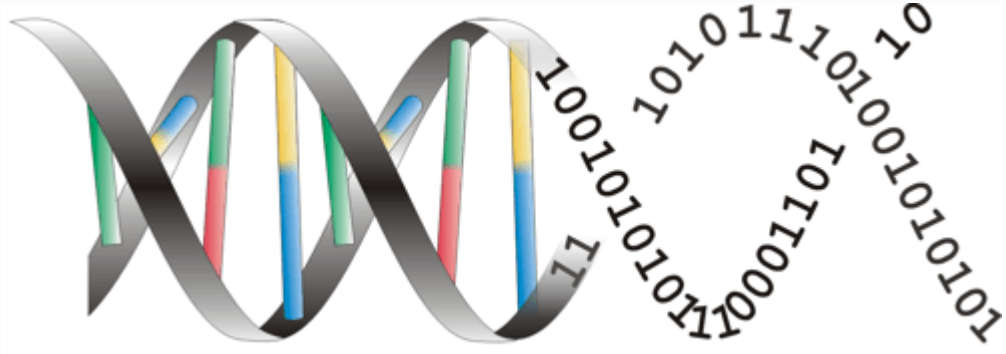
BMS Bootcamp Coding Workshop



Emily Jones

7 September 2017

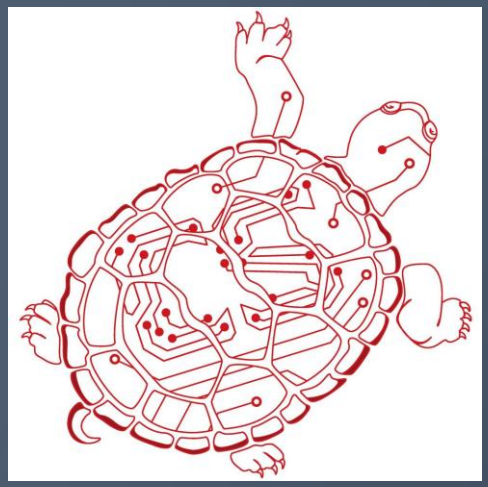
Workshop Outline

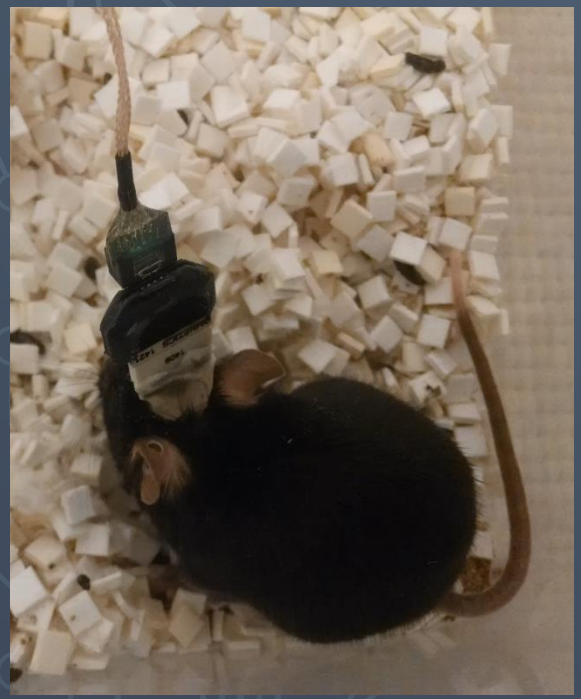
- Why learn to code?
- Coding in practice
- Paper exercise
- Python exercise
- Bioinformatics pipeline
- R Exercise
- Resources

