STAT 215A Fall 2021 Week 2

Omer Ronen

Announcements

- Lab 0 example code available at gsi repo
- Make sure your **stat-215-a** repo is private
- Lab 1 will be released at the end of today's discussion section. **Due Thursday 16 at 11:59pm**

GitHub repos I have access to:

100shpaik
aashen12
albertqu
andleb
andreamirandagz
austinvzane
baturalpyalcinn
cz-ye
ellawang55
floricaconstantine
han9704
Harry970804
hysk79
ias5
ilinabg
ishaans99
jbbutler
jeremy-goldwasser
kferger320
licong-lin
Mark-Oussoren
mbowen97
n-mehandru
NataliaSV
PrejudiceDDH
salwanbutrus
ssaxena00
tiffanyding
tor-n
wtorous
xinzhou97
Yax-H
yuhaod

If you don’t see your name, please email / Slack me.

Today's outline

- `here()` and `across()`
- Some practice with Tidyverse
- Workflows
- Lab 1 Introduction

A quick poll on last week’s discussion

Did you learn something new in last week’s discussion?

sli.do

Event code #: 87666

continued...

dplyr: `across()`

- `across()` supplants some of the “scoped verbs” that end with `_if()`, `_at()`, and `_all()`.
- Two primary arguments:
  - `.cols`: selects columns to operate one.
  - `.fns`: functions (can be more than one) to apply to the selected columns.

With `across()`:

```
> iris %>%
+   mutate(across(.cols = contains("Sepal"), ~ 2 * .x)) %>%
+   head()
```

With `mutate_at()`:

```
> iris %>%
+   mutate_at(vars(contains("Sepal")), list(~ 2 * .)) %>%
+   head()
```

dplyr: `across()` and `summarize()`

- `across()` works particularly nicely with `summarize()`

```
> iris %>%
+   summarize(
+     across(where(is.numeric), mean),
+     across(where(is.factor), nlevels),
+   )
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1     5.843333    3.057333        3.758    1.199333       3
```

- There are some examples where the `across()` syntax is not as nice:

```
# mutate + across
mtcars %>% mutate_all(mean)
# ->
mtcars %>% mutate(across(everything(), mean))
```

- Learn more: https://dplyr.tidyverse.org/articles/colwise.html


here: find your PATH!
seriously, just here.
"/NOT/HERE"
here!
IT'S ALL RIGHT HERE!

here()

here()

- here is a very simple package that **increases reproducibility**
- When you run library(here) it checks the current working directory (i.e. whatever getwd() returns) for:
  - A file named .here
  - An RStudio project: foo.Rproj
  - An R package: DESCRIPTION
  - A git repo: git
  - Some others
- If it doesn't find any of those, it moves up to the parent directory and starts over.

`here()` example

PWD is `week2`

`here()` starts at the git repo top-level directory

```
> getwd()
[1] "/home/james/school/215a/stat-215a-fall-2020/week2"
> library(here)
here() starts at /home/james/school/215a/stat-215a-fall-2020
> here()
[1] "/home/james/school/215a/stat-215a-fall-2020"
> here("week2", "data", "mtcars.rds")
[1] "/home/james/school/215a/stat-215a-fall-2020/week2/data/mtcars.rds"
> mtcars2 <- readRDS(here("week2", "data", "mtcars.rds"))
> head(mtcars2)
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
```

