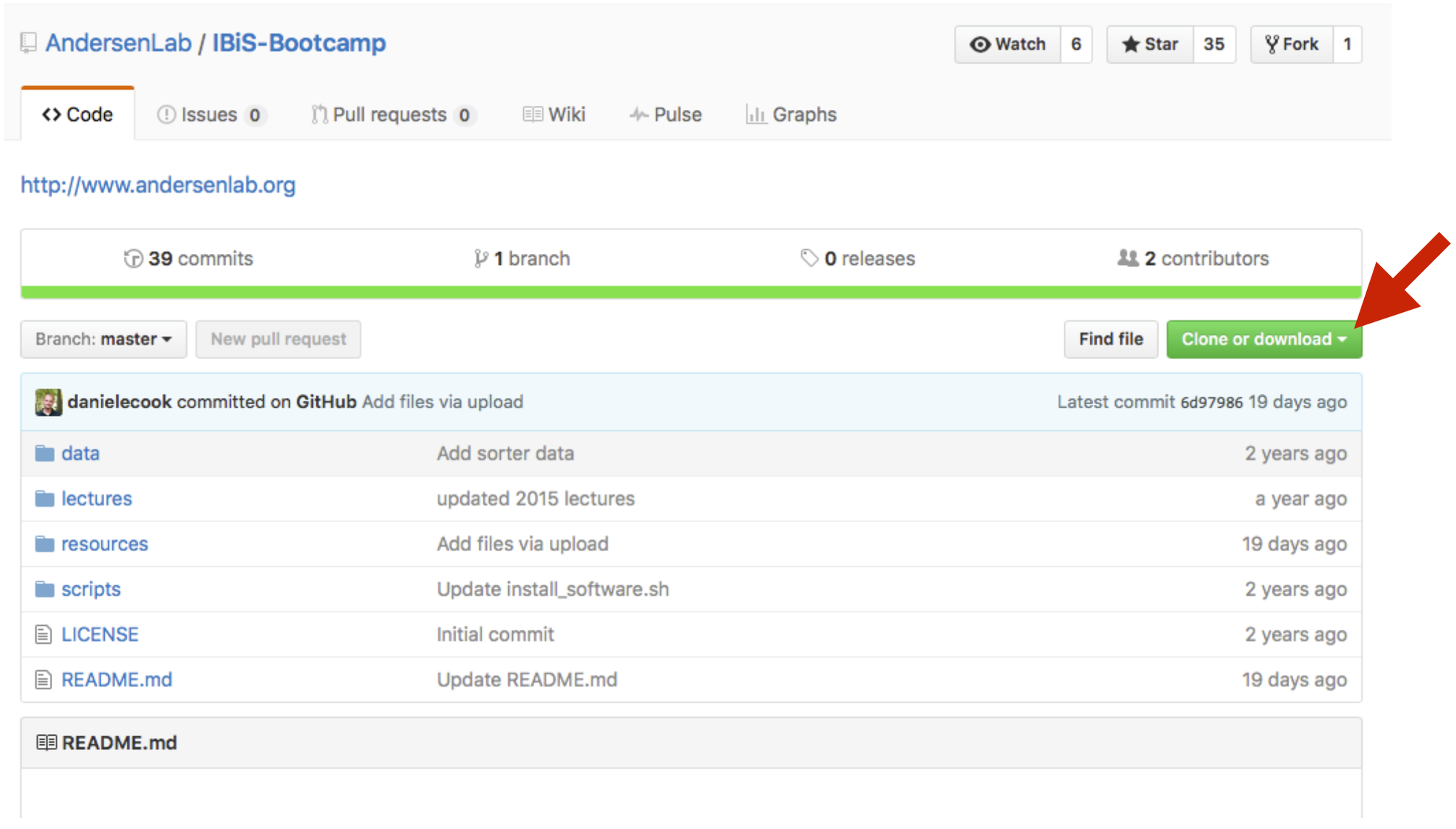
Check out: www.github.com/andersenlab

github takes git from local to the world

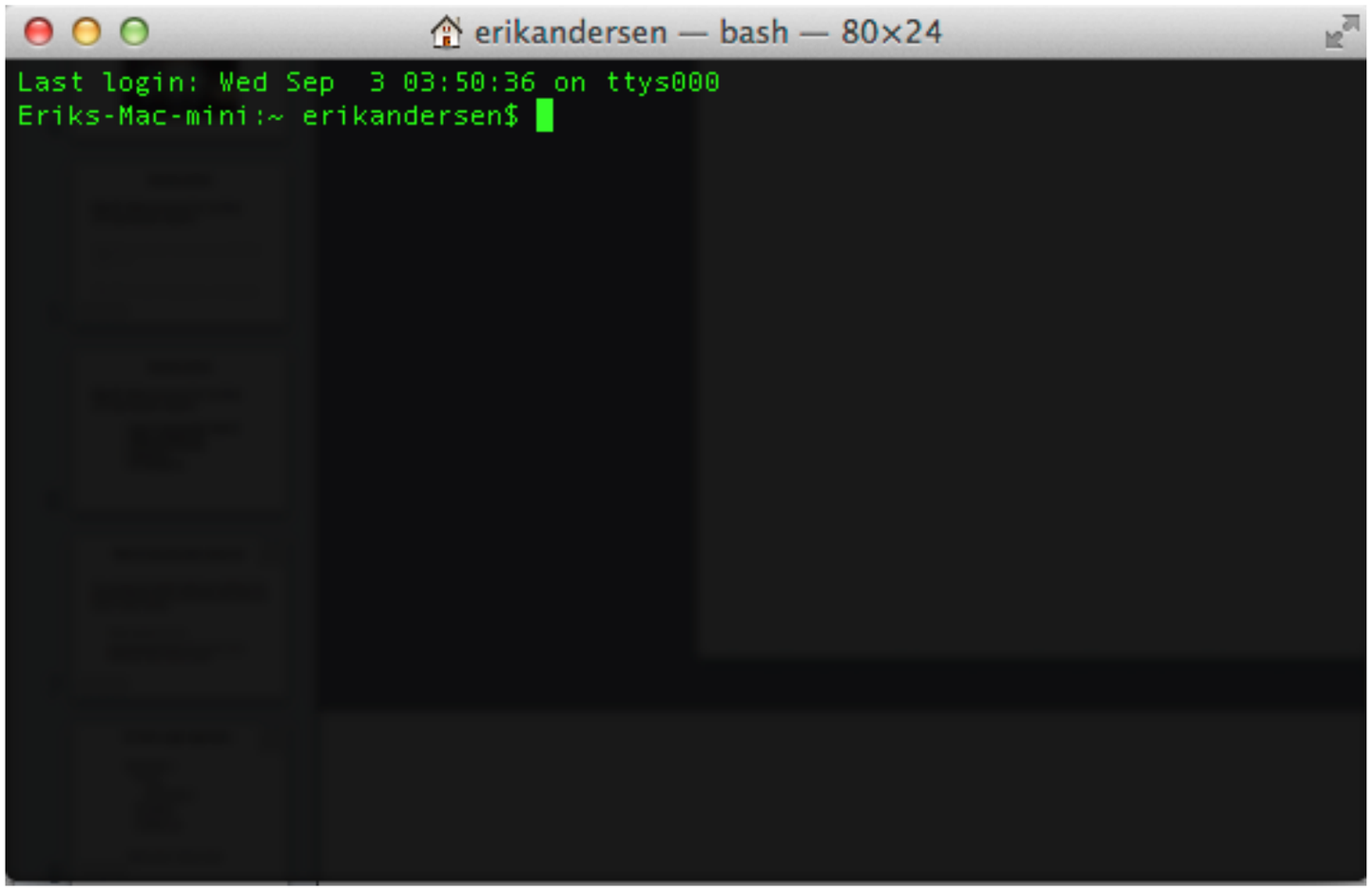


Let's clone the IBiS Bootcamp repository

Go to github.com/andersenlab/ibis-bootcamp



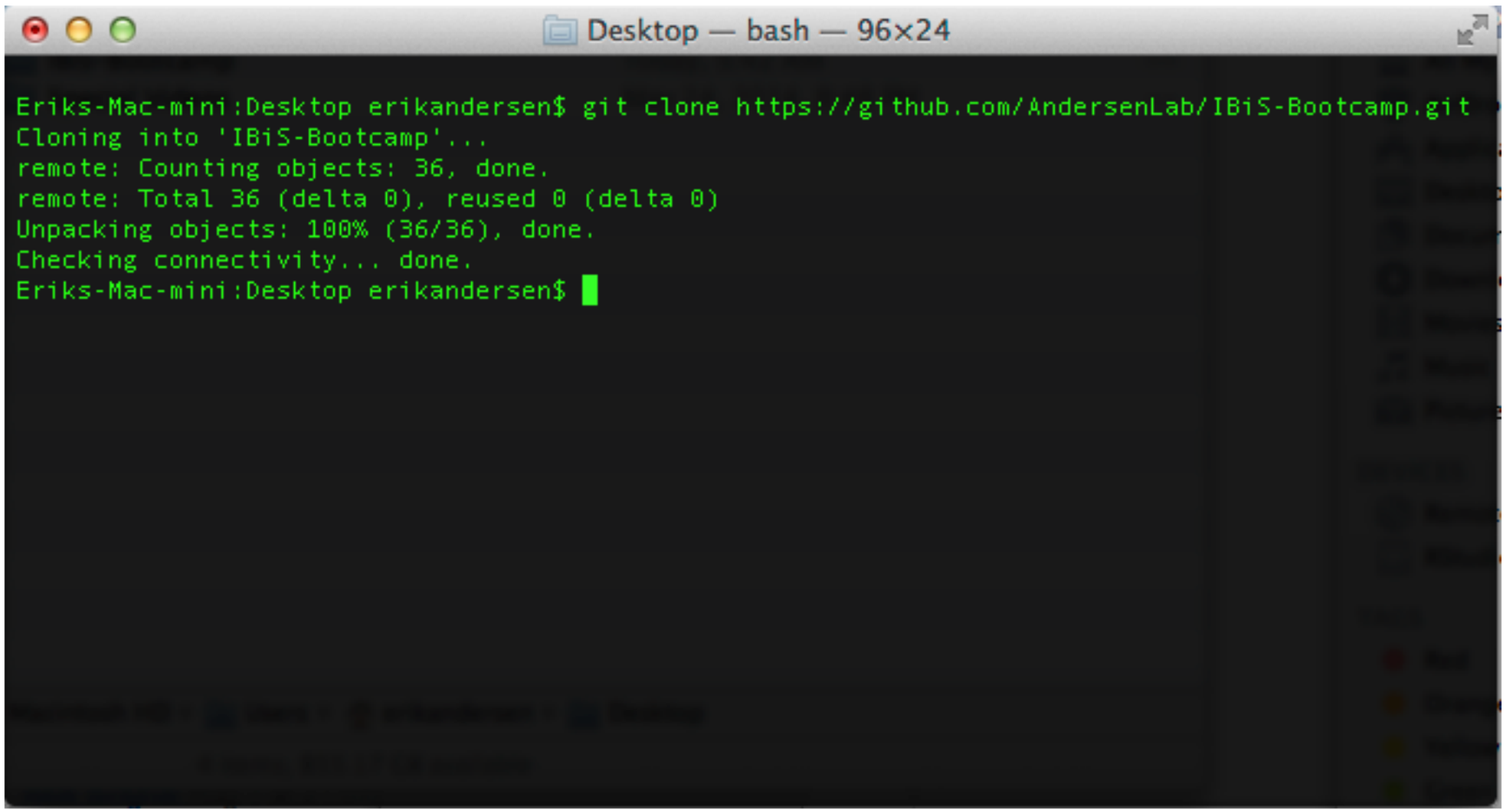
Command-line interface on a Mac



Open Terminal (in your dock)

Let’s clone the IBiS Bootcamp repository

`git clone` invokes git to clone a repository



1. Go to home directory. Type `cd ~/`
2. Type `git clone` and then paste directory after `clone`.

Git/Github Commands

Move files to staging area `git add [file/s]`

Check status of local repo `git status`

Commit files to local repo `git commit -m “short message”`

Push files to remote repo `git push origin master`

Further experiences:
http://git.io/vGoaJ

xkcd.com/1597

Bootcamp outline

Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

- **Intro to reproducible research**
- **Git and github**
- **Directory structures**
- **Basic command line**
- **Markdown**
- **More command-tool utilities**
- **R and Studio**
- **Data input**

Notes vs `what you type`

It’s time to get organized…

```
MyProject/
  data/
    raw/
    processed/
  scripts/
  results/
  readme.md
```

`MyProject/data/raw/`

Basic directory commands for UNIX

`man` = read the manual for a command

`ls` = list the contents of a directory

Try typing `man ls`

Try typing `man ls -a`

Further experiences:
http://freeengineer.org/learnUNIXin10minutes.html
http://www.ee.surrey.ac.uk/Teaching/Unix/
or just google: “unix command line tools”

Basic directory commands for UNIX

`man` = read the manual for a command

`ls` = list the contents of a directory

`pwd` = display the present working directory

`cd` = change directory

`mkdir` = make a directory

`rmdir` = remove a directory

Further experiences:
http://freeengineer.org/learnUNIXin10minutes.html
http://www.ee.surrey.ac.uk/Teaching/Unix/
or just google: “unix command line tools”

Keyboard shortcuts for UNIX

Press *tab* to complete a filename

Navigation shortcuts
ctrl + a : go to beginning of line
ctrl + e : go to end of line
alt + <- (->) : skip between delimiters
alt + delete : delete previous word
command + up (down) arrow: beginning (or end) of text file

Selection shortcuts
option + shift + <- (->): highlight text to next delimiter
shift + up (down) arrow : highlight previous (next) line
command + shift <- (->) : highlight to beginning (or end of line)
command + shift + up (down) arrow : highlight to beginning (or end of text)

Further experiences:
http://goo.gl/ukS77X

Let's each make the directory structure using the command line.

```
MyProject/
   data/
      raw/
      processed/
   scripts/
   results/
   mistakes_happen/
```

`man` = read the manual for a command

`ls` = list the contents of a directory

`pwd` = display the present working directory

`cd` = change directory

`mkdir` = make a directory

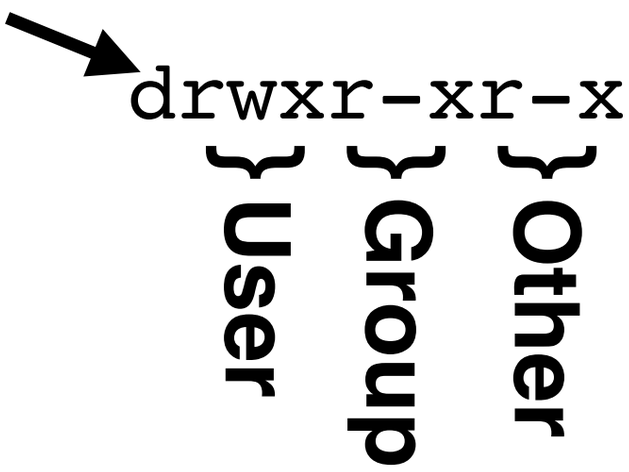
`rmdir` = remove a directory

`cd ..` allows you to go back one directory in tree

Permissions (modes) matter.