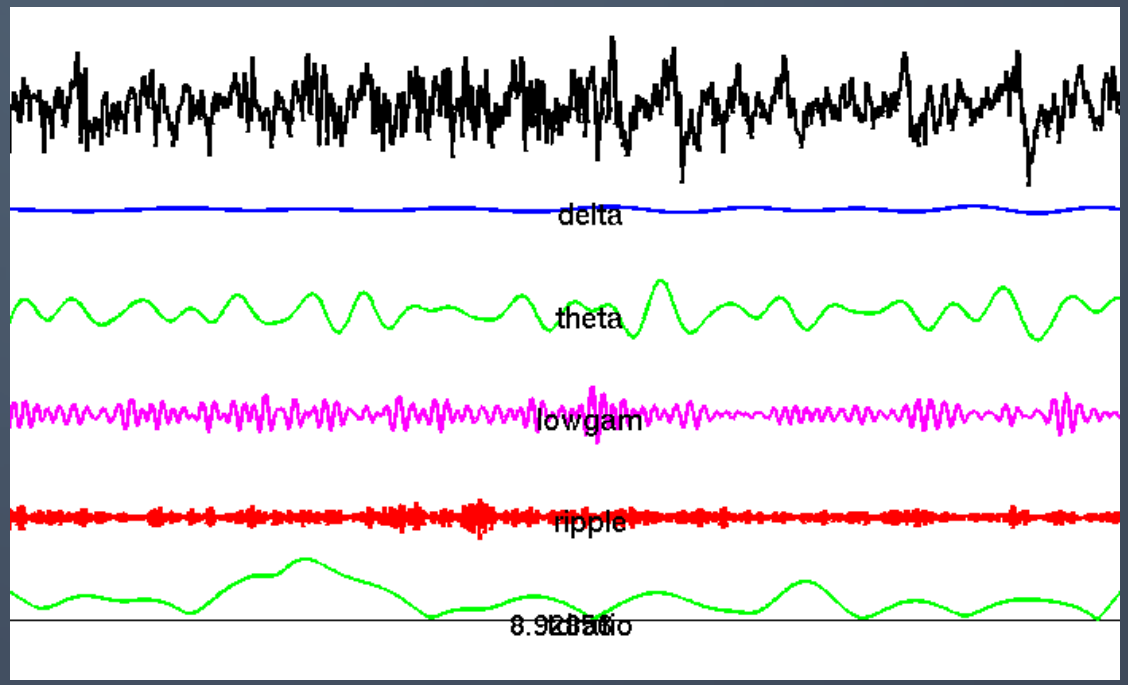
Workshop Outcomes

- Think like a programmer
- Get your feet wet
- Become familiar with terms, concepts, languages, workflows, and resources
- See coding as a useful, attainable skill
- Teach yourself to code after this session

My Background







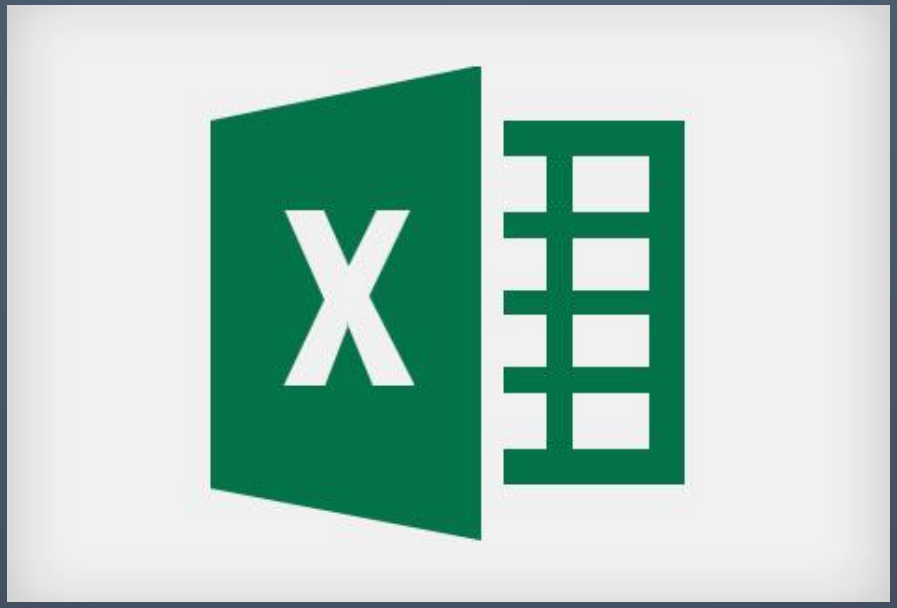
Your Background

- Languages you know:
- How you learned:
- Interests:

