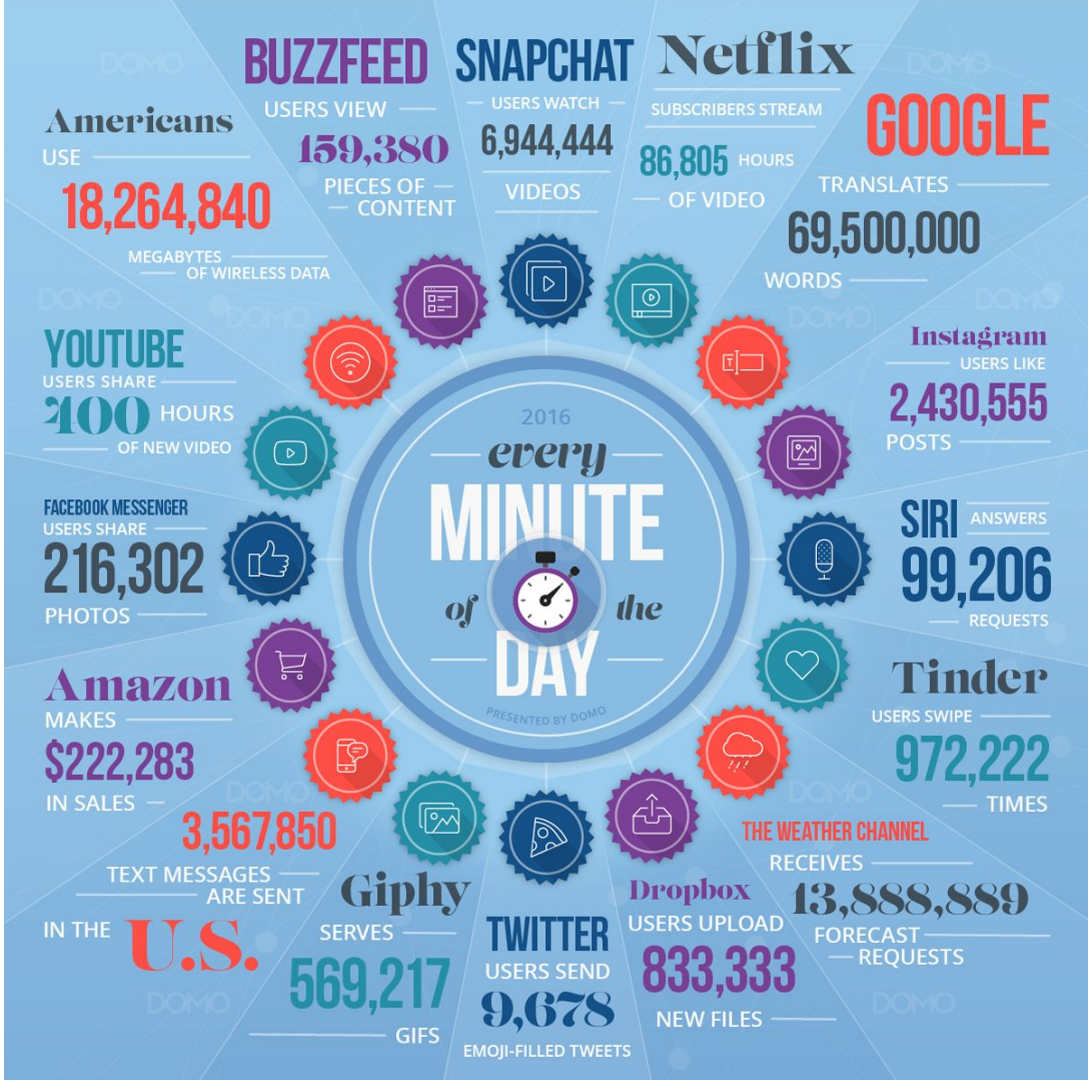
INFO 201

Technical Foundations of Informatics

By the time I get to the next slide, lots of data will have been produced.


2016
every
MINUTE
of the
DAY
PRESENTED BY DOMO
Americans
USE
18,264,840
MEGABYTES
OF WIRELESS DATA
BUZZFEED
USERS VIEW
159,380
PIECES OF
CONTENT
SNAPCHAT
USERS WATCH
6,944,444
VIDEOS
Netflix
SUBSCRIBERS STREAM
86,805 HOURS
OF VIDEO
GOOGLE
TRANSLATES
69,500,000
WORDS
YOUTUBE
USERS SHARE
100 HOURS
OF NEW VIDEO
Instagram
USERS LIKE
2,430,555
POSTS
FACEBOOK MESSENGER
USERS SHARE
216,302
PHOTOS
SIRI ANSWERS
99,206
REQUESTS
Amazon
MAKES
$222,283
IN SALES
Tinder
USERS SWIPE
972,222
TIMES
3,567,850
TEXT MESSAGES
ARE SENT
IN THE U.S.
Giphy
SERVES
569,217
GIFS
TWITTER
USERS SEND
9,678
EMOJI-FILLED TWEETS
Dropbox
USERS UPLOAD
833,333
NEW FILES
THE WEATHER CHANNEL
RECEIVES
13,888,889
FORECAST
REQUESTS

source

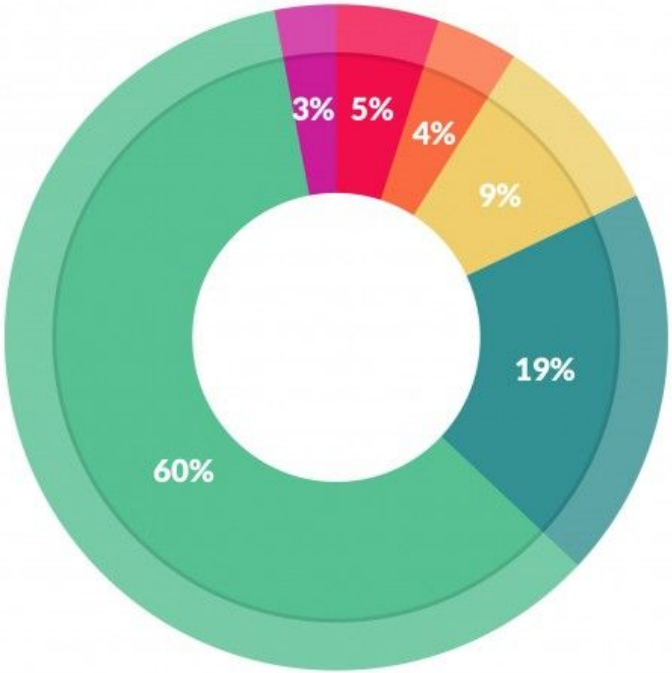
That’s a lot of data

Some of which is interesting...