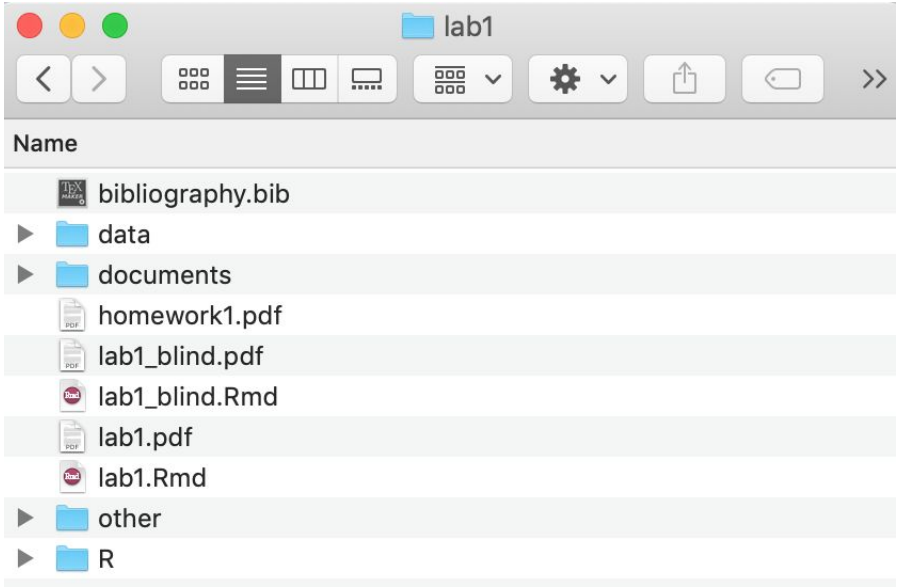
`here()` concatenates the path

Let’s get some hands-on practice

- About Gapminder: https://www.gapminder.org/about/
- Resources for this tutorial:
  - ggplot: http://swcarpentry.github.io/r-novice-gapminder/08-plot-ggplot2/
  - dplyr: http://swcarpentry.github.io/r-novice-gapminder/13-dplyr/
- See **lab_gapminder.Rmd** in the week2 folder on my GitHub

Project file structure

- **data/:** store raw and processed data
- **documents/:** store relevant papers, instructions, meeting notes, etc.
- **R/:** store R code, utility functions, scripts
- **other/:** miscellaneous



Project File Structure

R/

- **load.R** – file containing function(s) for reading in the data

```
> loadData(path_to_data)
```

- **clean.R** – file containing function(x) for cleaning the loaded data

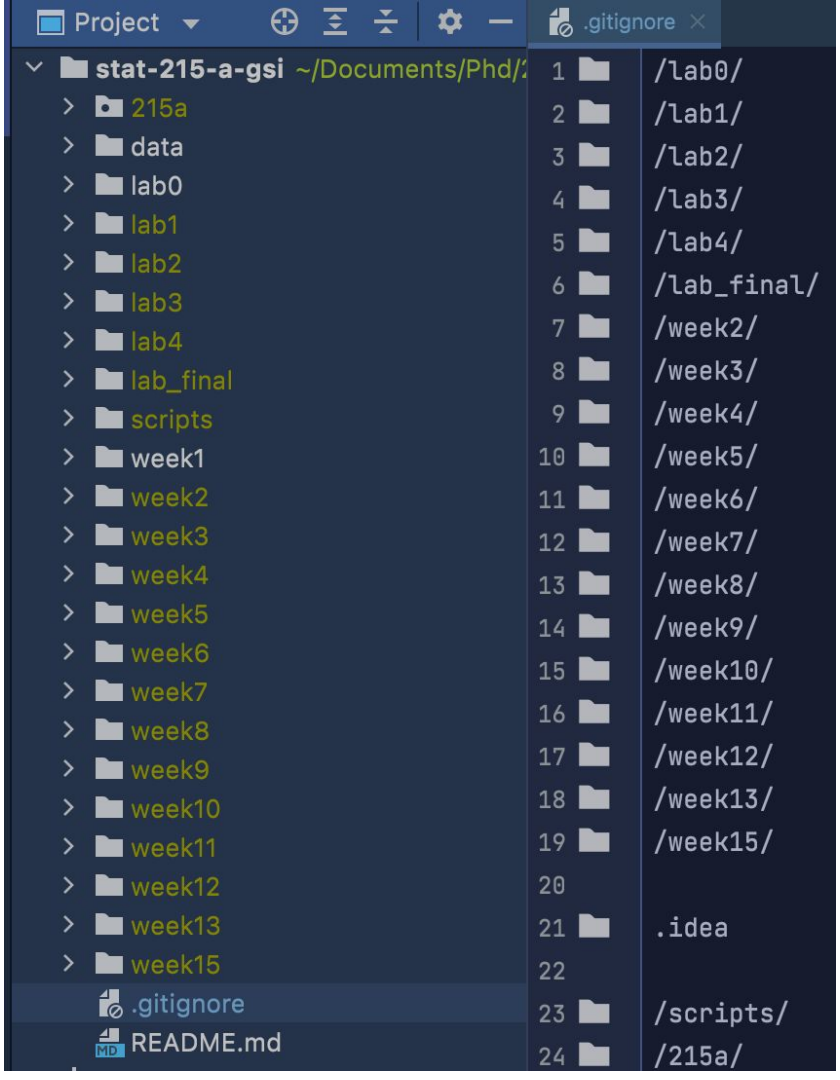
```
> cleanData(loaded_data)
```

data/

- Contains datasets
- Not uploaded to GitHub (can automate this using .gitignore)