You already know how to code

=AVERAGE(A1:A10)

=SQRT(A1:A10)



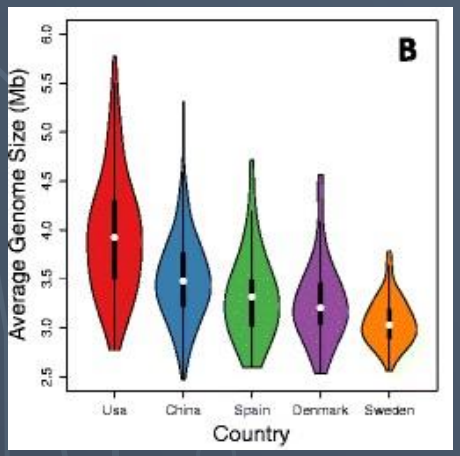
Why learn to code?

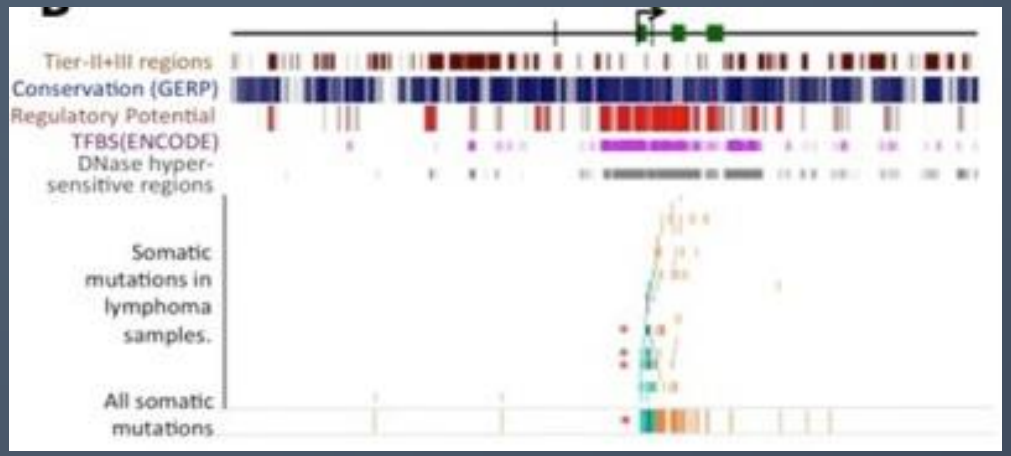
EDUCATION

Computing Has Changed Biology—Biology Education Must Catch Up

Biologists need better computational education so that researchers can benefit from the bioinformatics revolution.

Pavel Pevzner[1]* and Ron Shamir[2]

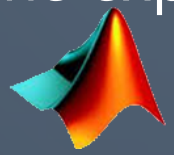




$

- R01: NLM Express Research Grants in Biomedical Informatics
- R21: Exploratory Innovations in Biomedical Computational Science and Technology
- NIH NRSA: NHGRI, NIGMS, NLM
- NSF GRFP

Coding in Practice: Languages

- Scripting:  Python  Ruby  Perl
  - e.g. read information from a text file of GWAS results
- Statistics: 
  - e.g. cluster gene expression by similarity
- Signal processing:  MATLAB
  - e.g. filter & process raw EEG data

Choosing a language:

1. Ease of use/your own knowledge
2. Packages
3. Community

Using a language:

1. Install the language
2. Install the IDE
3. Find & install a package

Coding in Practice: A Simple Script, Start to Finish

Sample problem: You have run a microarray on your sample tissue and want to compare expression of different genes.
Learning how to write the script is similar to learning how to do the assay.