Go to your `data` directory, type `ls -la`

Filetype (- , d, or c)



`r` = read

`w` = write

`x` = execute

`chmod` changes the mode

Symbolic Notation	Octal Notation	English
`----------`	0000	no permissions
`---x--x--x`	0111	execute
`--w--w--w-`	0222	write
`--wx-wx-wx`	0333	write & execute
`-r--r--r--`	0444	read
`-r-xr-xr-x`	0555	read & execute
`-rw-rw-rw-`	0666	read & write
`-rwxrwxrwx`	0777	read, write, & execute

Bootcamp outline

Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

- Intro to reproducible research
- Git and github
- Directory structures
- Basic command line
- **Markdown**
- More command-tool utilities
- R and Studio
- Data input

Notes vs `what you type`

What about that readme.md file?

md = markdown

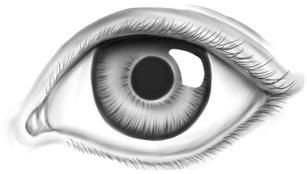
Markdown is the emerging standard for software documentation.

Easily written, read, and translated.

Portable to HTML, PDF, etc.

We need files that the computer and we can read.

Binary files
are read by computers



Markdown files

CSV or TSV files

Plain text files
are read by computers
and us

MS files
are read by
us

What about that readme.md file?

md = markdown

what you type

```
Heading
=======

Sub-heading
-----------

h3. Traditional html title

Paragraphs are separated
by a blank line.

Let 2 spaces at the end of a line to do a
line break

Text attributes *italic*,
**bold**, `monospace`.

A [link](http://example.com).
<<<   No space between ] and (  >>>

Shopping list:

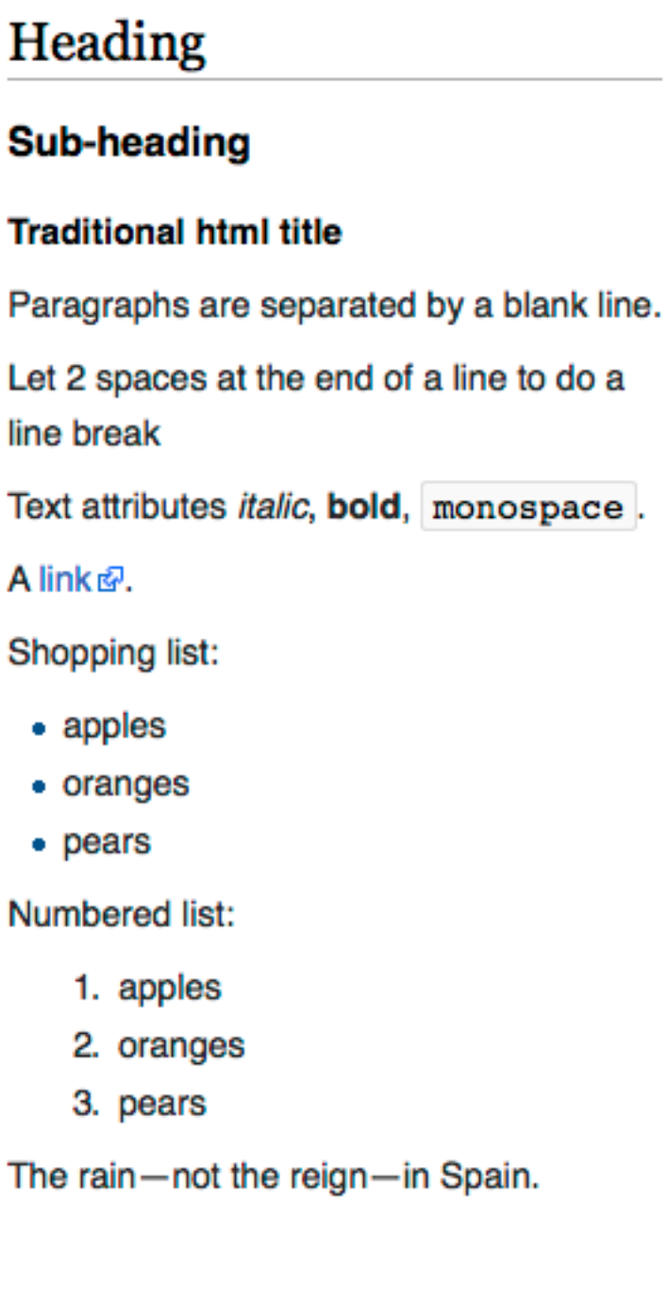
  * apples
  * oranges
  * pears

Numbered list:

  1. apples
  2. oranges
  3. pears

The rain---not the reign---in
Spain.
```

what is rendered



Heading

Sub-heading

Traditional html title

Paragraphs are separated by a blank line.

Let 2 spaces at the end of a line to do a line break

Text attributes *italic*, **bold**, `monospace`.

A link.

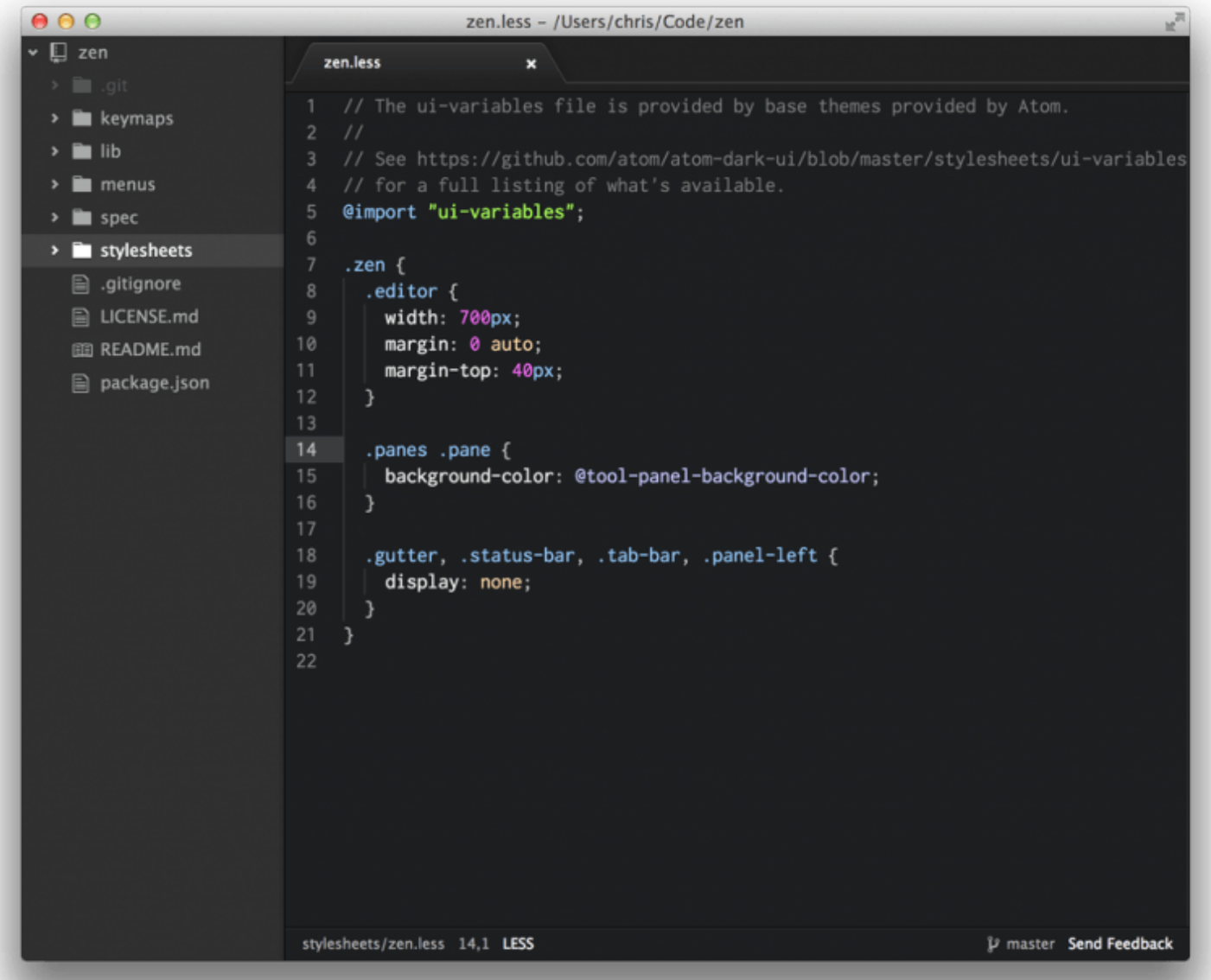
Shopping list:

- apples
- oranges
- pears

Numbered list:

1. apples
2. oranges
3. pears

The rain—not the reign—in Spain.

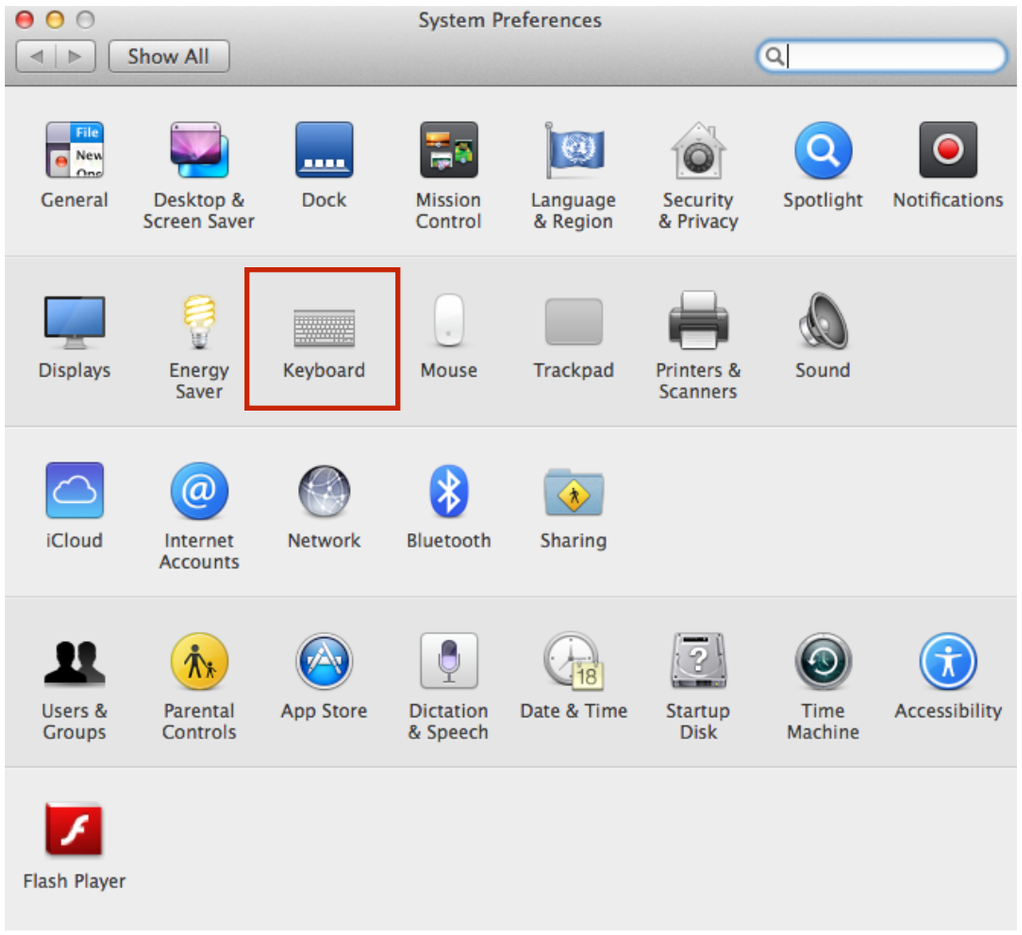


A plain text editor different from Microsoft Word

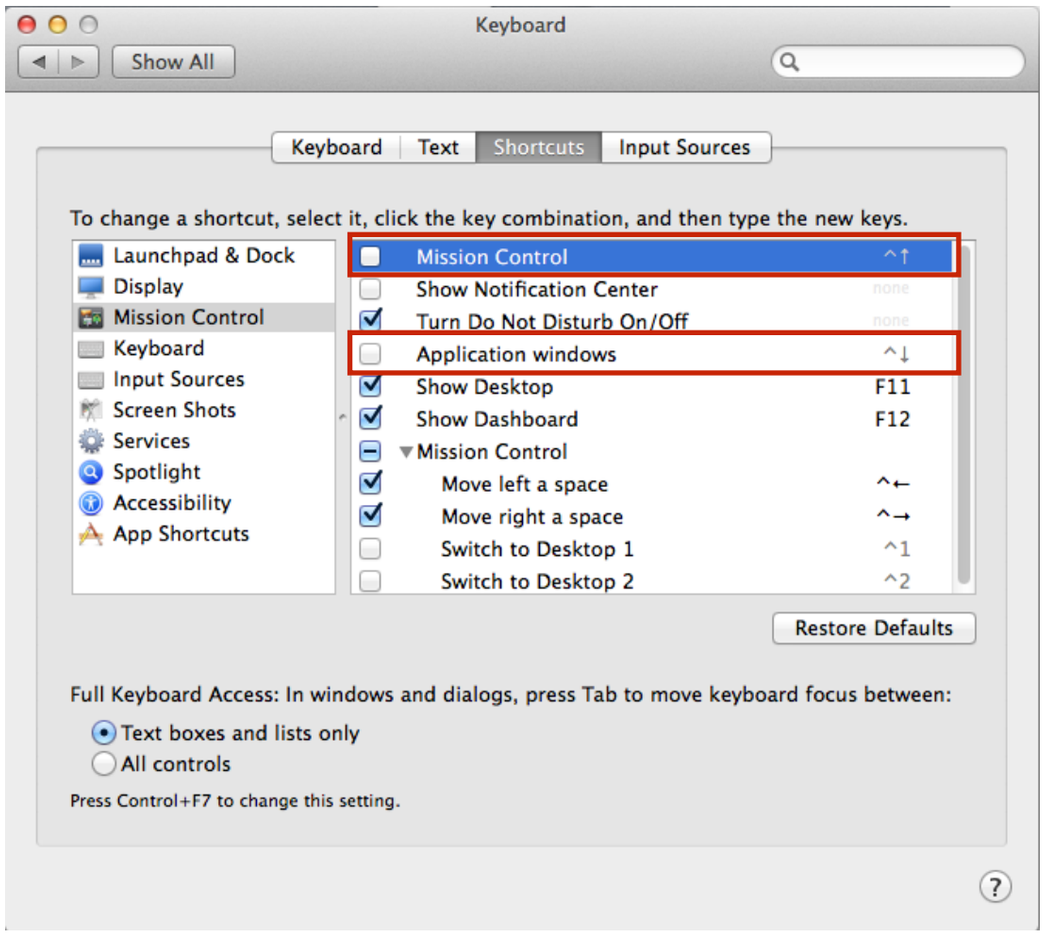
ATOM

1)