How Computer Scientists Are Using Twitter to Predict Gentrification

Cambridge researchers have created a way to predict a neighborhood's fortunes in coming years by analyzing social media data

Many pertinent / interesting questions to be asked



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

Need to know foundational skills

Today’s Outline

Course Overview

Introductions

Course Structure

Programming Language Landscape

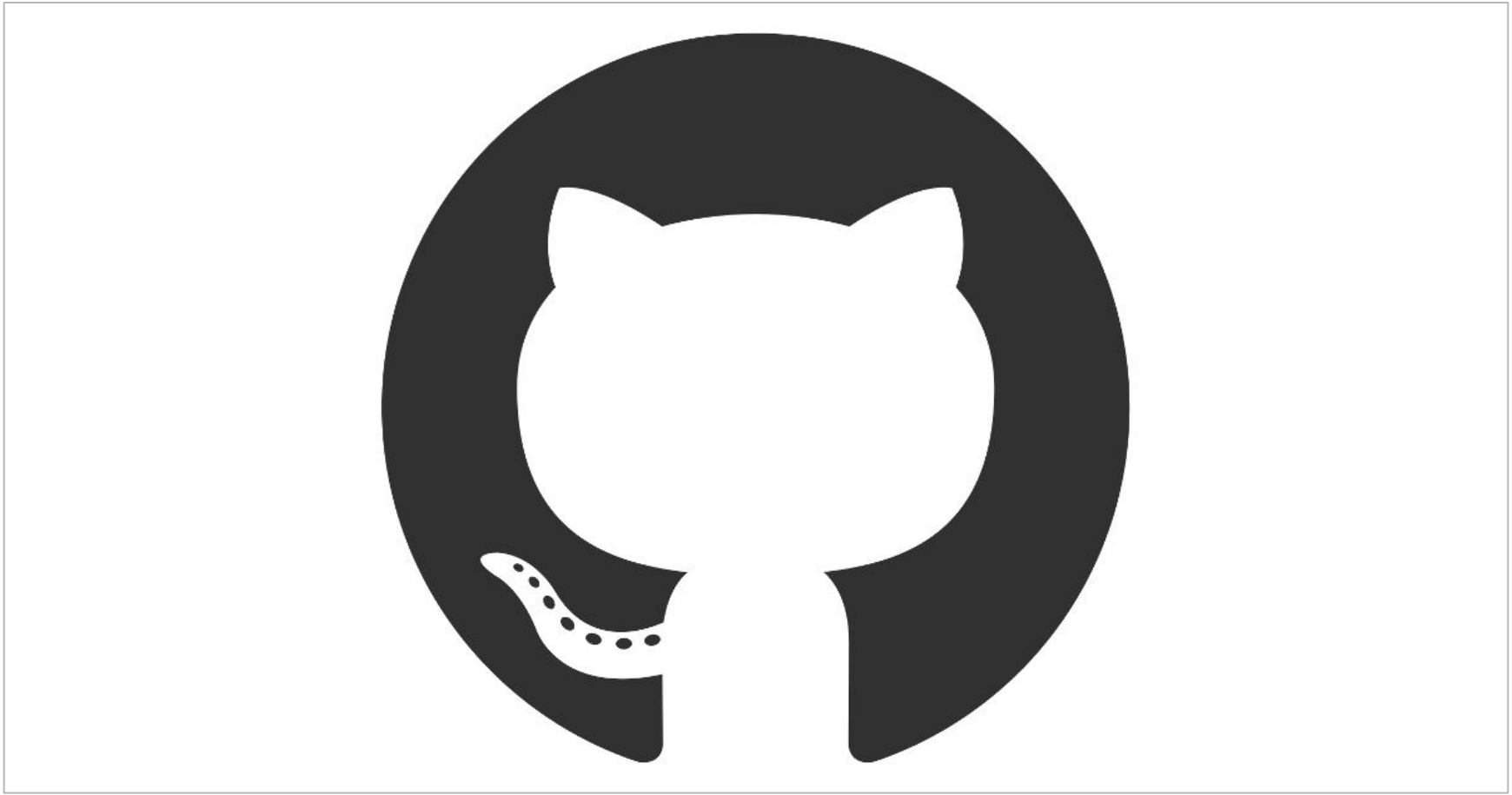
Using the Command Line

Course Overview

Learn the technical skills and tools necessary for working with information.

```r
# You'll write lots of code in this class.

# Right now it might look like nonsense:

x <- 10

# But it's super powerful
square <- function(a) {
    return (a*a)
}

# Any guess what y is?
y <- square(x)
```