1. Choose your algorithm: what assay should I use?
   - Expression clustering
   - Microarray
2. Choose your language: which protocol should I use?
   - R, Hierarchical clustering (hclust)
   - Tissue IHC (Abcam kit)
3. Prototyping: test your assay to make sure it works
   - Run your script on fake data
   - Run microarray with positive & negative controls
4. Debugging: analyze your results before you start the next round
   - Don't wait until your script is finished to start testing the output
   - Don't run an entire paper's worth of microarrays just to discover the antibody doesn't work
5. Keep records
   - Document your code, record bugs and results in your lab notebook
   - Record each sample, assay, and results in your lab notebook

FizzBuzz Test

With a partner, in pseudocode, describe a function called "FizzBuzz." This function does the following:

1. Takes a whole number "x"

```
function FizzBuzz(x):
    _________
```

2. Prints all the numbers between 1 and x in order, BUT

```
for i = a to b
    _________
```

2. If x is a multiple of 3, then print "Fizz" instead of the number AND
3. If x is a multiple of 5, then print "Buzz" instead of the number

```
if (condition) then
    _________
```

```
if (condition) then
    _________
else
    _________
```

FizzBuzz Test: Tracing

Swap papers with another pair.
Let x = 20. Write down what their function outputs, talking through each iteration of the loop as you go.

Pseudocode:

```
function FizzBuzz(x):
    for i = 1 to x
        if (i is divisible by 3 OR by 5) then
            if (i is divisible by 3) then
                print "Fizz"
            if (i is divisible by 5) then
                print "Buzz"
        else
            print i
```

Trace:

1	11
2	Fizz
Fizz	13
4	14
Buzz	FizzBuzz
Fizz	16
7	17
8	Fizz
Fizz	19
Buzz	Buzz

Let's Learn Python in 10 Minutes Flat

Go to repl.it and select Python3

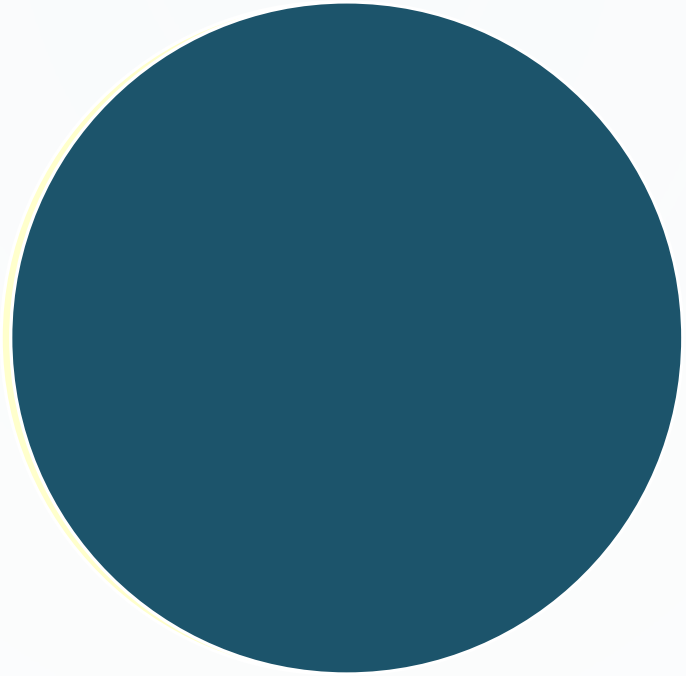
Let's Learn Python in 10 Minutes Flat

```
#prints the bridge of "Hey Ya"
#input: string shakeitlike
def heyya(shakeitlike):
for i in range(0,4):
        if i%4==3:                          #every fourth line
            print("shake it like " + shakeitlike)
        else:                               #all other lines
            print("shake it, shake shake it")
    return "written by the BMS Bootcamp 2016 Class"
```

Topics we covered:

1. Function
2. Print
3. Loop
4. Zero-indexing
5. Documentation
6. Tabs
7. Variables
8. If..Then…
9. Parameters
10. Types
11. Return