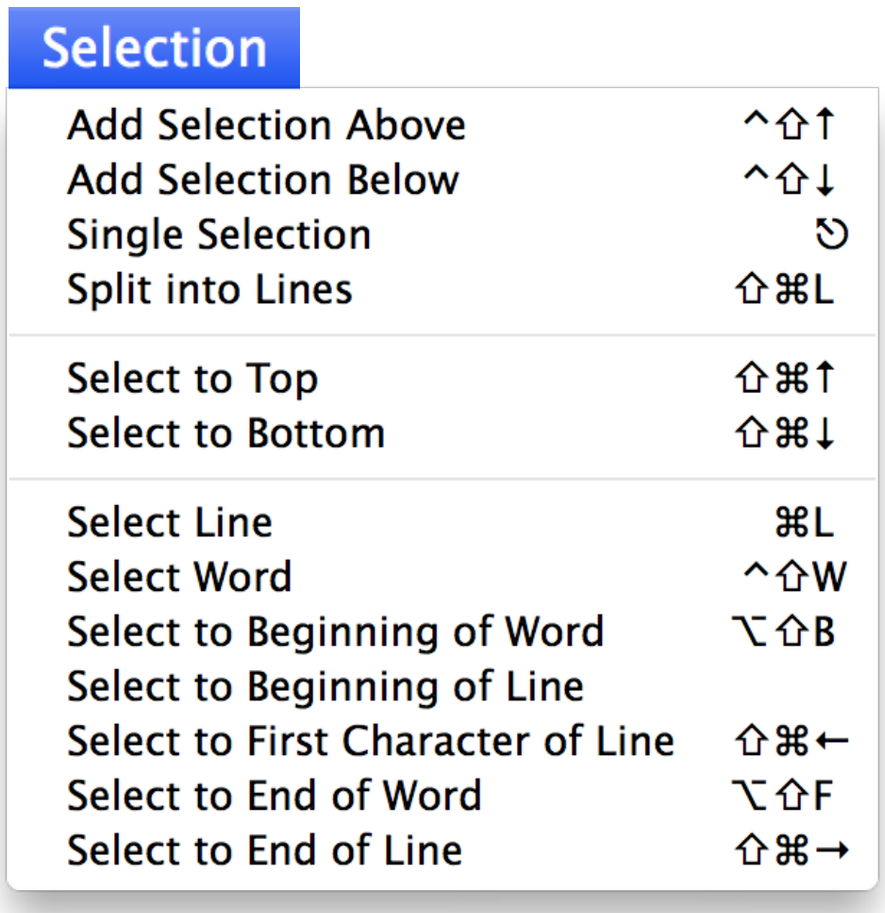
2)



3)



uncheck these two shortcuts

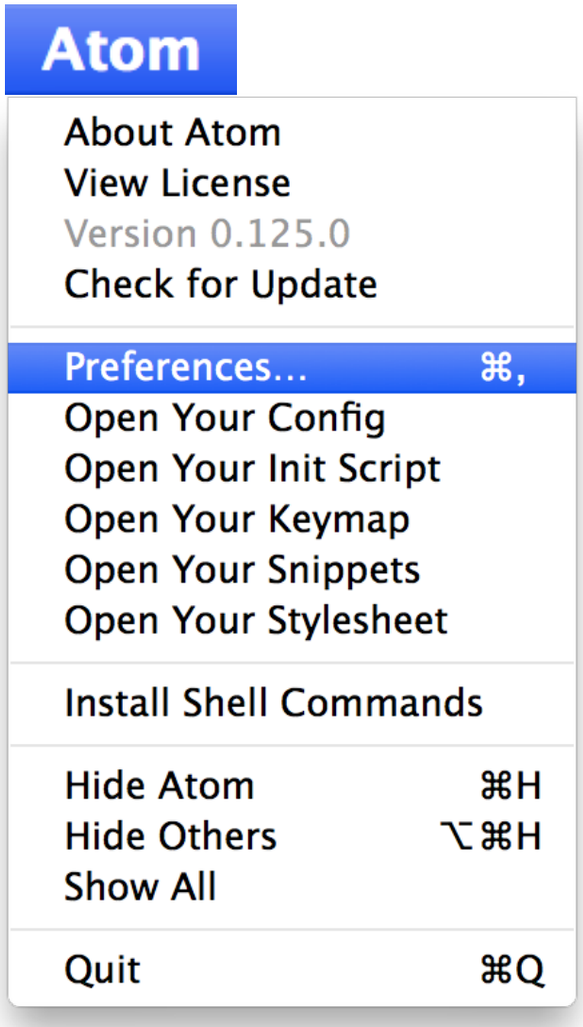


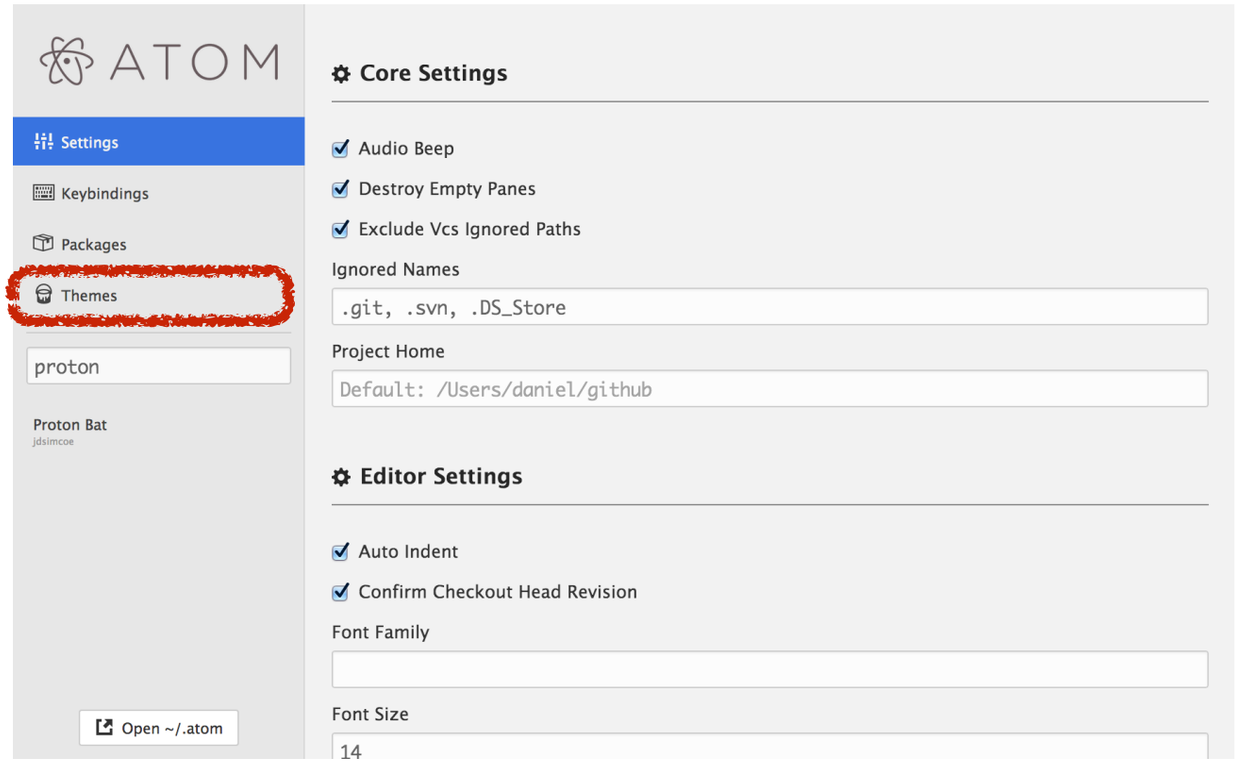
Open IBiS_bootcamp/resources/atom_selections_example.txt

Try the exercises, then try some of the options in the selection menu.

These functions and keyboard shortcuts are designed to *save you time*.

ATOM is customizable!





ATOM is customizable!

Choose a Theme

You can also style Atom by editing your stylesheet

UI Theme Atom Light

This styles the tabs, status bar, tree view, and dropdowns

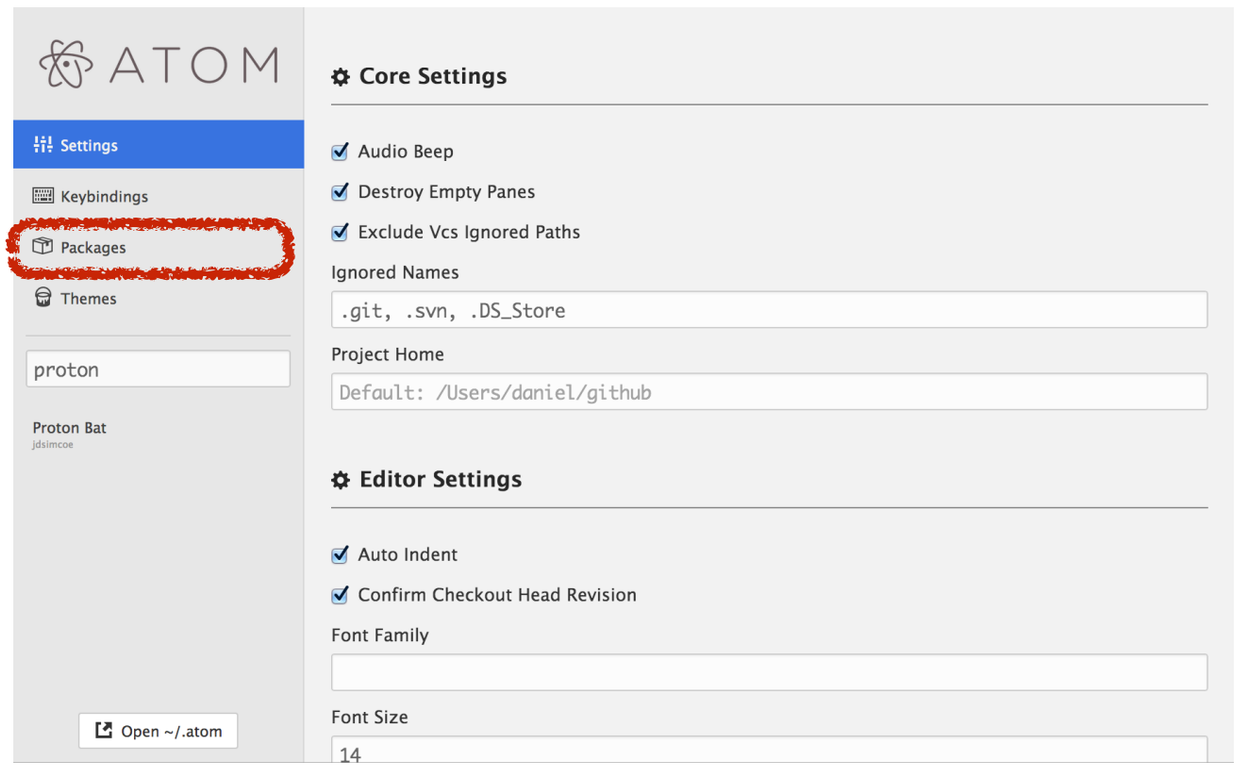
→ Overall look of program

Syntax Theme Proton Bat

This styles the text inside the editor

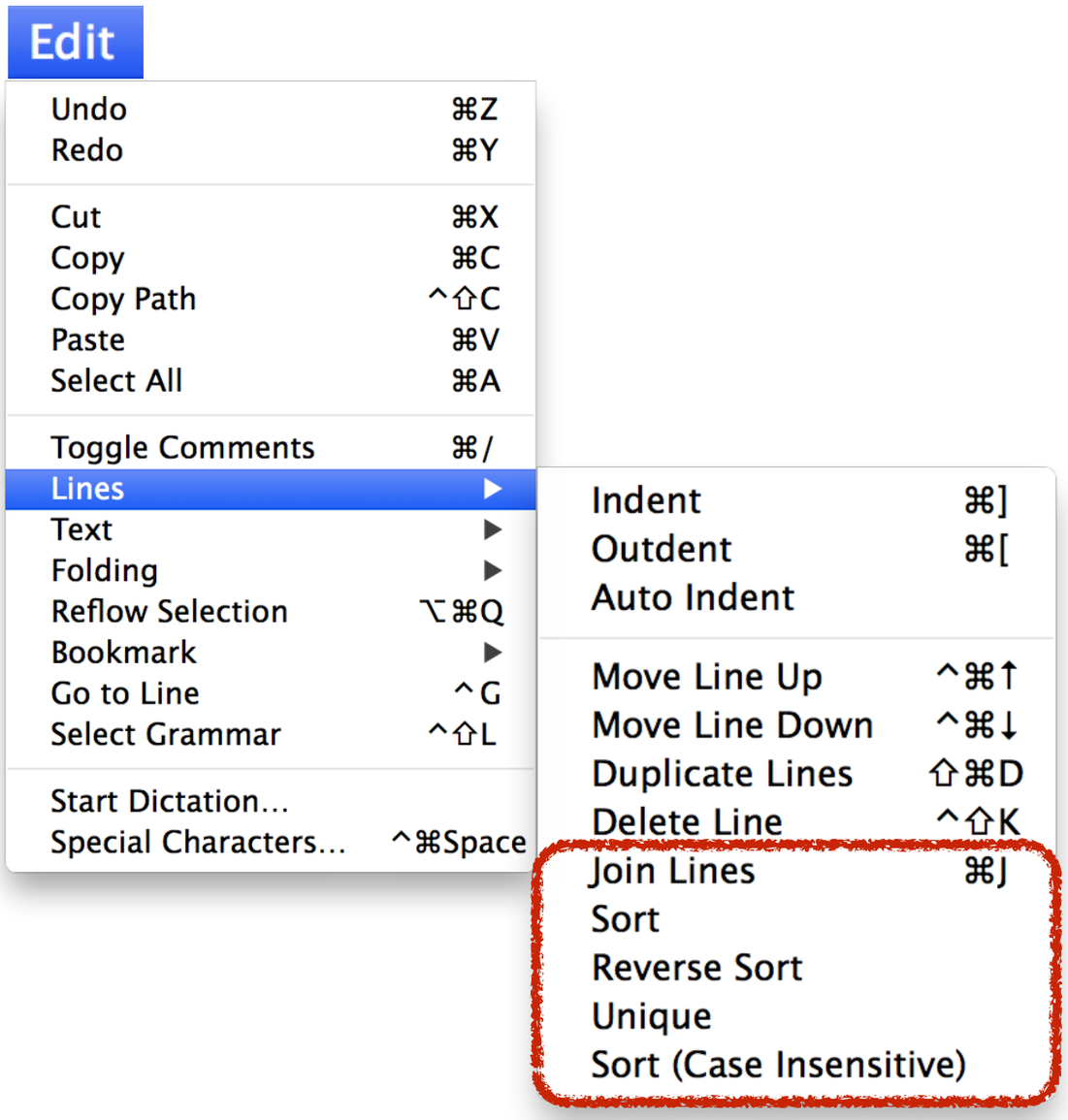
→ Color of text in the editor

ATOM is extensible!