Write Code

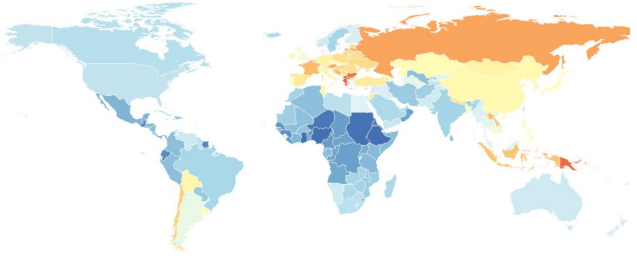


Manage Code

	A	B	C	D	E
1	country	region	income_group	le_1960	le_2013
2	Aruba	Latin America & Caribbean	High income: nonOECD	65.56936585	75.33217073
3	Afghanistan	South Asia	Low income	31.58004878	60.93141463
4	Angola	Sub-Saharan Africa	Upper middle income	32.98482927	51.86617073
5	Albania	Europe & Central Asia	Upper middle income	62.25436585	77.5372439
6	United Arab Emirates	Middle East & North Africa	High income: nonOECD	52.24321951	77.13129268
7	Argentina	Latin America & Caribbean	High income: nonOECD	65.21553659	76.18729268
8	Armenia	Europe & Central Asia	Lower middle income	65.86346341	74.5407561
9	Antigua and Barbuda	Latin America & Caribbean	High income: nonOECD	61.78273171	75.82929268
10	Australia	East Asia & Pacific	High income: OECD	70.81707317	82.19756098
11	Austria	Europe & Central Asia	High income: OECD	68.58560976	80.8902439
12	Azerbaijan	Europe & Central Asia	Upper middle income	60.8362439	70.69314634
13	Burundi	Sub-Saharan Africa	Low income	41.23604878	54.09719512
14	Belgium	Europe & Central Asia	High income: OECD	69.70195122	80.38536585
15	Benin	Sub-Saharan Africa	Low income	37.27826829	59.28756098
16	Burkina Faso	Sub-Saharan Africa	Low income	34.47790244	56.27502439
17	Bangladesh	South Asia	Lower middle income	47.02456098	70.69339024
18	Bulgaria	Europe & Central Asia	Upper middle income	69.24756098	74.46585366
19	Bahrain	Middle East & North Africa	High income: nonOECD	52.08936585	76.66987805

(you'll write the code to do this!)

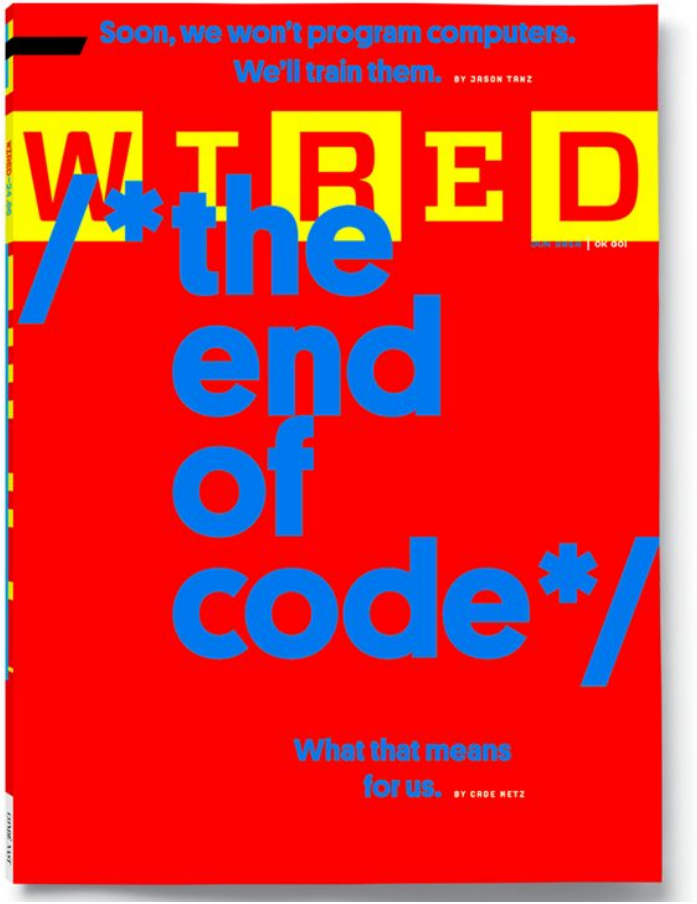
Largest Changes in Life Expectancy
Maldives (+42)
Bhutan (+35)



Learn how to teach yourself new skills.

With the people around you, come up with a list of 3 reasons...

Why ***write code*** to work with data?



With the people around you, come up with a list of 3 reasons (***why not***)...

Why ***not*** *to* write code to work with data?