10 Minute Break



Collaborating Through Code

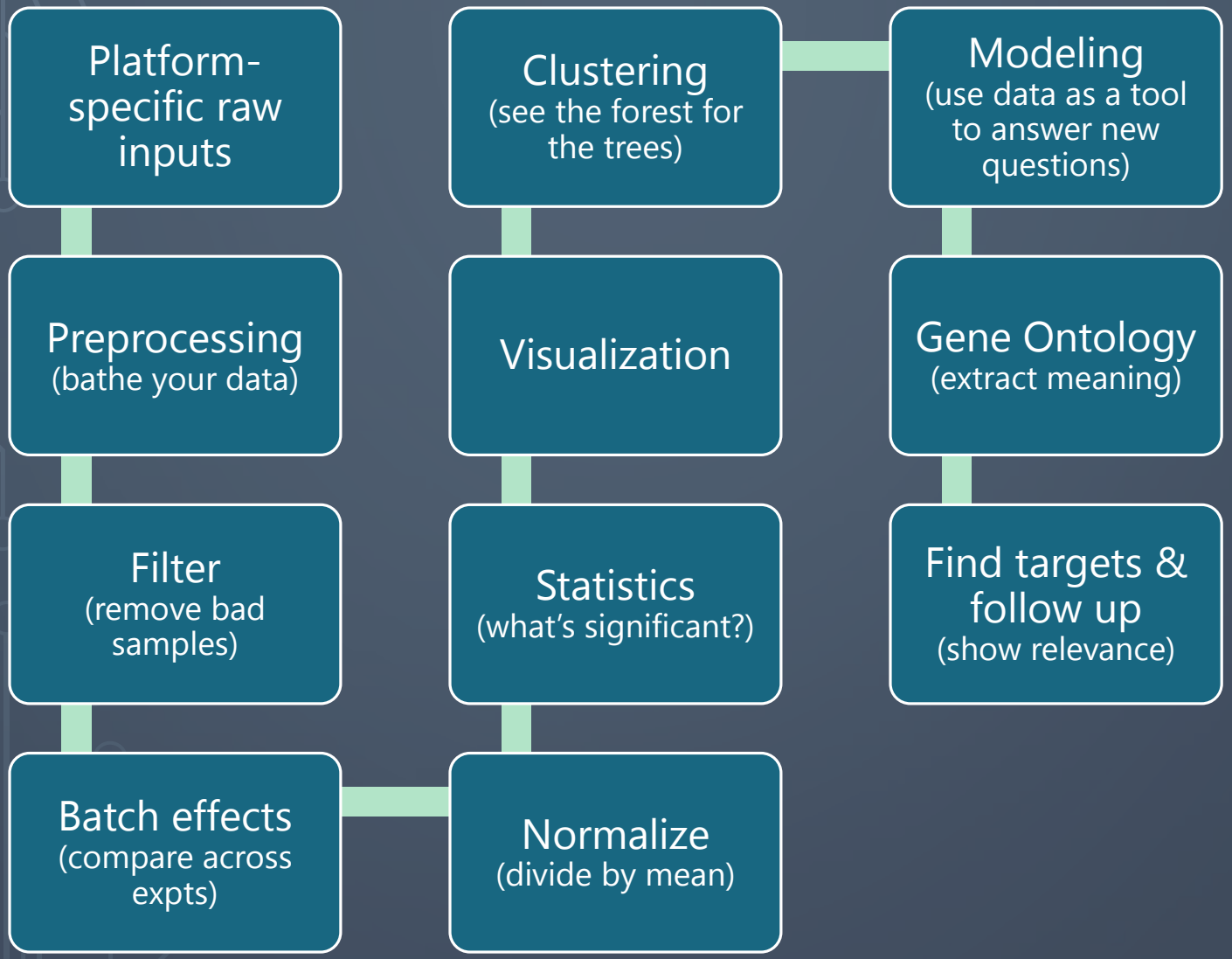
Bad Code:

```
thing1 = makestuff(10)
thing2 = makestuff(20)
thing3 = makestuff(30)
if(thing2>thing1) {print(thing2)}
if(thing3>thing2) {print(thing3)}
```