Install ‘sort lines’

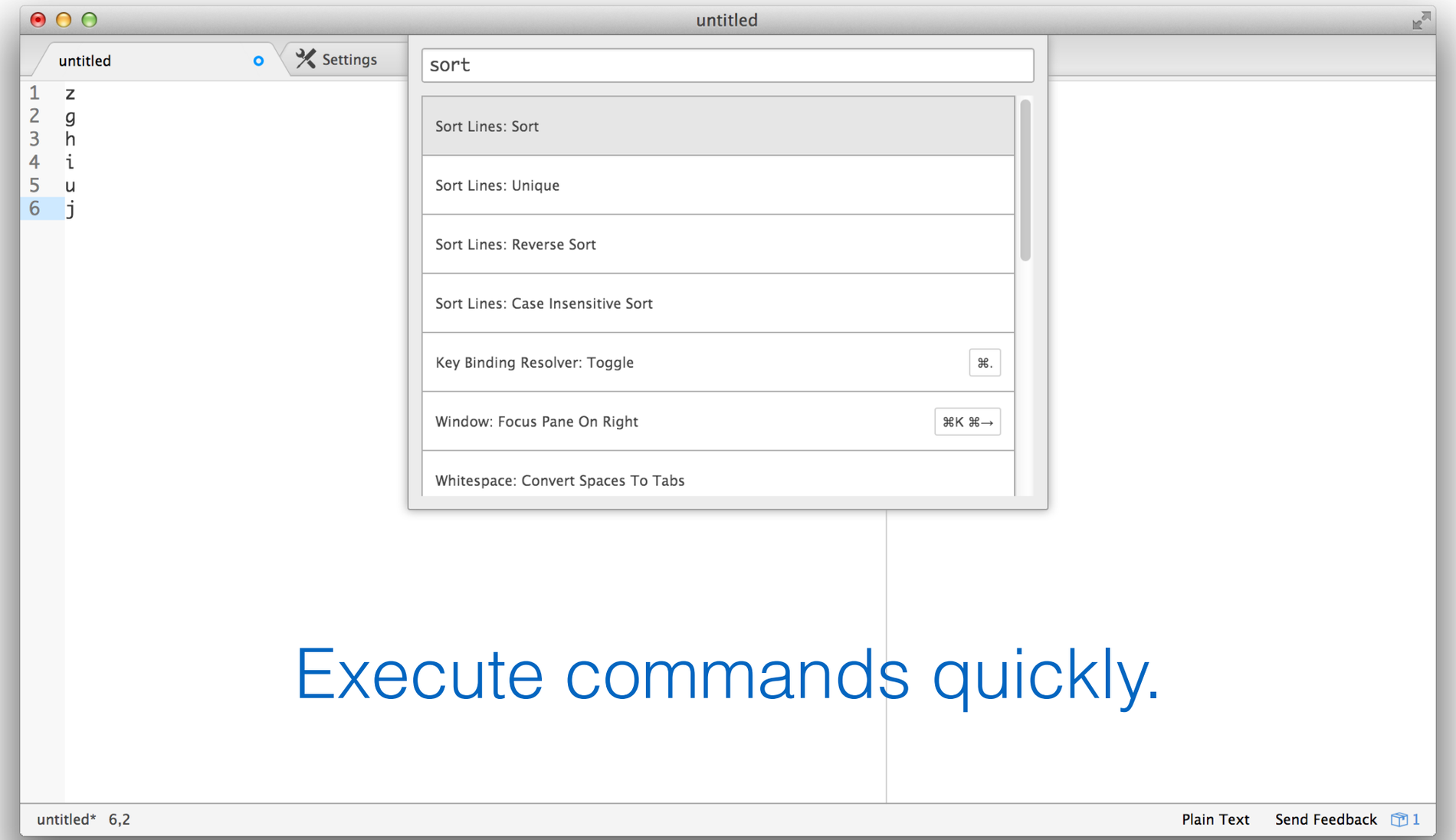
ATOM is extensible!



New functions!

ATOM

ATOM is has “command palette”! Command-shift-p



Let’s make readme.md from ATOM

1. Open Atom
2. Check out the welcome.md
3. Go to the untitled pane.
4. Type:

```
This example *project* will teach us **a lot** about basic computational biology.
```

5. Press command-shift-p
6. Type “markdown” and press enter
7. Check it out!
8. Close the markdown preview
9. Press control-shift-m
10. Check it out!
11. Save as “readme.md” into MyProject folder

Further experiences:
https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
http://daringfireball.net/projects/markdown/basics

Make an ibis.md file

1. Title: IBiS is awesome
2. “awesome” in bold
3. Make it a first level header with a line underneath it
4. Add normal text below: “Our webpage is here.”
5. Add the link for the IBiS webpage to “here”.
6. Make a list of the top three reasons you love IBiS
7. Add the IBiS image to the bottom

Bootcamp outline

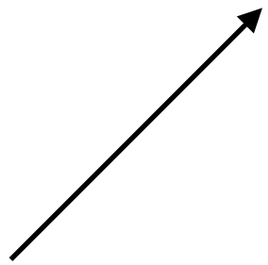
Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

- Intro to reproducible research
- Git and github
- Directory structures
- Basic command line
- Markdown
- **More command-tool utilities**
- R and Studio
- Data input

Notes vs `what you type`

The command line can be personalized!

.bash_profile



A hidden file in your home directory

- Your *bash profile* is run every time you open your terminal.
- You can use your bash profile to store common settings, functions, modify commands, or personalize your terminal.