No shortage of point-and-click tools…

Writing Code to Work with Data

Why

Customizable

Repeatable

Transparent

Scalable

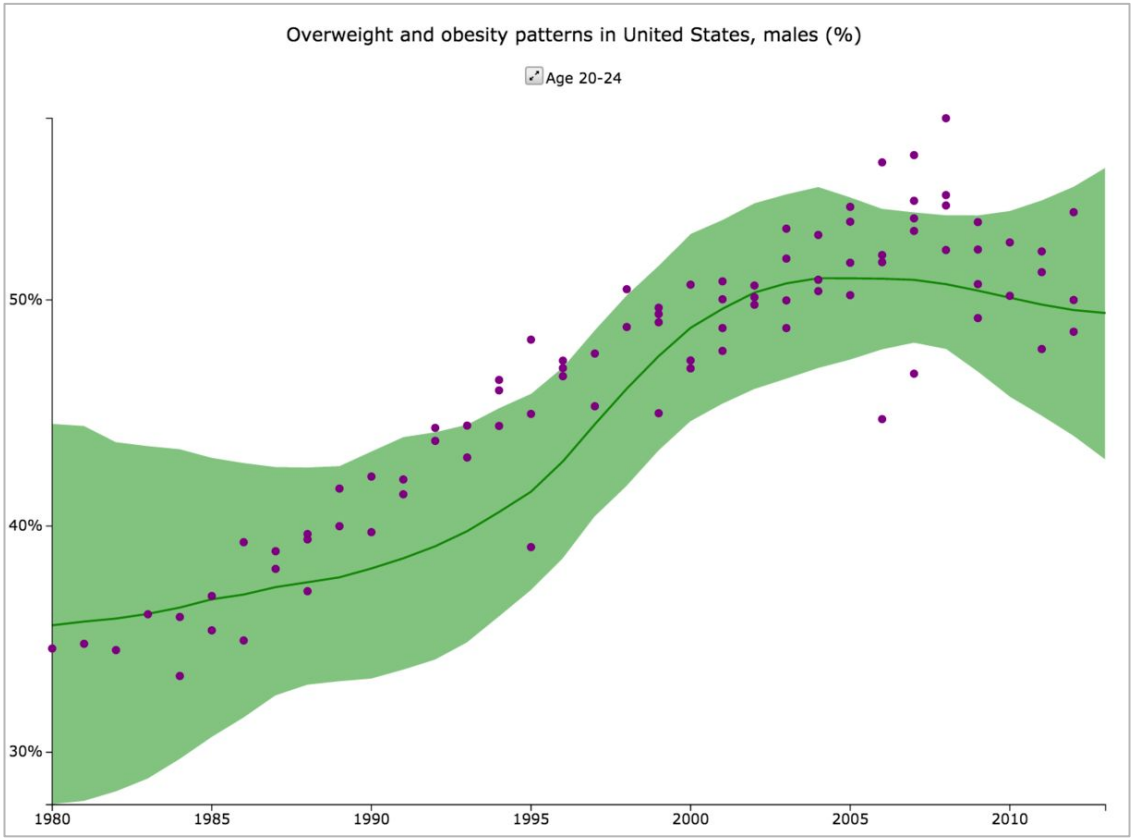
Why Not

Time consuming