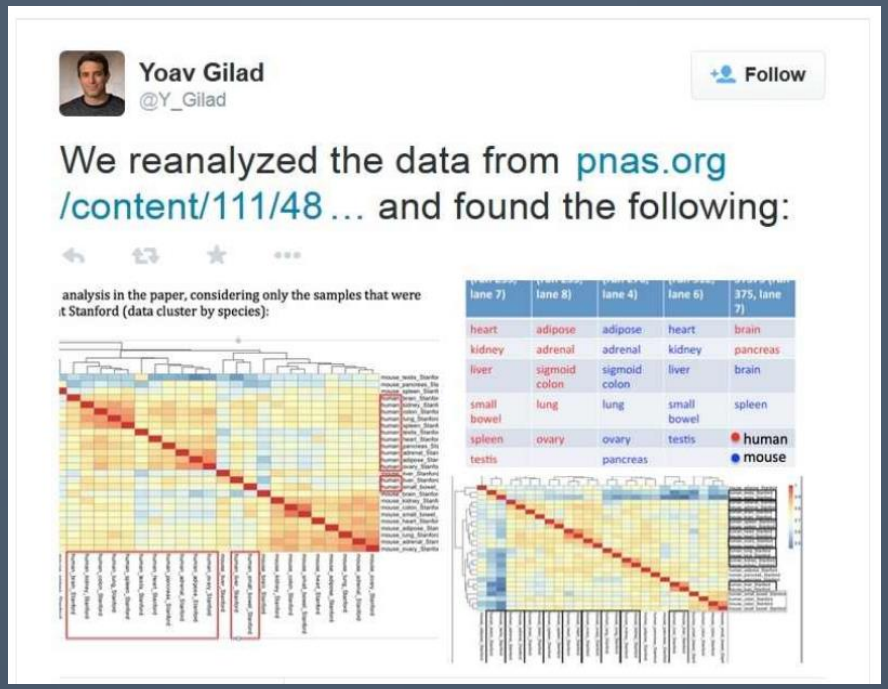
Good Code:

1. Useful variable & function names
2. Modular & reusable (don't repeat yourself)
3. Readable
4. Documented & reproducible