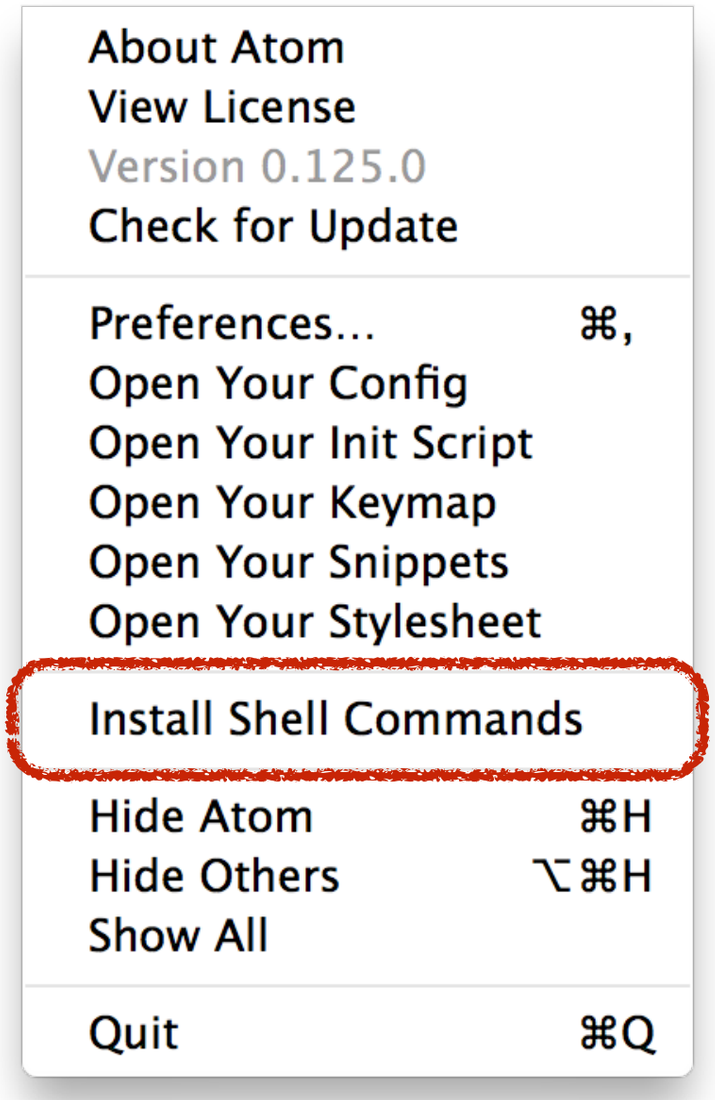
Editing your bash profile

1. Install the atom shell commands.

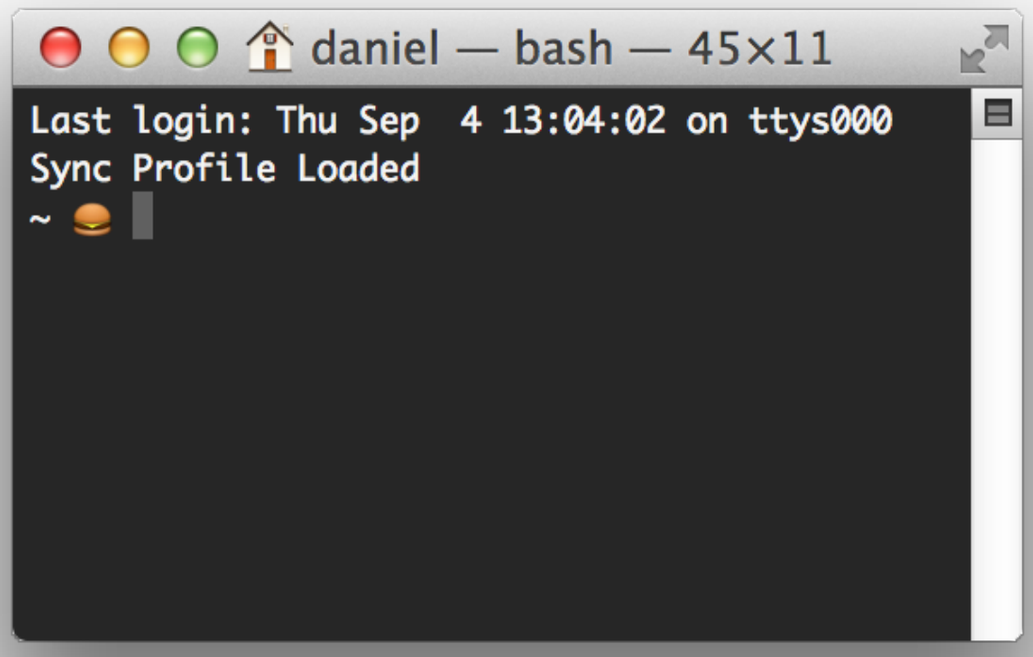
Atom > Install Shell Commands

2. Type:

```
atom .bash_profile
```

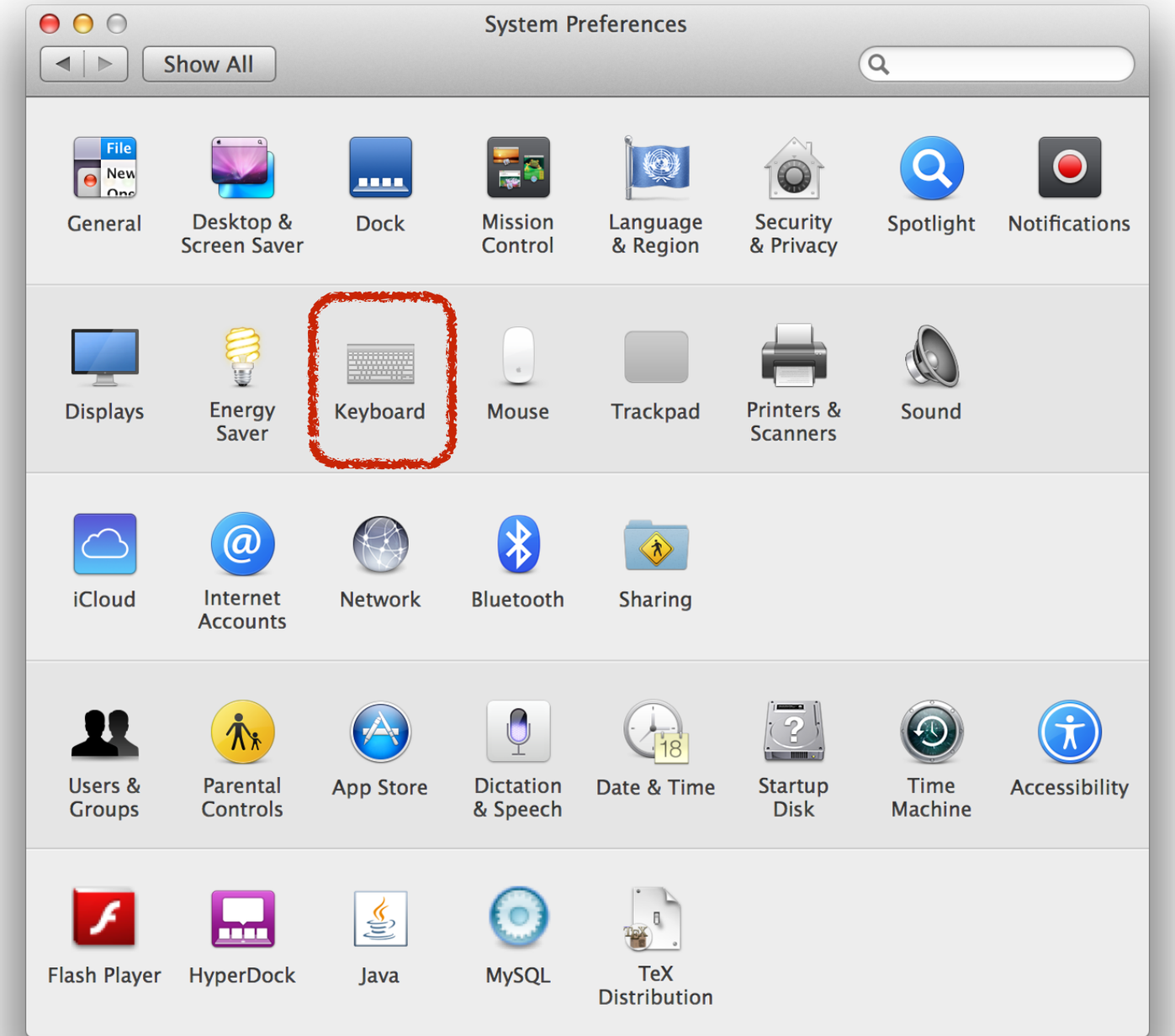


Personalize your Terminal

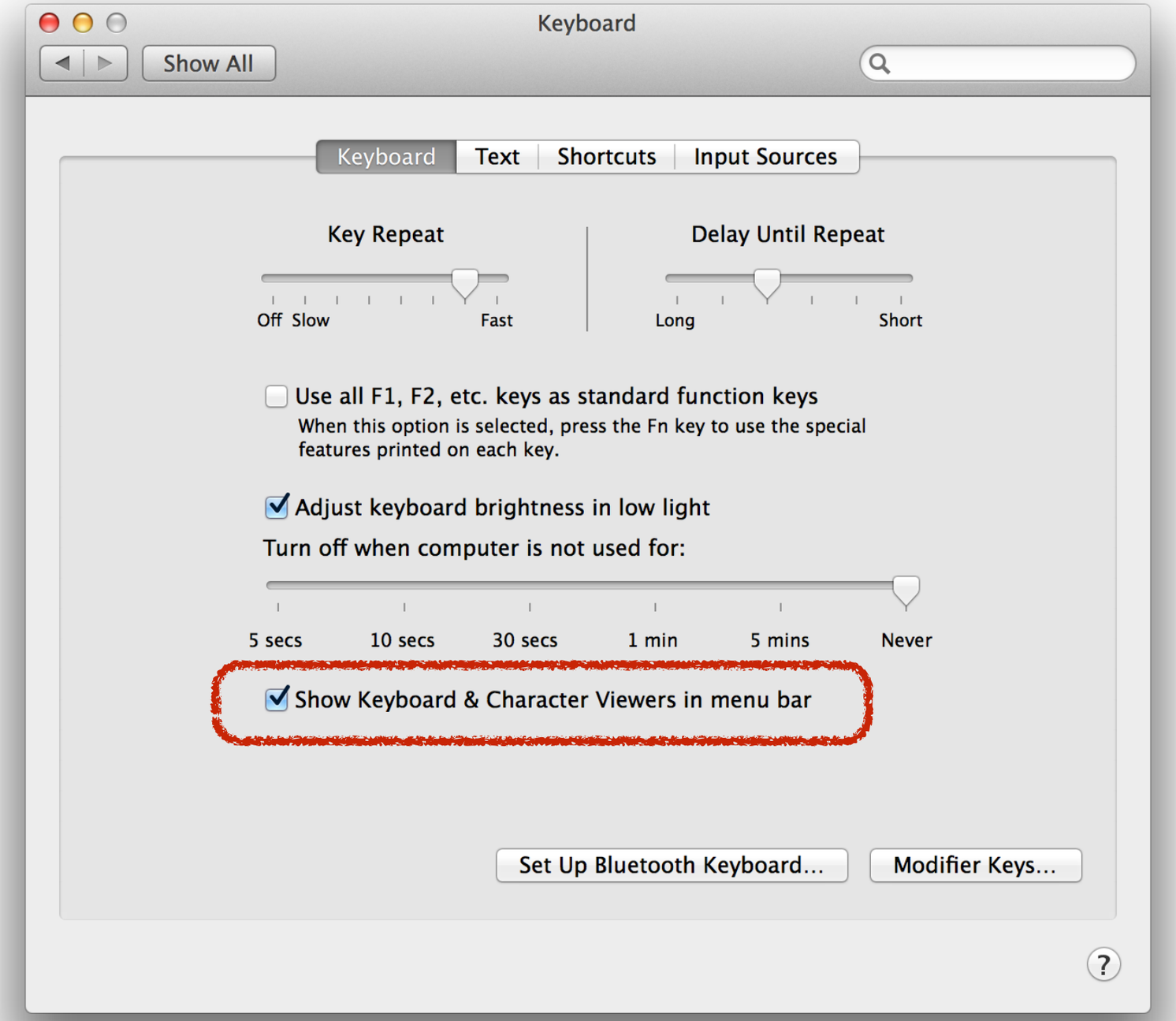


Add an icon to see command lines

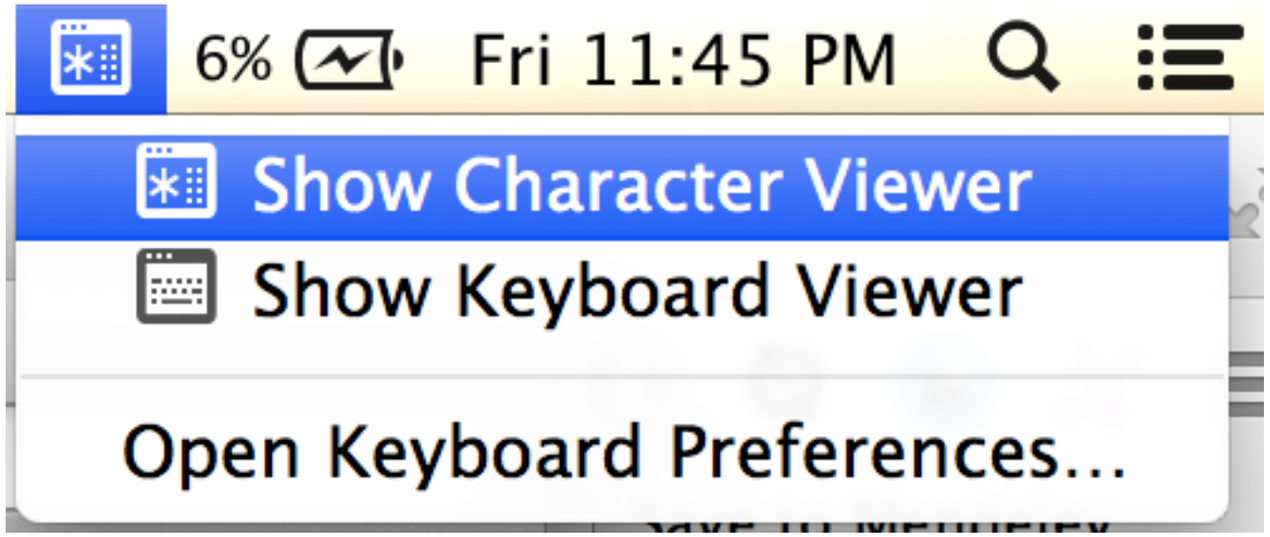
Open System Settings



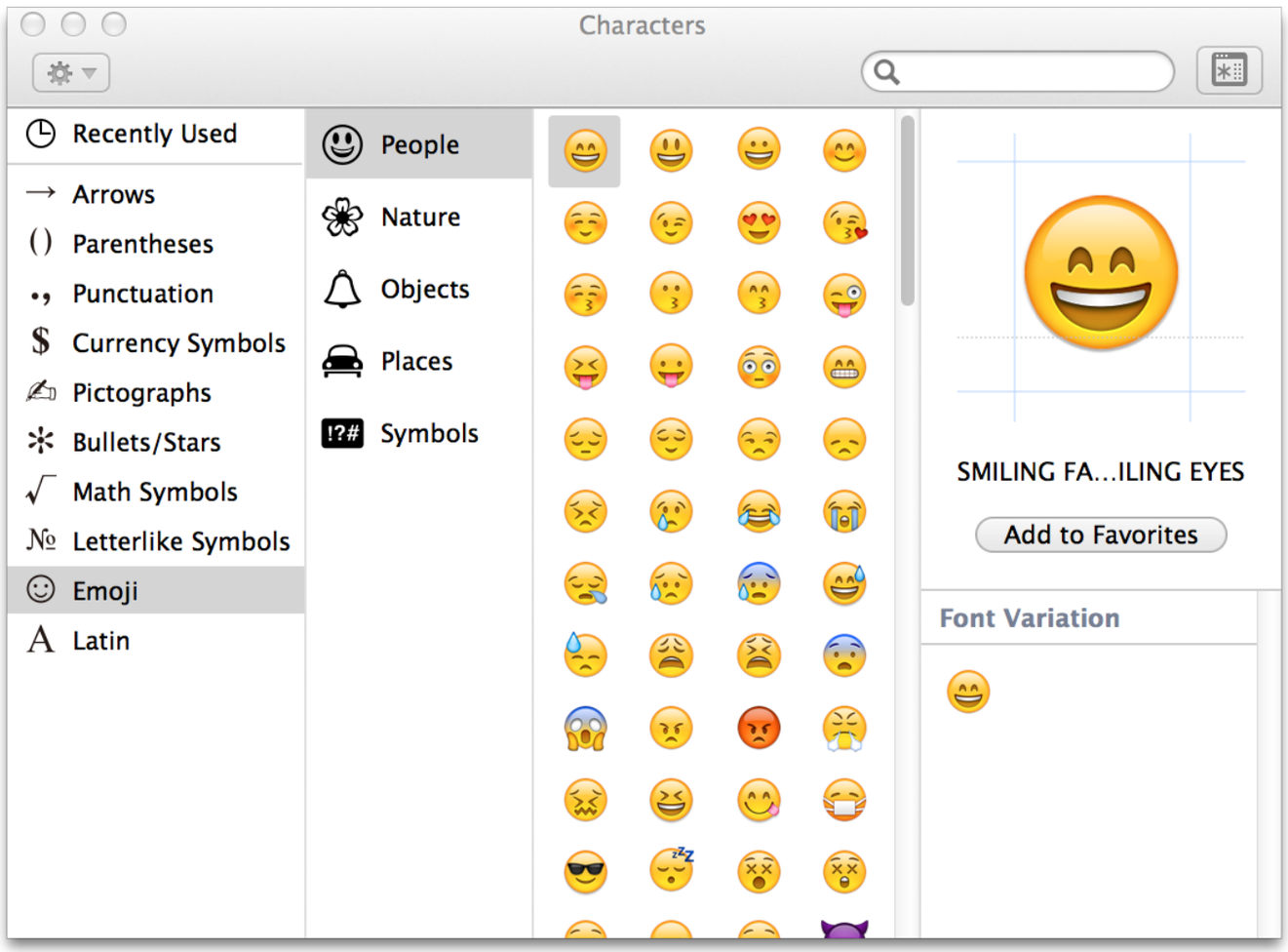
Click “Keyboard”



Check “Show Keyboard & Character Viewers in menu bar”

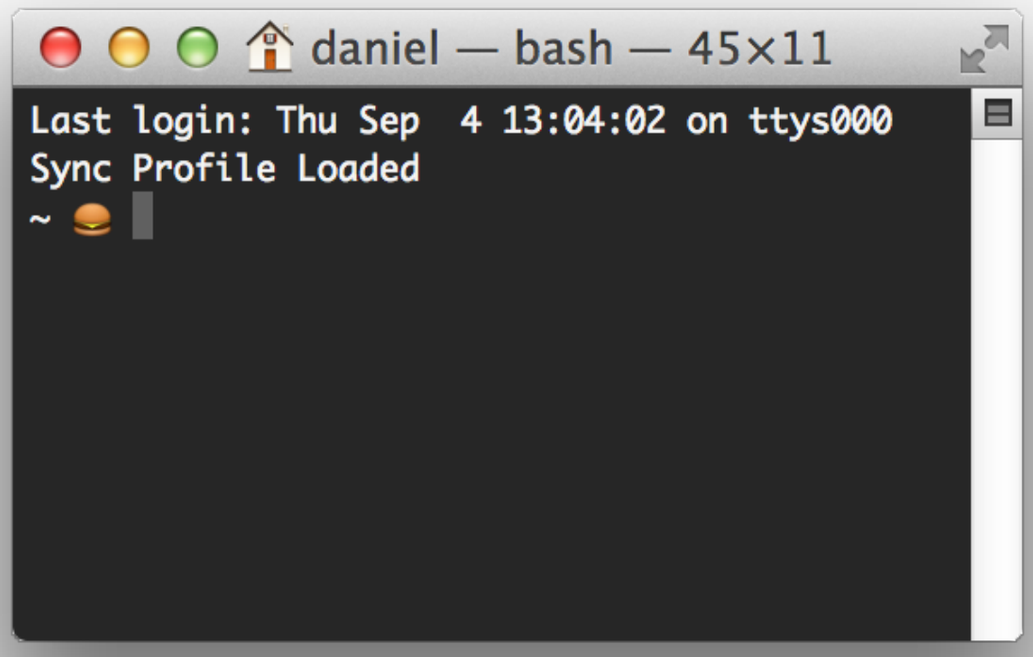


Open the character viewer.



Pick a fun icon!

Personalize your Terminal



Put your cursor here
and double click on the icon

```
export PS1="\w   "
```

Shortcuts can be added with aliases

```
alias refresh="source ~/.bash_profile"
alias dbx="cd ~/Dropbox/"
```

Typing ‘refresh’ will refresh your bash profile.

Typing ‘dbx’ will take you to the Dropbox directory.

Functions can be used to execute common tasks.

```
new_project() {
    mkdir $1
    touch $1/Readme.md
    mkdir $1/Data
    mkdir $1/Data/Raw
    mkdir $1/Data/Processed
    mkdir $1/Scripts
    mkdir $1/Results
}
```

This function will create a project directory structure.

Example:

```
new_project chip-seq-analysis
```

What to do when you are lost?

Google

stackoverflow

coursera

CODE ACADEMY

A few additional commands for UNIX

`time` = time how long an operation takes

`screen` = allows multiplexing of terminal windows

`nohup` = no hang ups, process errors go to a log

`sed` = search and edit text

`awk` = search, extract, or transform text

Also, check out datamash utilities

Further experiences:
http://freeengineer.org/learnUNIXin10minutes.html
http://www.ee.surrey.ac.uk/Teaching/Unix/
or just google: “unix command line tools”

Remote commands for UNIX

`ssh` = secure shell, remote access to UNIX systems

`rsync` = remote synchronization, remote copy

Let’s have some fun!

1. *Cool computer voice:* Type `say` and whatever you want. Make sure your sound is on

2. *Get your fortune:* Type `brew install fortune`. Then type: `fortune`

3. *Download youtube videos:* Type `brew install youtube-dl`. Then type: `youtube-dl cHK428vSIz8`

4. *Watch an asci movie:* Type: `telnet towel.blinkenlights.nl 23`

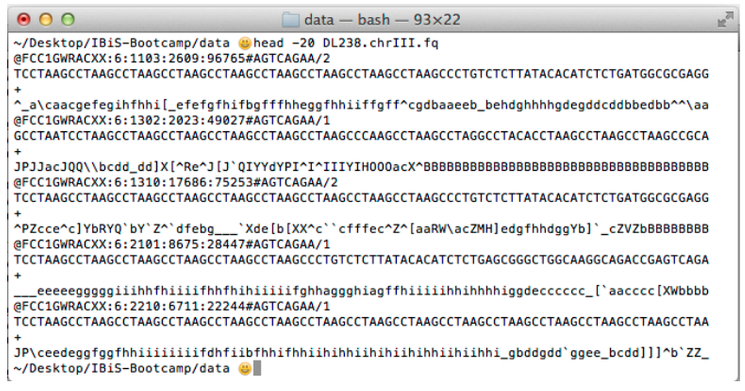
Bootcamp outline

Day #1: Reproducible research, Command line introduction, Intro to R and RStudio, and data input

- **Intro to reproducible research**
- **Git and github**
- **Directory structures**
- **Basic command line**
- **Markdown**
- **More command-tool utilities**
- **R and Studio**
- **Data input**

Notes vs `what you type`

The computational biology triumvirate



Free and open source are core principles of science

You ARE learning to code!

is a software environment for statistical computing

is free!

Software	Cost
	$8,700 - $140,000 / year
	$2150 + $1,000s for modules
	**$0**

is managed by users!