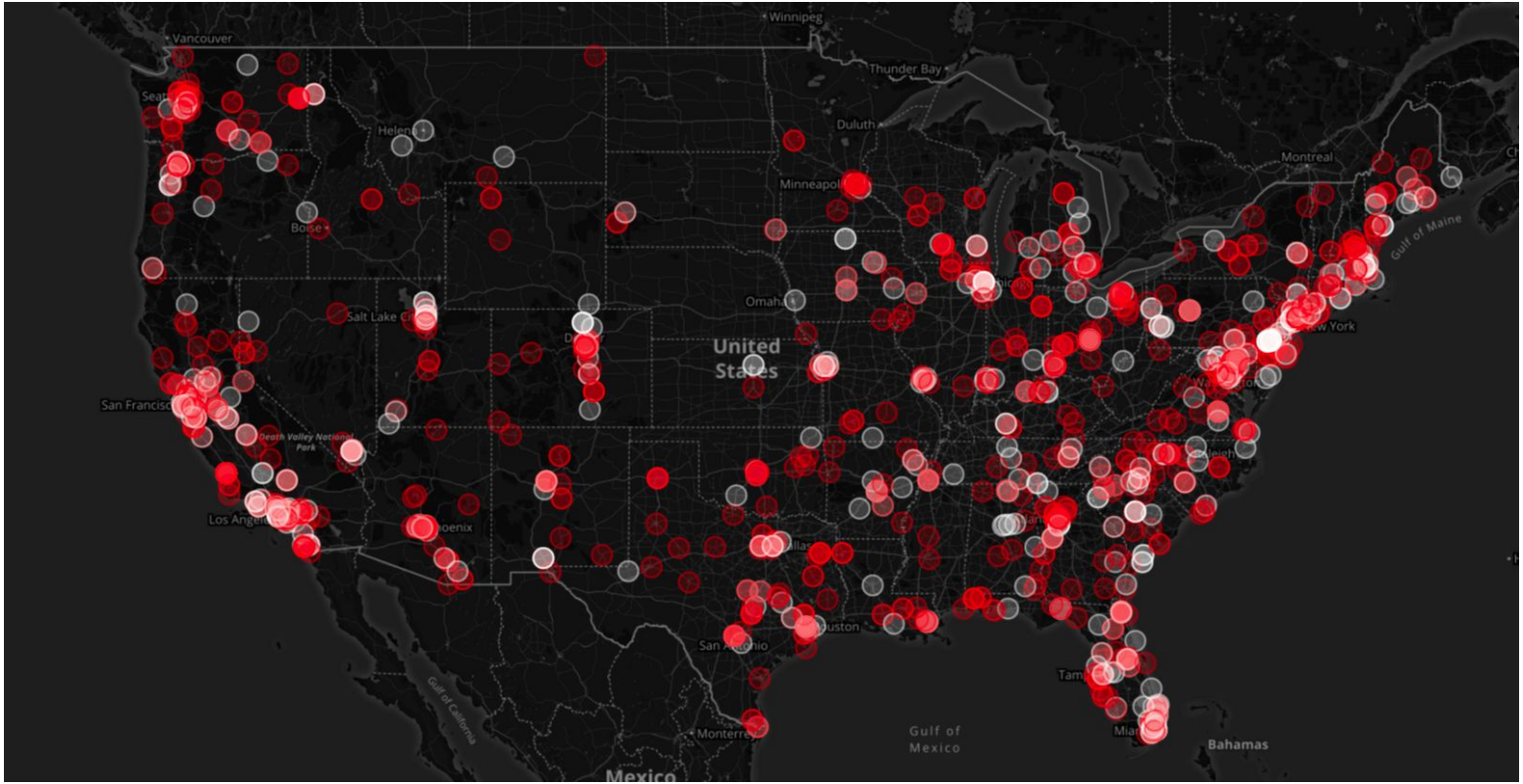
Error prone

Sometimes less clear

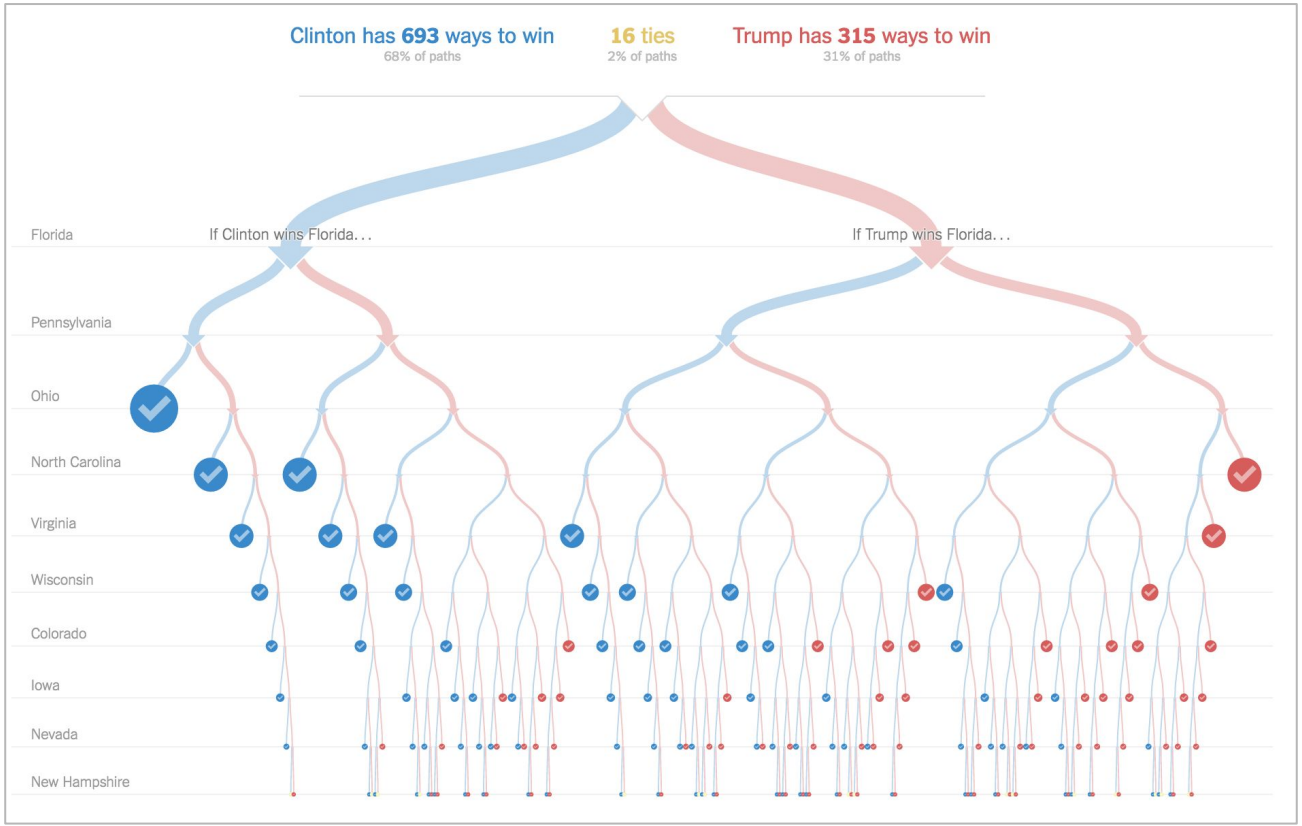
What is the ***purpose*** of writing code to work with data?