Bioinformatics Processing Pipeline

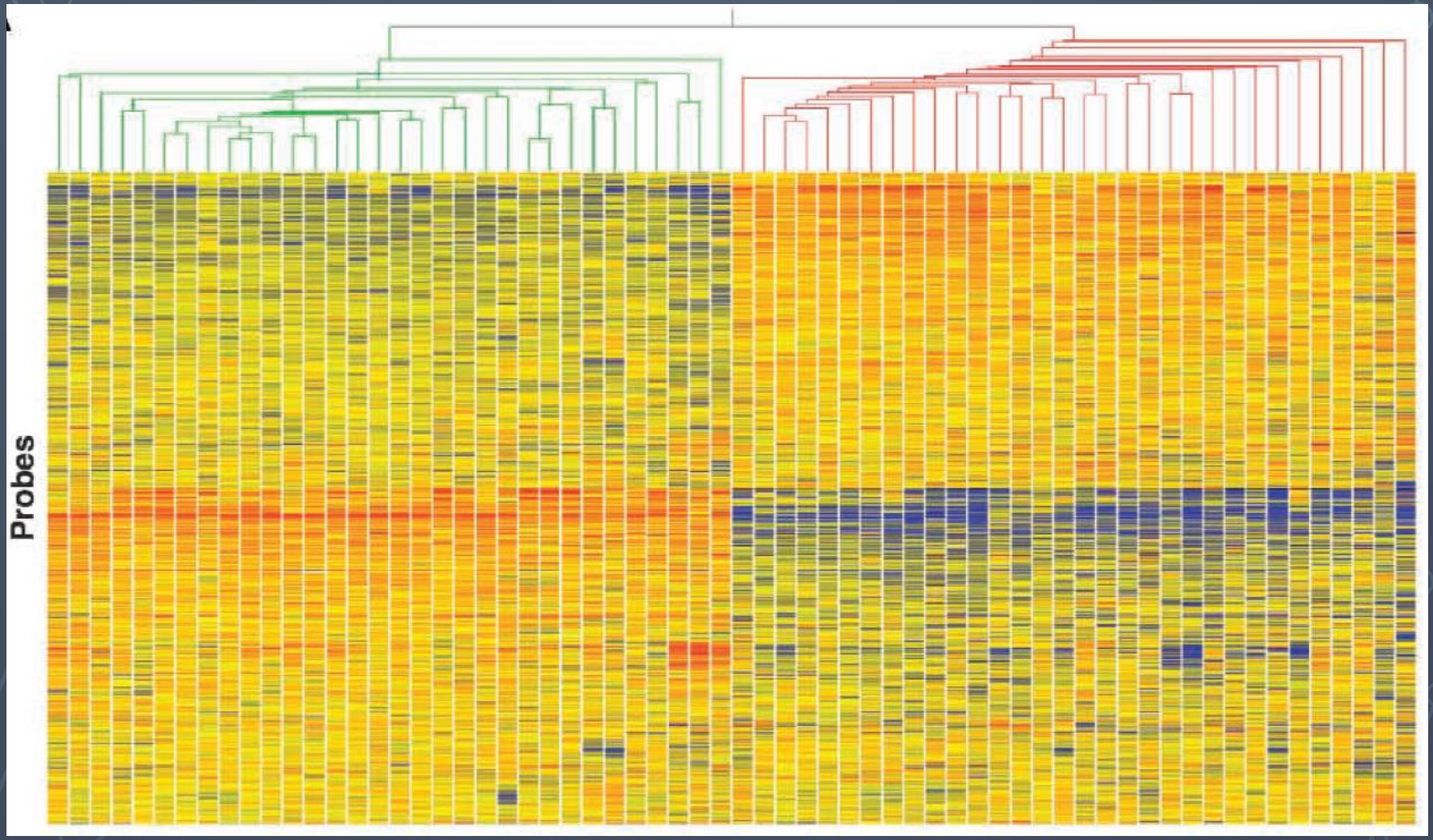


Good Processing is Important



"These effects can arise from variations in the way different batches of samples are handled, which can affect the data and be mistaken for biological effects. Gilad concluded that the analysis involved five different sequencing runs to analyse RNA. *One of those runs included both mouse and human tissues, but the other four contained samples from only mouse or only human tissues.* He argues that any differences in the runs themselves could have exaggerated the differences between the two species. According to Gilad's reanalysis…*removing possible batch effects from the data reverses the conclusion of the PNAS paper* — such that the gene-expression data cluster by tissue and not by species."

Let's Learn R in 30 Minutes Flat



Pop Quiz!

www.socrative.com

Student login, room: BMSBOOTCAMP

Campus Resources

Library: https://www.library.ucsf.edu/help/classes

Cell Hackers: https://www.facebook.com/cellhackers

Bioinformatics Core: http://cores.ucsf.edu/bioinformatics-analysis.html

BMI Program Courses: http://bioinformatics.ucsf.edu/degree-program/courses

Mini-courses:

- Intro to Bioinformatics and Computer Programming for Biologists
- Practical Bioinformatics with Programming
- Scientific Software Development
- …or others, depending on the year