Comprehensive R Archive Network

http://cran.us.r-project.org

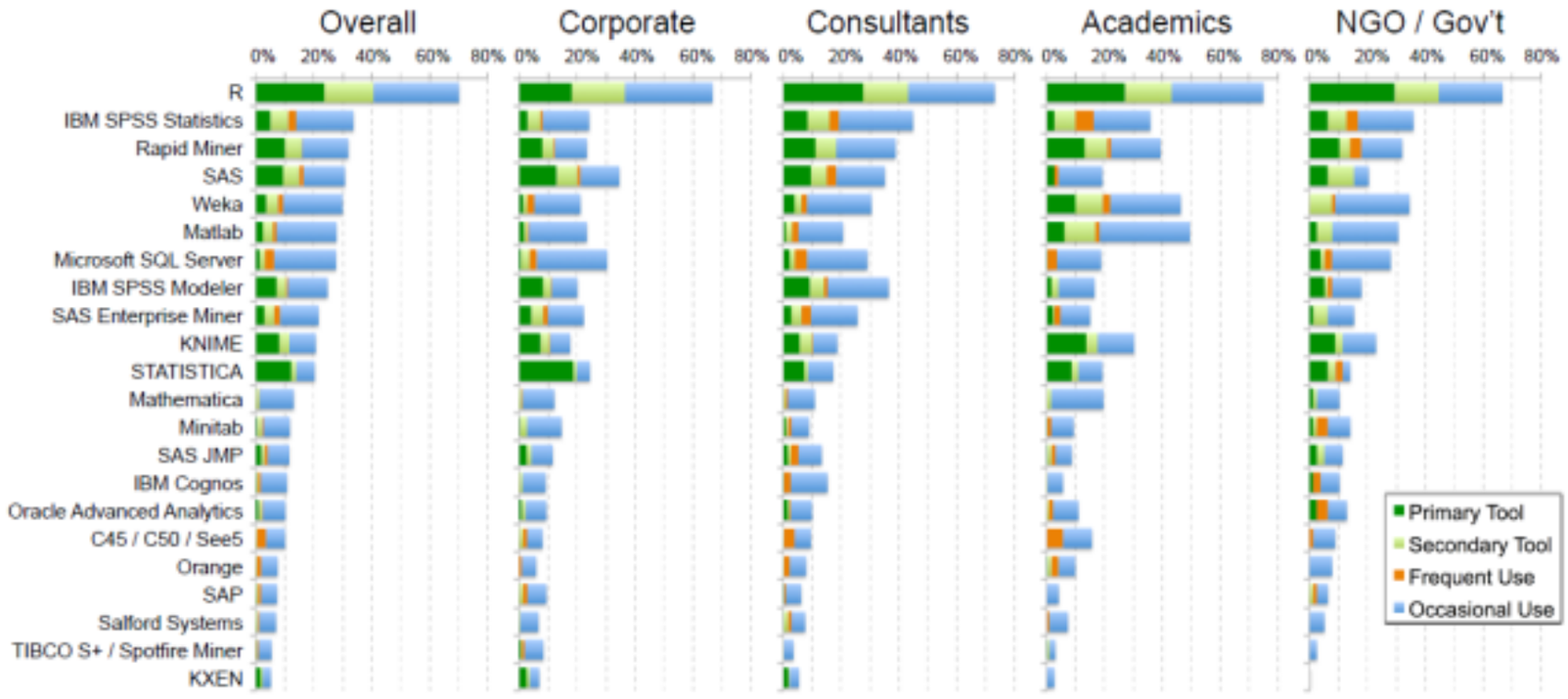
Over 5000 free add-on *packages*

Stack Overflow

http://stackoverflow.com

Free help

is popular for data analysis!



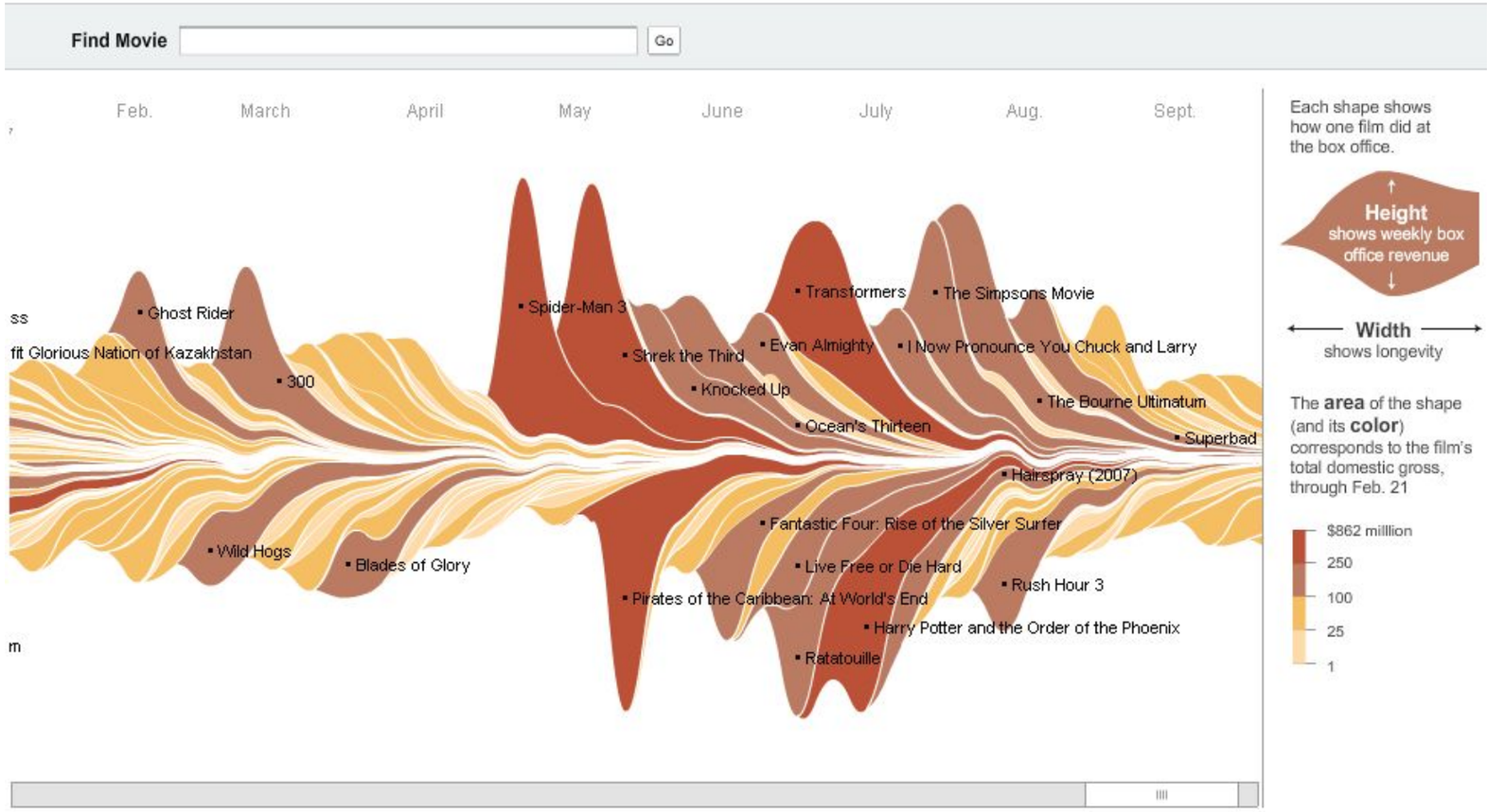
has amazing graphics!

February 23, 2008

SIGN IN TO E-MAIL OR SAVE THIS | FEEDBACK

The Ebb and Flow of Movies: Box Office Receipts 1986 — 2008

Summer blockbusters and holiday hits make up the bulk of box office revenue each year, while contenders for the Oscars tend to attract smaller audiences that build over time. Here's a look at how movies have fared at the box office, after adjusting for inflation.

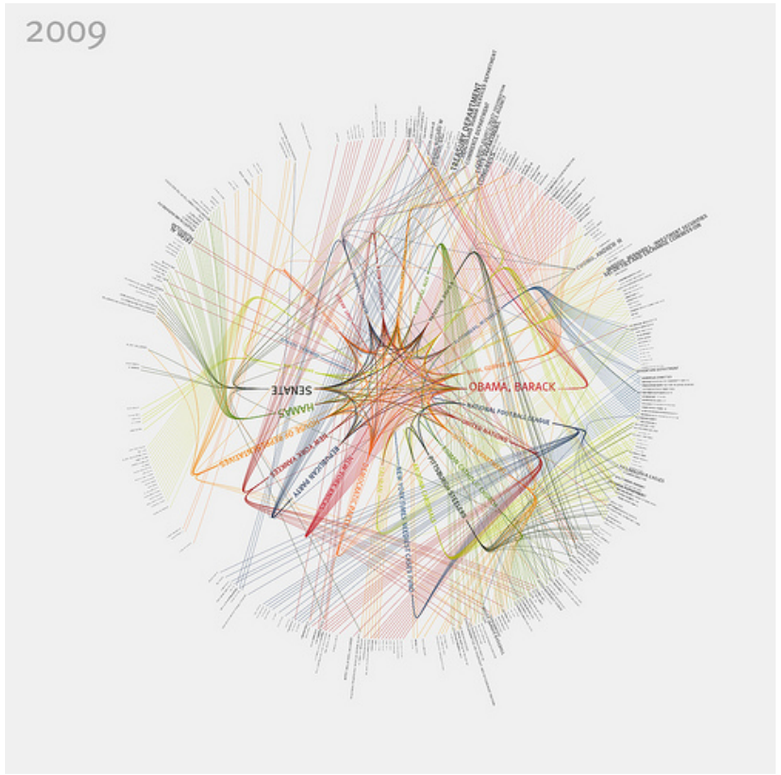


Sources: Baseline StudioSystems; Box Office Mojo

Mathew Bloch, Lee Byron, Shan Carter and Amanda Cox

New York Times favorite tool

has amazing graphics!



is a programming language!

- String handling, functions, object oriented, file i/o
- Parallel programming
- Statistically orientated data structures
- Runs command line utilities, python, C, etc.
- Integrates with a variety of database systems

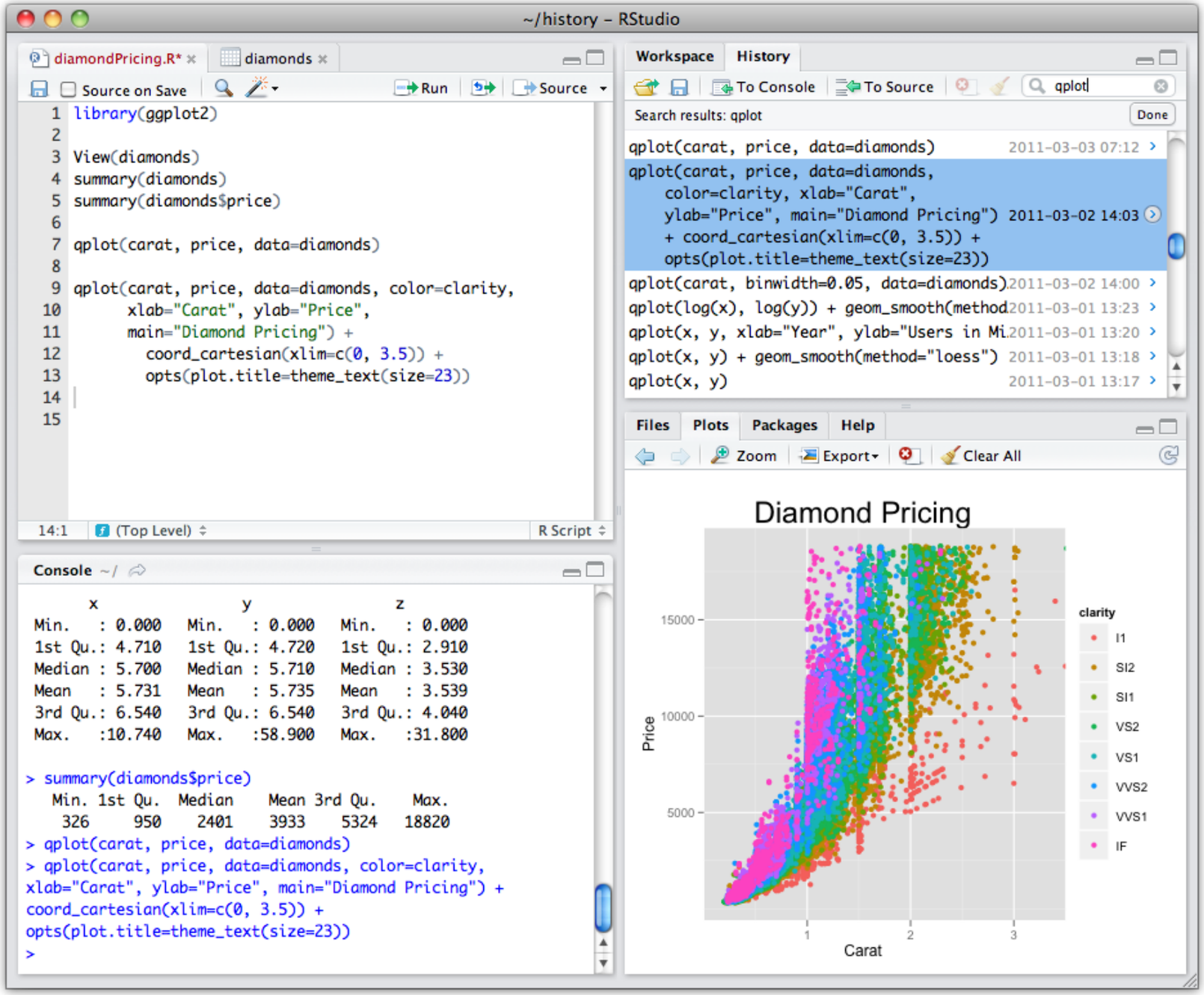
RStudio makes R even easier!