Discover Patterns



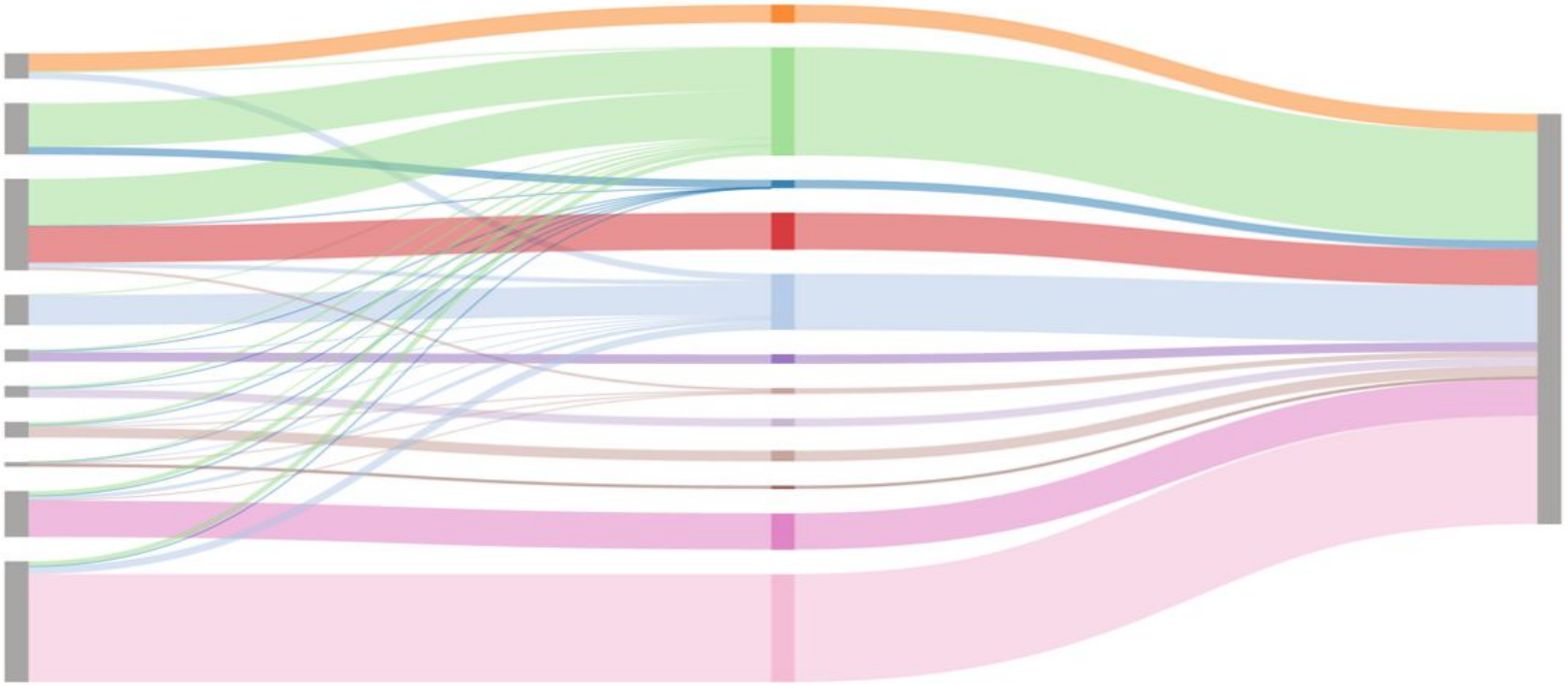
Communicate Insights



Make Predictions

Introductions


Information School
UNIVERSITY of WASHINGTON



Institute for Health Metrics and Evaluation

My Background

Meet your TAs!

Introduce yourself to your neighbors

Polite small talk...

Why you enrolled in this class?

Something outside of class you enjoy?

A topic/field that you would like to apply data/code skills to?

Course Structure

Course Resources

Canvas: used for *submitting* **assignments** and accessing **slides** and **policies**

GitHub**:** where you will *save* **assignments** and access **learning modules**

Slack: how you will **collaborate**, ask **questions**, and see **announcements**

Class and Lab Time

Distinct time for lecture/discussion/activity in each session

Bring your laptop if you have one

Will determine your participation grade

Expectations:

- Participate
- Be **respectful**
- Be **on time**

Assignments

~Weekly individual assignments, (70% of your final grade)

Group project (20% of your final grade)

Require independent learning

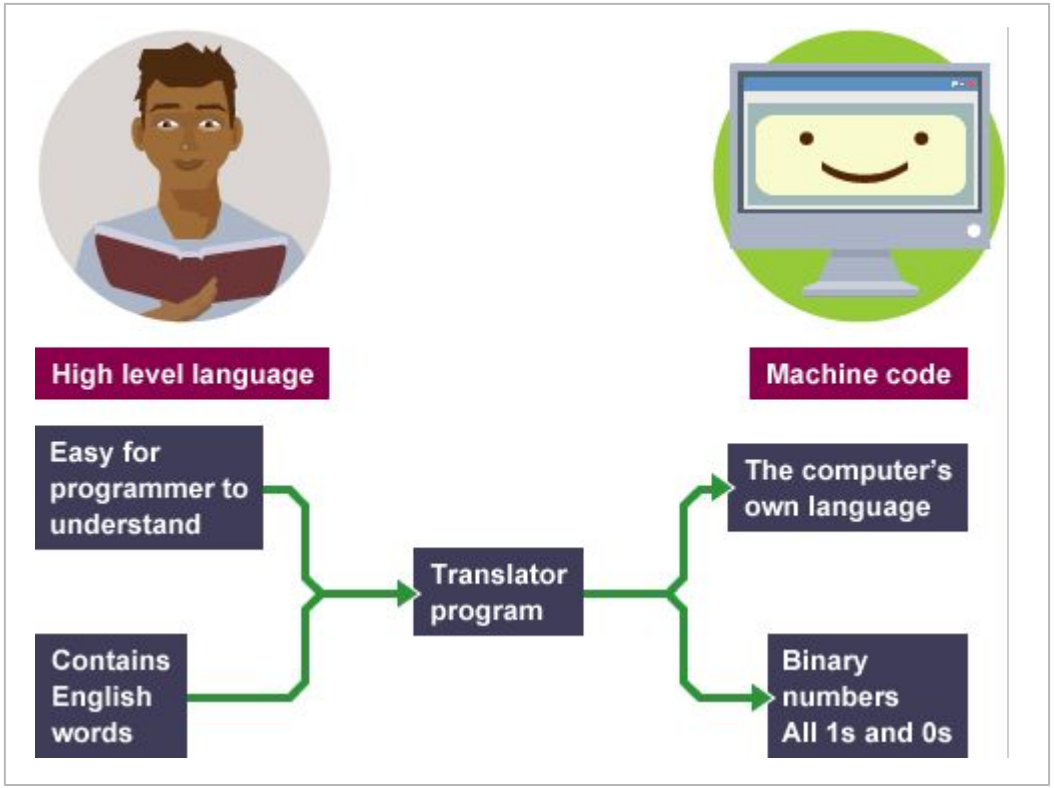
Assignment Policies

Due **before** class

Penalized 10% each 24 hour period, down to 50%

Plagiarism ***will not*** be tolerated

Programming Language Landscape



High v.s. Low level languages

Types of High-Level Languages

Interpreted

Executed *one line at a time*

Less cumbersome/strict

R, Ruby, Python

Compiled

Write *entire* program

Faster

Java, C, Scala

R

Built for managing and analyzing data

Open source

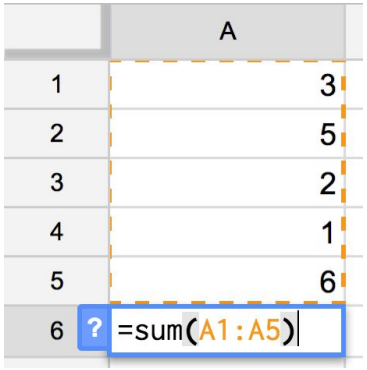
Extraordinarily popular

RStudio is a great ***Integrated Development Environment*** (IDE)

Using The Command-Line

First, a Distinction:

Leveraging Syntax



Interact with software

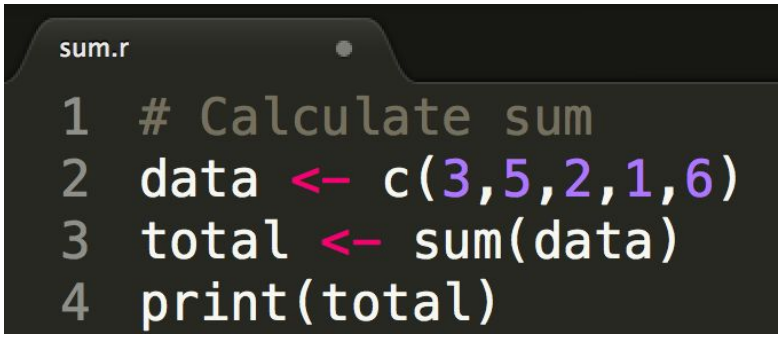
Done once

Abilities are *predefined*

both use syntax!