.gitignore

The `.gitignore` file is a text file that tells Git which files or folders to ignore in a project.




knitr
www.rstudio.com

Project File Structure

lab1.Rmd – your final report combining code (not printed in the output) and text/narrative

- Should be written like a paper; focus on communicating well

lab1.pdf – pdf output from lab1.Rmd

lab1_blind.Rmd – same as lab1.Rmd but without name

lab1_blind.pdf – pdf output from lab1_blind.Rmd

explore.Rmd (optional) – a separate .Rmd file that contains your exploratory code and figures

- A useful place for exploring the data and saving avenues of exploration that you don't necessarily want to include in your final report

bibliography.bib (optional) – a .bib file for easy citations within the lab reports

homework1.pdf – can be submitted electronically or in person at Friday lab section

Workflow: General Tips

Make code readable

- Be kind to both your peer reviewer and your future self

Keep your code modular – write functions

- Separate your functions from your analysis file (lab1.Rmd) and store them in R/
- In doing so, you create a bank of useful functions that you can load into any analysis script for your project (or future projects)
  - To load in a single file:

    ```
    > source("./R/filename.R")
    ```

  - To load in all files in the R/ directory:

    ```
    > library(R.utils)
    > sourceDirectory("./R/", modifiedOnly = F, recursive = F)
    ```

- Group together related functions in the same .R script

  (e.g. put all data cleaning functions in clean.R)

Workflow: General Tips

Documentation

- Write lots of comments in your code and ask yourself: why are you writing this particular piece of code?
- Document functions (think about the R help pages)
  - Always add comments section immediately below the function definition line
  - What does this function do?
  - Describe the inputs and outputs

```
CalculateSampleCovariance <- function(x, y, verbose = TRUE) {
  # Computes the sample covariance between two vectors.
  # Args:
  #   x: One of two vectors whose sample covariance is to be calculated.
  #   y: The other vector. x and y must have the same length, greater than one,
  #      with no missing values.
  #   verbose: If TRUE, prints sample covariance; if not, not. Default is TRUE.
  # Returns:
  #   The sample covariance between x and y.
...
}
```

Workflow: General Tips

Test your code

- Write tests to make sure your functions are doing the right thing
- Write these tests as you go

Don’t Repeat Yourself (DRY)

- If you find yourself copying and pasting similar lines of code, write a reusable function instead

Establish consistencies – follow Google R Style Guide

Workflow: Code Style

Follow Google's R Style Guide when writing code

(See https://google.github.io/styleguide/Rguide.xml and part I Analyses of https://style.tidyverse.org/syntax.html#object-names)

Variable names

- All lowercase
- Separate words by "." or "_" (be consistent with the one you choose)

Good: `avg.tmp, avg_tmp`

Bad: `AvgTmp`

Workflow: Code Style

Function names

- Camel-case
- Make function names verbs

Good: `CalculateAvgClicks, calculateAvgClicks`

Bad: `calculate_avg_clicks, calculuate.avg.clicks`

Line Length: maximum length of 80 characters

- Go to: Preferences □ code □ display □ check show margin and set margin column = 80

Indentation: When indenting your code, use two spaces (rather than tabs)

Workflow: Code Style

Spacing

- Place spaces around all binary operators (=, +, –, <–, etc.)
- Always put a space after a comma, never before, just like in regular English

Good:
```
df.prior <- df[df$days.from.opt < 0, "campaign.id"]
x[, 1]
```

Bad:
```
calculate_avg_clicks, calculuate.avg.clicks
x[,1], x[ , 1]
```

Line Length: maximum length of 80 characters

- Go to: Preferences □ code □ display □ check show margin and set margin column = 80

Indentation: When indenting your code, use two spaces (rather than tabs)

Workflow: Code Style

Assignment

- Use `<-` instead of `=`

Curly Braces

- An opening curly brace should never go on its own line; a closing curly brace should always go on its own line
- Always begin the body of a block on a new line