Editor

Workspace

Console

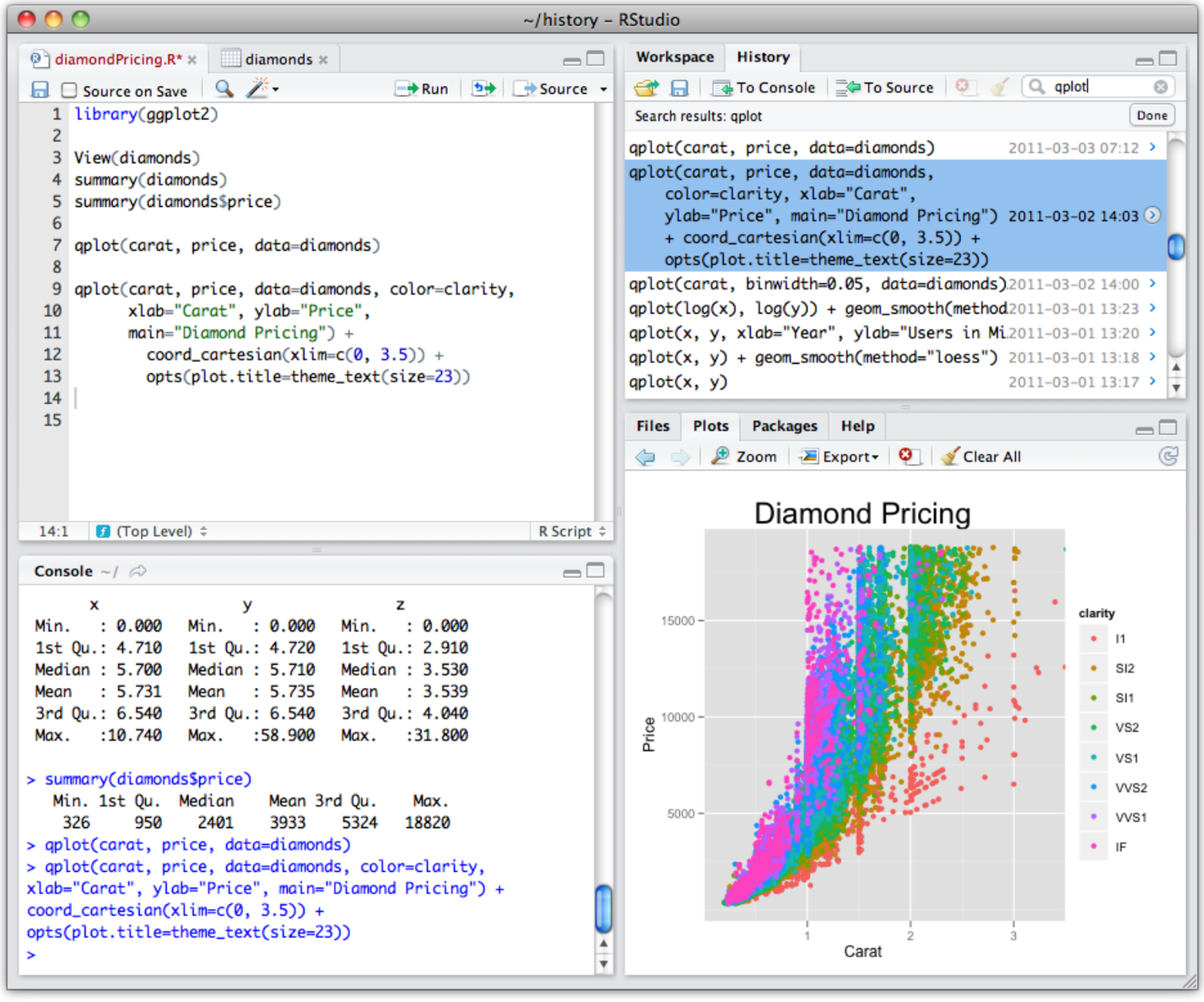
Graphics



Two main ways to interact with R

Editor

Console

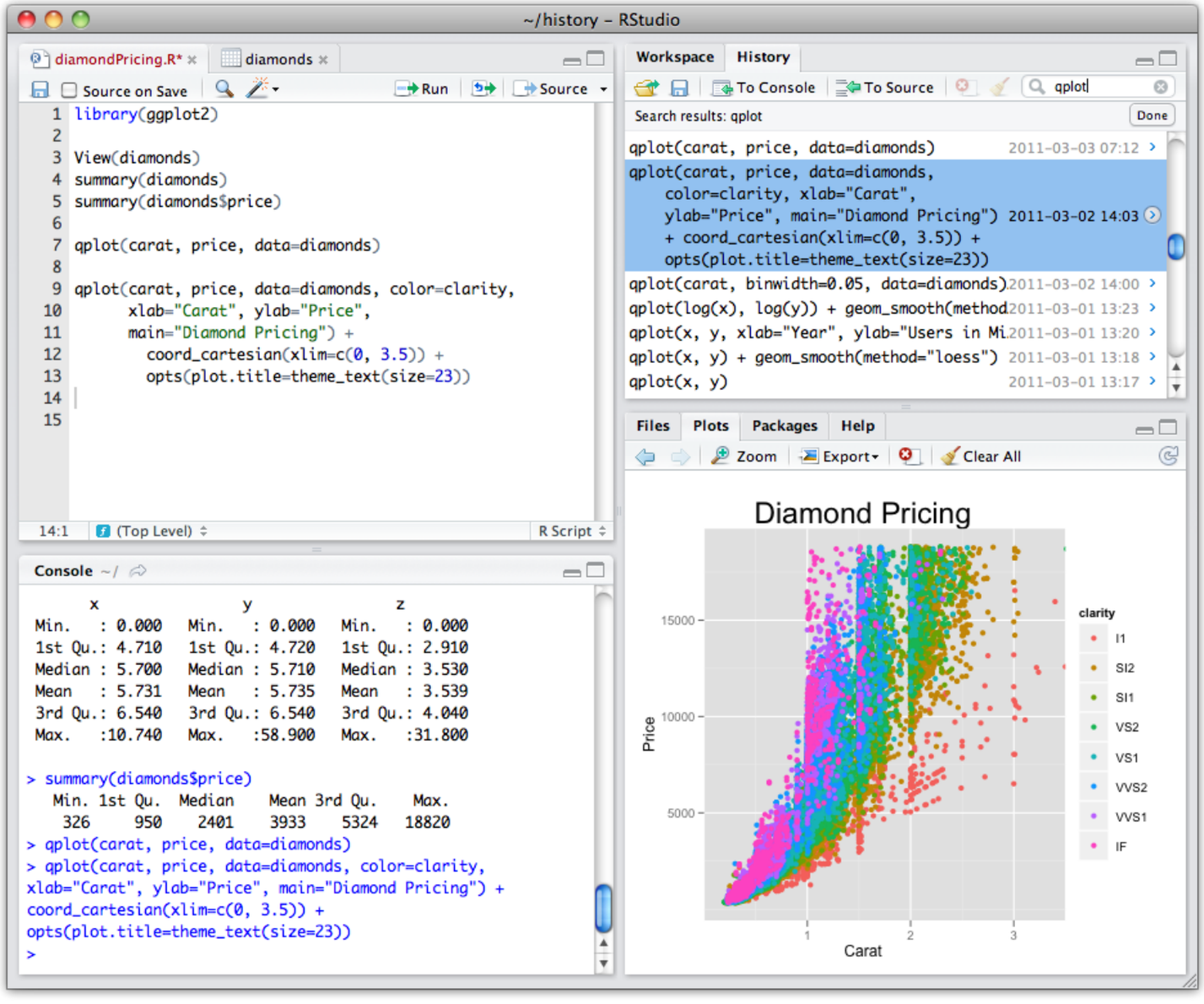


Think of the console as asking R questions about data.

Two main ways to interact with R

Editor

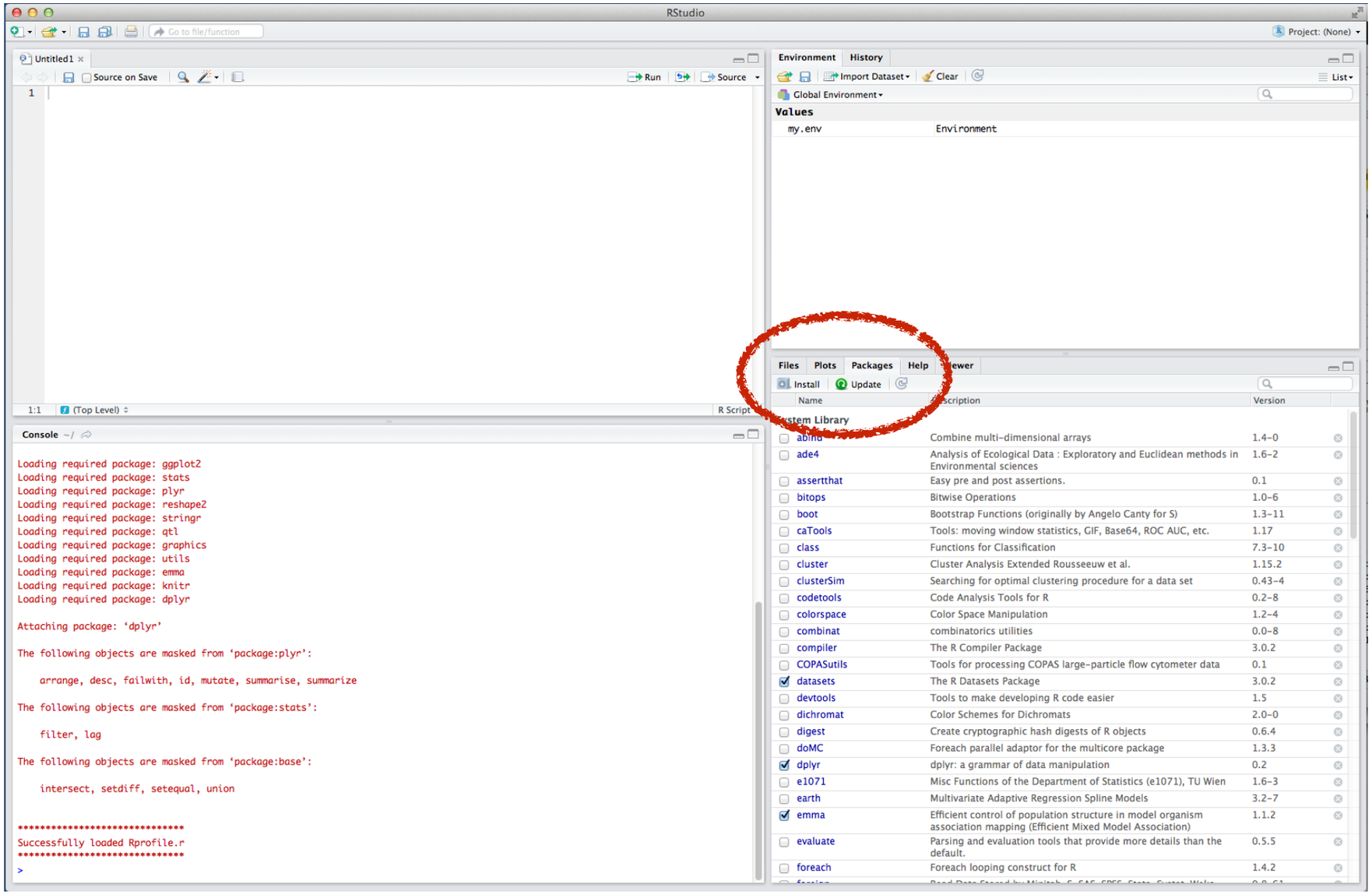
Console



The editor is a plain text file editor (like Atom)

For scripting and saving work (markdown too)

Let’s install some packages



1. Click on Packages in bottom right pane
2. Click on Install
3. Make sure the Install from drop down is CRAN
4. Enter **`lubridate`**

Basic R syntax

> The prompt

+ at the prompt means that something isn't complete

Functions are words followed by (), *i.e.* **`mean(x)`** for calculating the mean of x

Equations are entered as assignments, *i.e.* **`y <- mean(x)`** for y is equal to the mean of x.

If you can't remember a function, use **`apropos()`**. For example, **`apropos("var")`** will display every item in the workspace with "var"

Same goes for **`??var`**

You can also use ?function. For example, **`?mean`**

Try **`????mean`**

is used to enter comments not read by R

Basic Data Types in R

Atomic vector types

character
numeric
integer
logical
complex
raw

atomic = only holds data of one type

```
> # Assign 1 into y
> y <- 1
> y

> # Assign 1 to 10 into y
> y <- 1:10
> y

> str(y)
> View(y)
```

Basic Data Types in R

Atomic vector types

character
numeric
integer
logical
complex
raw

atomic = only holds data of one type

```
> # Assign 1 into y
> y <- 1
> y

> # Assign 1 to 10 into y
> y <- 1:10
> y

> str(y)
> View(y)
```

```
> length(y)
> y <- as.numeric(y)
> y
> str(y)

> as.character(y)
```

Basic Data Types in R

Atomic vector types

character
numeric
integer
logical
complex
raw

atomic = only holds data of one type

```
> length(y)
> y <- as.numeric(y)
> y
> str(y)

> as.character(y)
```

```
> y <- logical(5)
> y
> as.numeric(y)

> y <- c(1,2,3)
> str(y)
> y <- c("TRUE", "TRUE", "FALSE")
> str(y)
> y <- c("Bob", "Rick", "Ishwar")
> str(y)
```

Basic data manipulation

Adding elements:

```
> y <- c(y, “Carole”)
> y
> y <- c(“Greg”, y)
> y
```

Missing data (NA):

```
> z <- c(0.5, NA, 0.7)
> z
> is.na(z)
```

Special values:

```
> 1/0
> 0/0
```

Sequences:

```
> series <- 1:10
> series
> seq(1:10)
> seq(from =1, to =10, by = 0.1)
> LETTERS
> letters
> as.roman(2015)
```

Data structures in R

Vectors are one-dimensional data sets of any one data type

Data frames are two-dimensional data sets of any data type

Lists are groups of vectors, data frames, or other lists.

How do we explore data objects?

`names(df)` = gives the names of columns of df

`rownames(df)` = gives the names of rows of df

`colnames(df)` = gives the names of columns of df

`dim(df)` = outputs the number of rows then columns of df

`length(df)` = outputs the length of df

`summary(df)` = outputs summary statistics of df

Let’s make our own functions

```
> add20 <- function(x) {x + 20}

> add20(10)

> # This function will convert Fahrenheit to Celsius

> FtoC <- function(fahrenheit) {
          celsius = (fahrenheit - 32) * 5 / 9
          return(celsius)}
```

What do we do with “big data”?

Flow of data analysis

1. Read — Today
2. Tidy
3. Process — } Tomorrow
4. Plot
5. Present — } Thursday

We all owe Hadley Wickham a beer.

- Chief Scientist at RStudio
- Made tidyr package to clean up data
- Made dplyr package for grammar of data manipulation
- ggplot2 package for beautiful R graphics
- Made reshape2, stringr, lubridate, devtools, plyr, rvest, etc.

Choose your own data set for analysis

Baby names, life tables, births

```
devtools::install_github("hadley/babynames")
```

Electric power consumption

Go To: https://goo.gl/rGS5GN

All 2015 police shootings

https://github.com/washingtonpost/data-police-shootings

All UFO sightings

Go To: https://goo.gl/R6KmP2

Every Daily Show guest with Jon Stewart

Google: “analysis of guests Jon Stewart Stephen Turner”, go to RPubs link

Query IMDB and pull data

```
devtools::install_github("hrbrmstr/omdbapi")
```

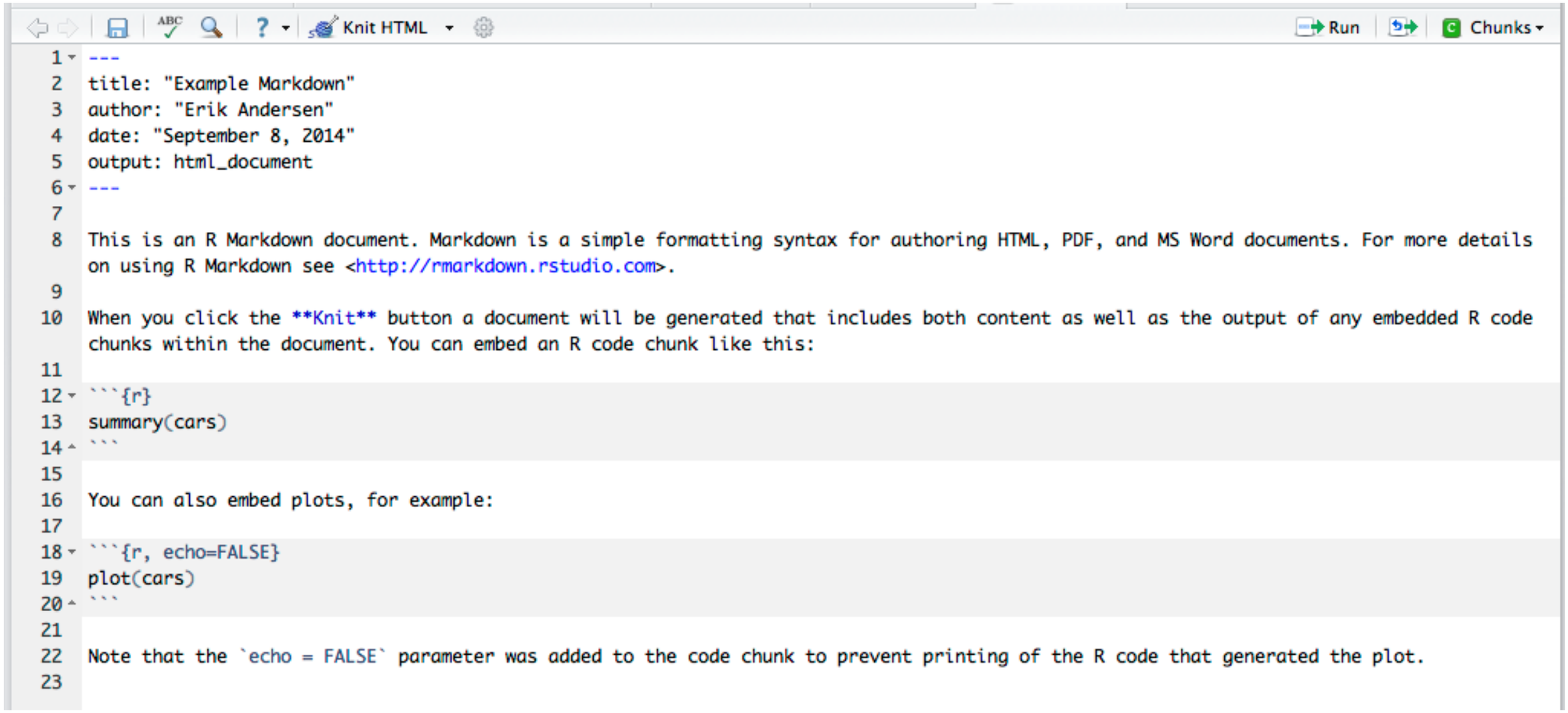
All LEGO sets from 1970-2014

```
devtools::install_github(“seankross/lego”)
```