The line is often blurry!

Writing a Program



Writing software

Reusable

Can define *your own* abilities

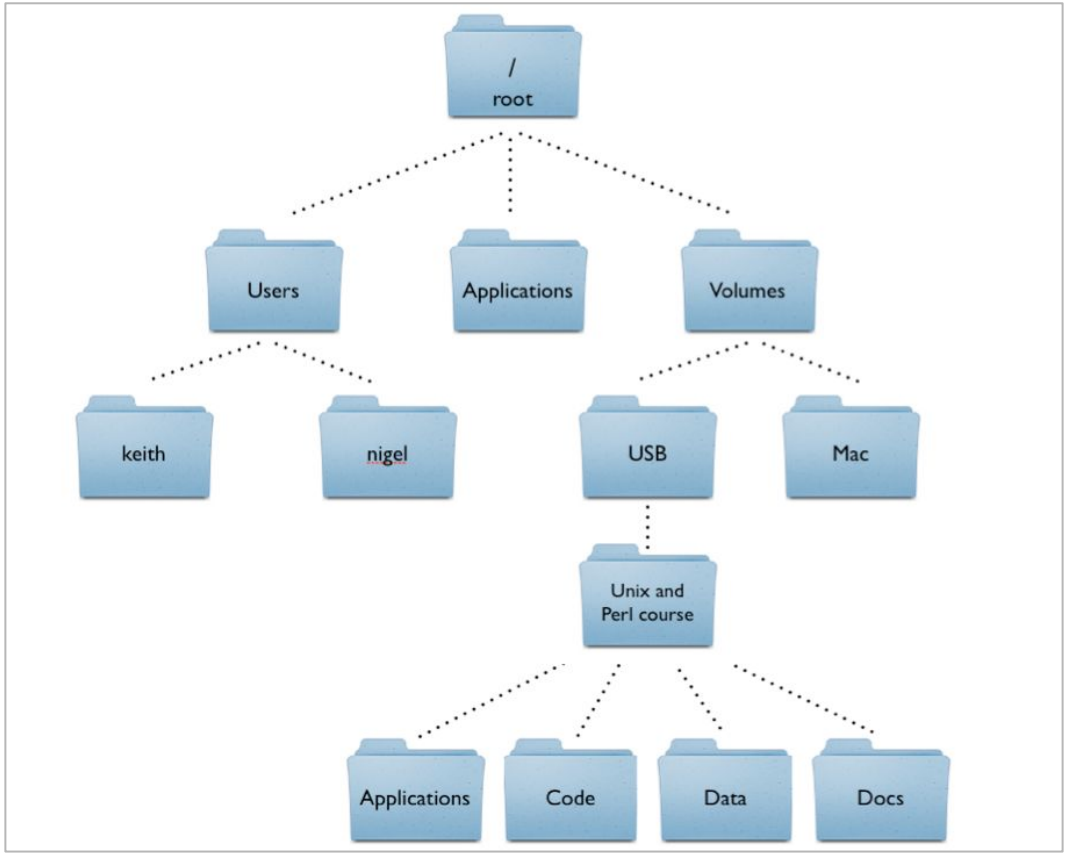
The Command-Line

Text based interface

Providing instructions to your computer

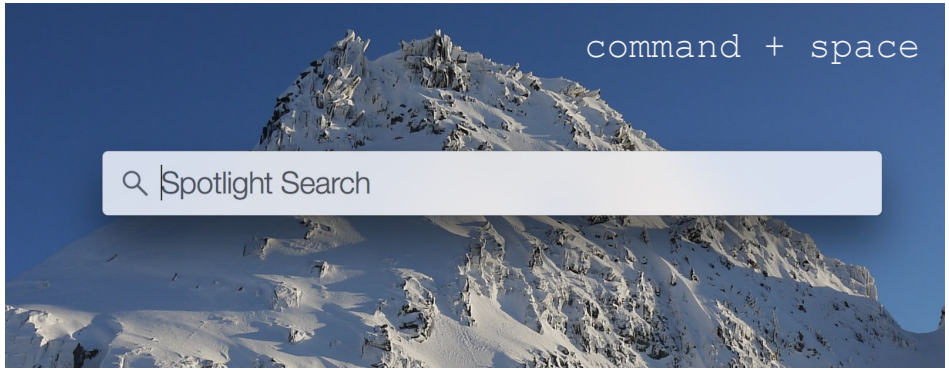
Very useful for directly interacting with programs