Programming and Pizza: Python/R

http://bit.ly/programmingandpizza

Thursday, September 21, 4-6pm

The Hub, Mission Hall 1302

Resources

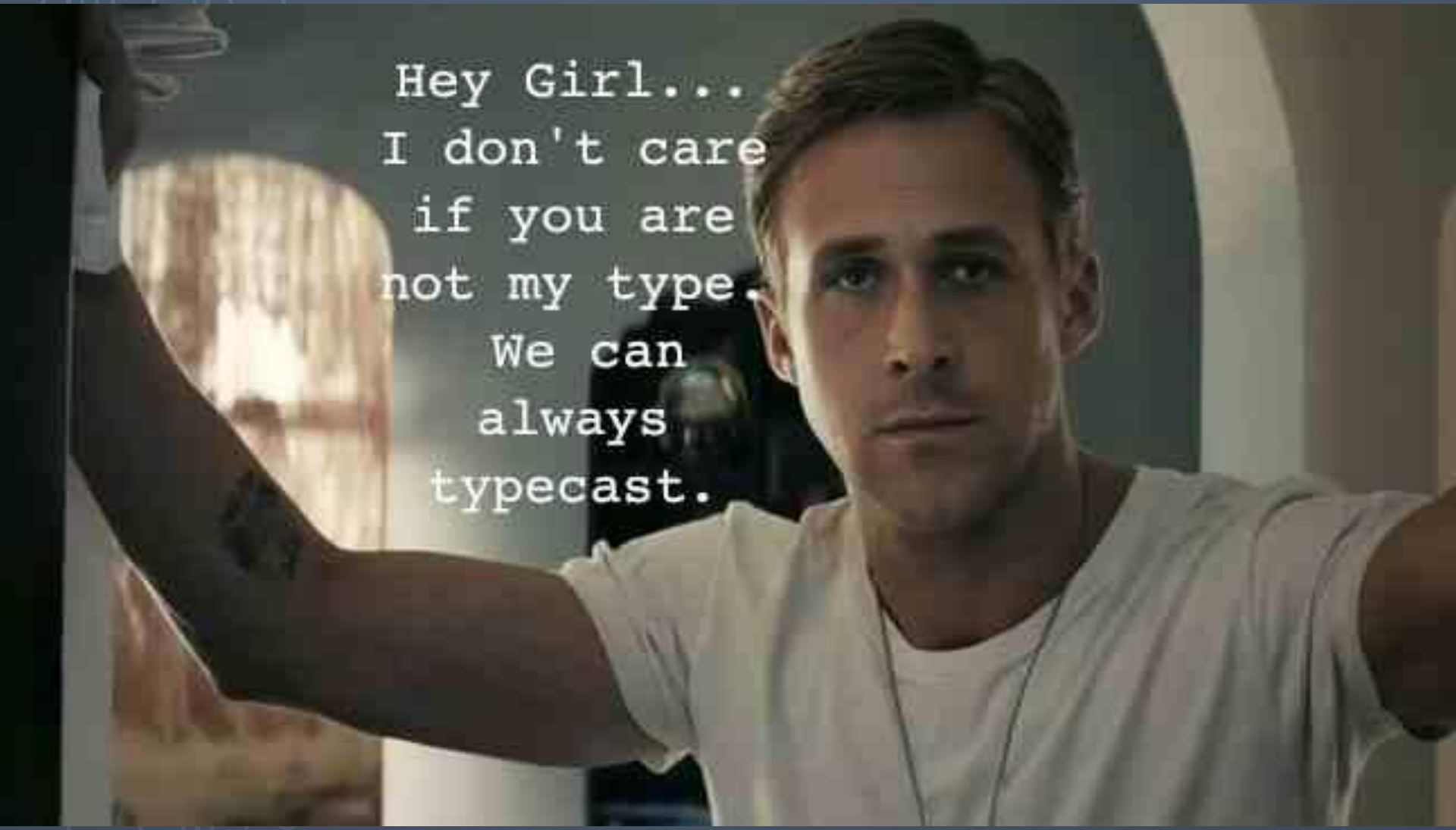
Handout online, including…

- Challenges to expand what we've learned here
- Campus resources
- General
- Python guides & courses
- R guides & courses
- Package managers (if you want to get fancy)

Also on the CLE website: slide deck, heyya.py, R markdown
Contact me at emily.jones@ucsf.edu

Workshop Outcomes

- Think like a programmer
- Get your feet wet
- Become familiar with terms, concepts, languages, workflows, and resources
- See coding as a useful, attainable skill
- Teach yourself to code after this session



programmerryangosling.tumblr.com

Next Steps

1. Think about what you'd like to accomplish
2. Pick 1 language
3. Pick 1 introductory resource
4. Put it on your calendar
5. Find other students who also want to learn and try a challenge with them
6. BONUS: pick 1 course