Others…you find them…the web is filled with data
Check out:
http://koaning.io/posts/fun_datasets.html
https://data.cityofchicago.org/
https://github.com/datasets
https://github.com/caesar0301/awesome-public-datasets

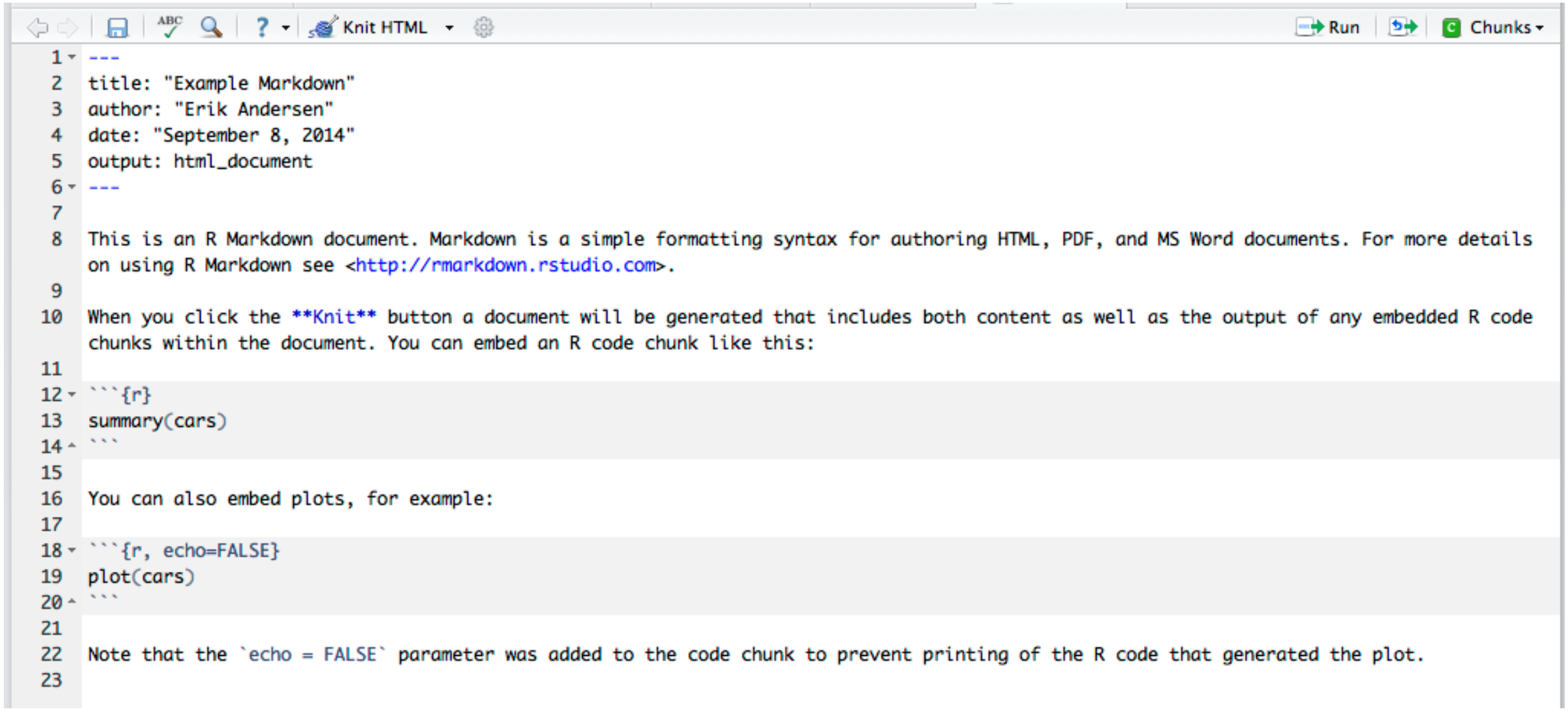
Time to find some data

RStudio can generate markdown reports



1. **Go to File, New File, R Markdown**
2. **Install the necessary packages**
4. **Knit an HTML report**

Make a markdown for your data analysis project



1. **Write a short description of the data and ideas you might have**

2. **Take a chunk to read in the data**

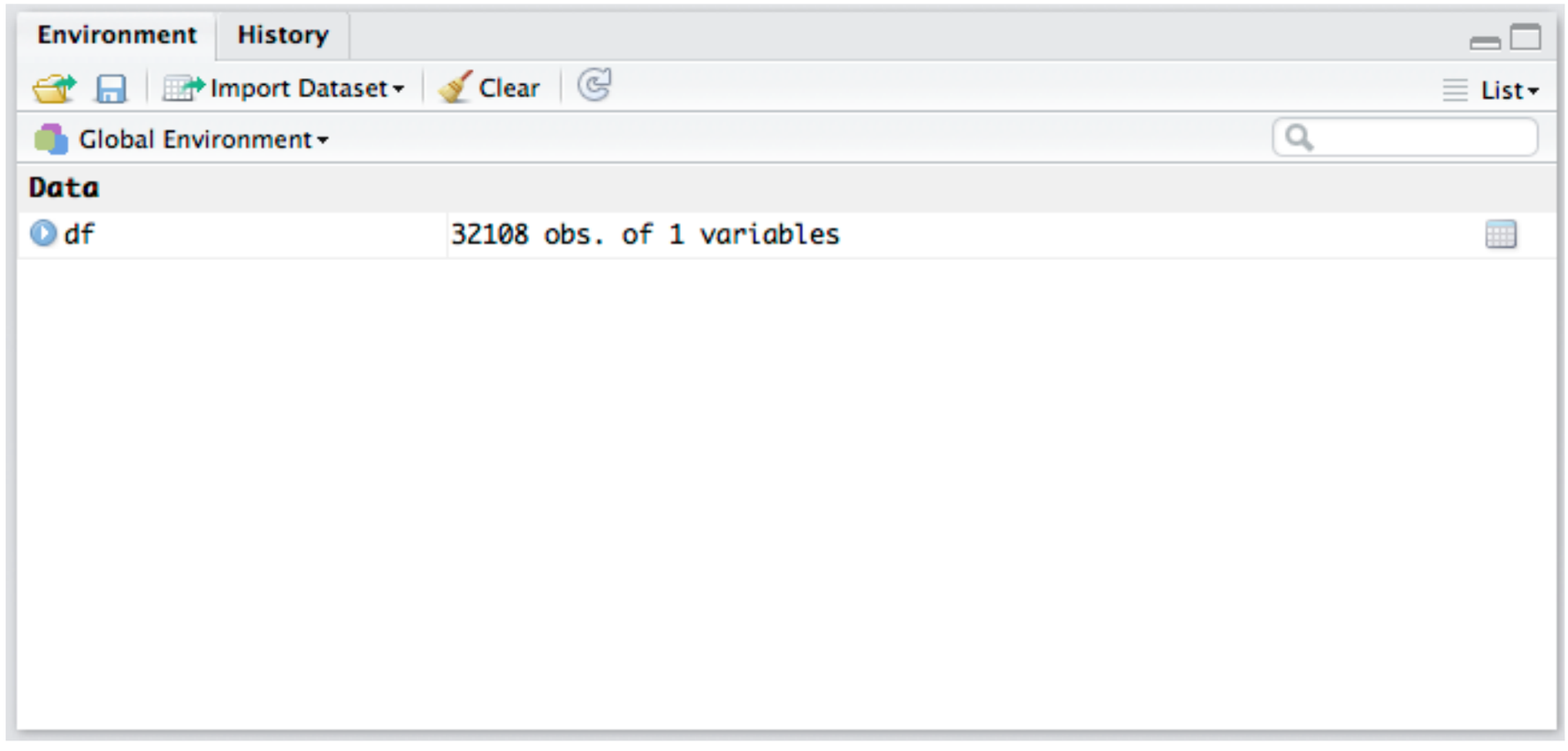
3. **Think of analyses you might want to do (time series, averages, etc.)**

4. **Prepare to clean the data!**

Reading in data

`read.delim()` is a generic function to read in simple text files, like csv or tsv files.

```
df <- read.delim(file="~/Desktop/IBiS-Bootcamp/data/sorter_data.txt", header=TRUE, sep="\t")
```



A new data object is in your workspace. Click on it. Type **`df`** to display all of it.
Try **`head(df)`** and **`tail(df)`**.
Try **`summary(df)`**.
Try **`View(df)`**.

Reading in data

Other packages have much more powerful data reading capabilities.

Try readr, `read_csv` or `read_tsv()`

Try rio, `import()`

A new data object is in your workspace. Click on it. Type **`df`** to display all of it.
Try **`head(df)`** and **`tail(df)`**.
Try **`summary(df)`**.
Try **`View(df)`**.

Reading in data

```
---
title: "20160911_nutrition"
author: "Erik Andersen"
date: "September 11, 2016"
output: html_document
---
For the 2016 bootcamp, I will analyze my nutrition data for the last year. The data were entered manually using the MyFitnessPal app.

```{r, echo=F}
library(rio)
library(lubridate)
library(dplyr)
library(ggplot2)
```

Read in data
```{r}
nut <- import("~/Dropbox/Personal/QuantifiedSelf/Nutrition-Summary2-2015-08-26-to-2016-08-28.csv")
```
```

Observing data

It looks like the tail has some data that do not match the pattern of the rest of the file.

```
df <- read.delim(file="~/Desktop/IBiS-Bootcamp/data/sorter_data.txt", header=TRUE, sep="\t", nrows=32093)
```

Type `str(df)` to look at the structure of the file

```
Console ~/
' ends field 1 on line 1 when detecting types: Green PMT voltage 600
> str(df)
'data.frame':   32093 obs. of  27 variables:
 $ Id         : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Plate      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Row        : chr  "A" "A" "A" "A" ...
 $ Column     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Clog       : chr  "N" "N" "N" "N" ...
 $ Scan.rate  : int  2500 2500 2500 2500 2500 2500 2500 2500 2500 2500 ...
 $ Status.sort: int  0 0 0 0 0 0 0 0 0 0 ...
 $ Status.sel : int  40 40 40 40 40 40 40 40 40 40 ...
 $ TOF        : int  52 556 88 639 531 34 247 152 96 79 ...
 $ EXT        : int  120 2375 130 2697 2520 36 534 474 210 178 ...
 $ Green      : int  2 148 3 205 151 1 16 66 20 4 ...
 $ Yellow     : int  3 193 5 241 170 3 21 26 7 5 ...
 $ Red        : int  0 90 2 102 71 2 14 11 4 2 ...
 $ PH.Ext     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Ext     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PC.Ext     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PH.Green   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Green   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PCGreen    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PH.Yellow  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Yellow  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PCYellow   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PH.Red     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PW.Red     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ PCRed      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Time.Stamp : int  2933417 2933417 2933432 2933432 2933463 2933463 2933463 2933495 2933495 2933495 ...
 $ X          : logi  NA NA NA NA NA NA ...
>
```

Data structures in R

Data.frames are data organized into rows and columns.

Data frames are organized by [row,column]. You must use the comma!

You can look at a vector of the data frame using $, *i.e.* **`df$TOF`**

Or you can specify the row or column to output, *i.e.* **`df[,9]`** outputs the ninth column and **`df[,"TOF"]`** outputs the column named “TOF”

Console ~/

```
> df[,"TOF"]
  [1]  52 556  88 639 531  34 247 152  96  79  54 133  64 471 417  21 491 400 504 163  25 436  57  36
 [25] 190  29 574 476 394  51 584  74 172 373  26  49  92  31  25 521 177 108 266  59  57 435  26 135
 [49]  62 197  52  38 236  97 545 438  82  28 129  76  67  48 105 140 147  22  88  74  24 265 162  21
 [73]  31 122  50  20  70  53 329  27  71  80 158 517 523 448  40  30  86  49 519 134  26  59  49  78
 [97]  65  46  91 219 259  21  85 128 244 216  59 188 182  56  43  75  51  60 136 114  80  77  52  58
[121] 219  26 203  81  25 127 190  53  63  54  37  40 468 197 172  59  85 196  45  23  24 245  58 209
[145] 314 400 229 488  99  35 260  46 103 199 421 116 481 133 112  40  55 164  51 161 212 161 128 212
[169]  64  23 491  47  20  22  69 471  23 120  28  51 524 124 265  20  69  22  62 481 103  32  53 143
[193]  48  37 203 116  89  31  45 255  73 281 296 105 472  49  20  43  20  46 113  92 143 194  46  87
[217] 194  23  70 525  54  52  51  46  36  59  71  27 506  22 331  73 847 438  58 160  50  55 402  31
[241] 208 524  58  50  78 479  54 102  55  73 295 516 109  38 121 142 127 132 486  44 101 335  66 393
[265]  94 205  77 172 205 565  29 133  54 443 511  34  97 139 182  25  39  27 182  50  60 101  56  59
[289] 146  44 206  48 118 324  36 310 127 537 194  59  46 111 411  61 439  74  32  51  91 115  46 250
[313] 467 229  47  58 181  25 103  35 184  42 295 185 406 119  32  61 549 635  84 186 111 179  49 142
```

Take some time to move through your data

1. Are there missing values? `is.na()`
2. What are the column names? `colnames()`
3. What are the dimensions? `dim()`
4. What are the row values? `rownames()`
5. Are the data organized long or wide?

Some thoughts about organizing raw data

- Be consistent.
- Write dates as `YYYY-MM-DD` or YYYYMMDD.
- Fill in all of the cells.
- Put just one thing in a cell.
- Make it a rectangle.
- Create a data dictionary.
- No calculations in the raw data files.
- Don’t use font color or highlighting as data.
- Choose good names for things.
- Make backups.
- Use data validation to avoid data entry mistakes. Unit tests!
- Save the data in plain text files.

from Karl Broman (UW Madison), check out his fantastic blog
Other great rules here: https://github.com/jtleek/datasharing

The most important part of data analysis is to *think* about your data

What do you want to test?

How will you show that conclusion?

Put the goal at the top of your markdown report

Time to read in data, explore your data, and plan analyses

What to do when you are lost?

Google

http://www.r-bloggers.com

http://gettinggeneticsdone.blogspot.com

http://rseek.org

Day #1 Homework

1. Sign up for github
2. Star our class repository
3. Download Github desktop and login with your Github login
4. Go and explore github! Find some cool code.
5. Install the swirl package
6. Type `swirl()`
7. Take the introduction to R programming class
8. Play with your new data set