Good:

```
if (x > 0) {
  print(x)
}
```

Bad:

```
if (x > 0) print(x)
```

Workflow: Code Style

Most importantly, BE CONSISTENT

Lab 1
Redwood Trees

Due: Thurs,
Sep 16 @ 11:59pm

Lab 1 Goals

Data cleaning

Exploratory Data Analysis and Visualization

Lab 1 Redwood Introduction

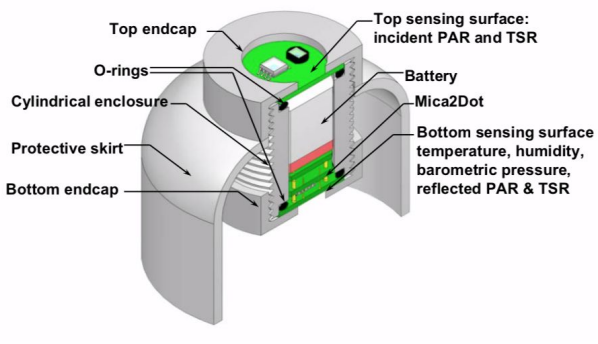


Figure 2: Sensor node and packaging

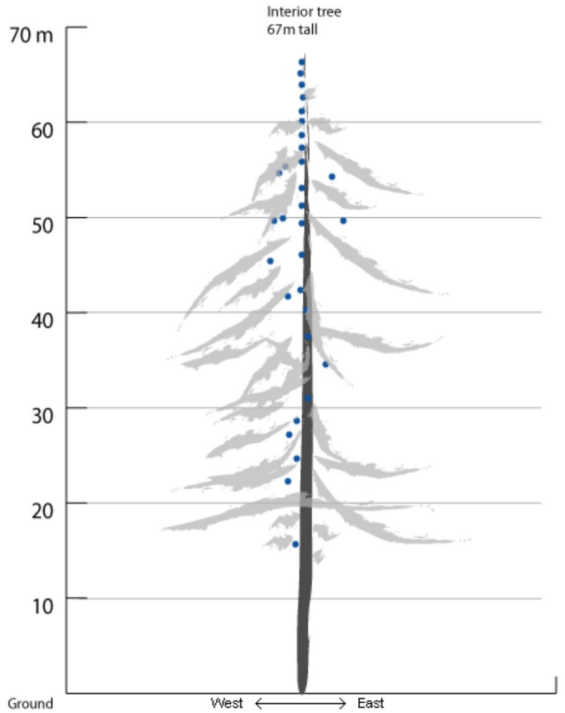


Figure 1: The placement of nodes within the tree

Lab 1 Introduction

- Read the paper carefully **sensys05-TollePolastreEtAl-redwoods.pdf** in the **lab1/lab1_template/documents** folder on GitHub.
- The **lab1/lab1_template** folder will also contain a template to follow when putting together your lab as well as loading and cleaning functions that you may use/fill in
- The `exploration.Rmd` file was put together by Rebecca, a previous GSI, to get you started looking at the data, but you cannot use these plots as part of your lab report
- Do **not** push this `exploration.Rmd` file (or your own explore files) or the data folder to your stat-215-a repo
    - Can easily do this with `.gitignore` file

Collaboration Policy

- You are allowed to discuss **ideas** with others, but you must submit your own report
- Do not share code or copy/paste any part of the writeup
- If you do discuss ideas with others, be sure to acknowledge these students in your report

Lab 1 Rubric

Redwood Tree Lab (~60 points)

- Readability and grammar
- Readability of code (+ comments)
  - Follow Google's R Style Guide (a slight modification of the Tidyverse Style Guide)
- Reproducibility of report
  - I should be able to pull your `lab1/` folder from GitHub, manually add the `data/` folder, open `lab1.Rmd`, click knit, and get the same .pdf file as you.
- Data cleaning (description and validity)
  - Describe *any* problems/inconsistencies you see with the data, how you cleaned the data, and why you cleaned the data in that way
- Three findings (creativeness, interestingness, and quality of figure)
  - Fix titles, axis and legend titles, choose appropriate color schemes, adjust size of figure
- Graphical critique
- Figures that are not for the findings (relevance and quality)
- Overall quality and level of detail of report
  - Attempts to incorporate domain information (from the paper) and place your analysis in the domain context

Homework – Some Basic Statistics (8 points)

Start Early!!!!!!!!