We'll use it for interacting with our ***version control*** software

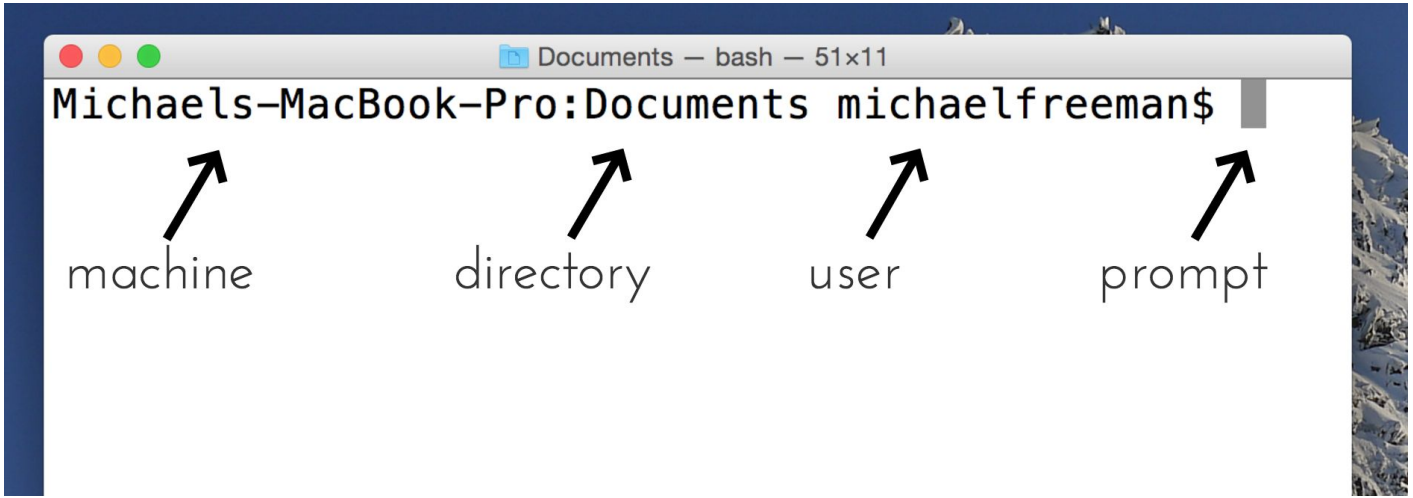


Your file-system is a tree structure

module-2



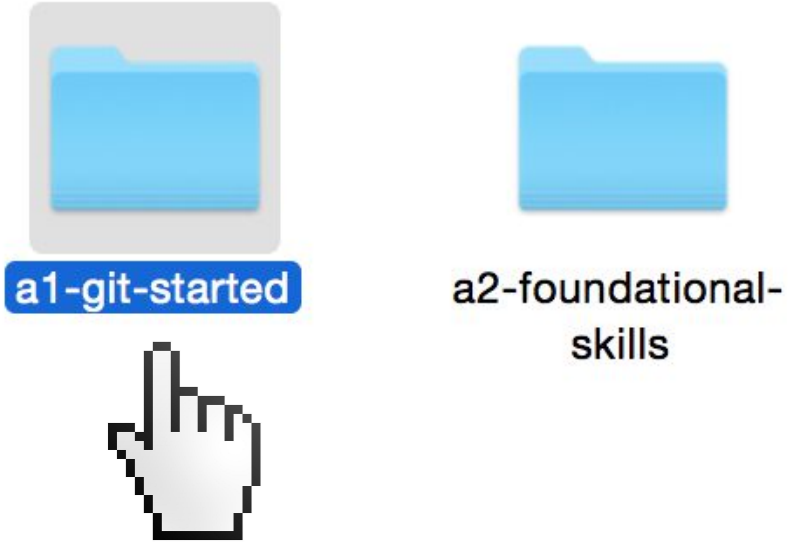
For windows installation, see module-1



Accessing your terminal

module-2

Graphical User Interface (GUI)



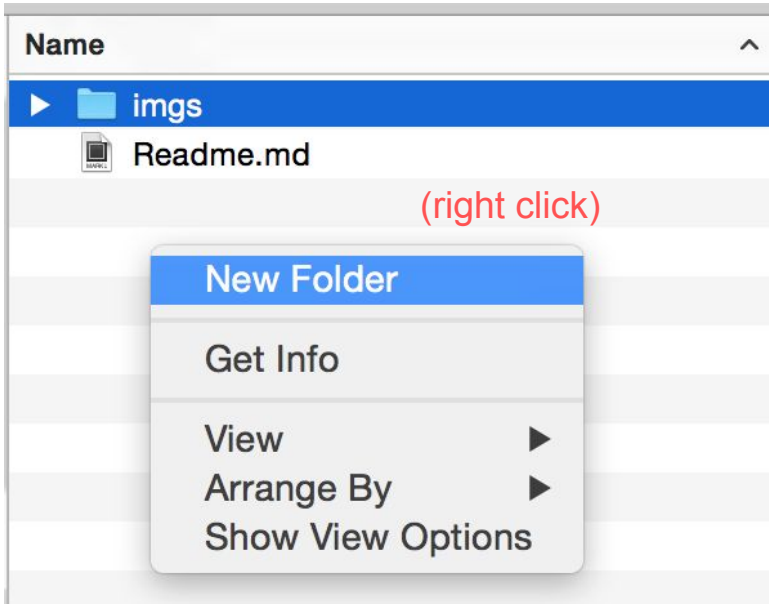
Command-line

Change Directory (cd)

```
cd DIRECTORY-NAME
```

```
cd a1-git-started
```

Graphical User Interface (GUI)



Command-line

Make Directory (mkdir)

```
mkdir NEW-DIRECTORY-NAME
```

```
mkdir imgs
```

Action	Syntax
Copy a file	`cp OLD_FILE NEW_FILE`
Move a file	`mv OLD_FILE NEW_FILE`
Delete a file (careful!)	`rm FILE_NAME`
Create a new file (windows alternative)	`touch FILE_NAME`
Open a file	`open FILENAME (windows: start FILENAME)`
View text of a file	`less FILE_NAME`
See previous commands executed	`history (also hit up arrow)`
View Manual information for a command	`man COMMAND`

More basic commands

module-2

Module-2 exercise-1

Upcoming...

By Tuesday: You should feel comfortable with **modules 0 - 2**

Software Installation (see module-1):

- A text editor
- RStudio
- Git / Git Bash (if you're on a Windows machine)

Sign up for an account on GitHub

Due Thursday, 10/6 (***before class***): a1-git-started