Reproducible Research Notes

Coursera Course by John Hopkins University

INSTRUCTORS: Dr. Jeff Leek, Dr. Roger D. Peng, Dr. Brian Caffo

Contents

Intro **3**
- Course Description 3
- Course Book 3
- What is Reproduciblity about? 3

Concepts, Ideas, & Structure **4**
- Concepts and Ideas (Part 1) 4
  - Replication 4
  - Why Do We Need Reproducible Research? 4
  - Example: Reproducible Air Pollution and Health Research 4
- Concepts and Ideas (Part 2) 5
  - Research Pipeline 5
  - Recent Developments in Reproducible Research 5
  - What do We Need for Reproducible Research? 5
  - Challenges 6
- Concepts and Ideas (Part 3) 6
  - Literate (Statistical) Programming 6
  - Sweave 7
  - knitr 7
  - Summary 8
- Scipting Your Analysis 8
- Structure of a Data Analysis (Part 1): Defining to Cleaning 8
  - Defining a question 9
  - Define the ideal data set 9
  - Determine what data you can access 10
  - Obtain the data 10
  - Clean the data 10
- Structure of a Data Analysis (Part 2): Exploring to Creating Reproducible Code 11
  - Subsampling our data into test & train sets 11
  - Exploratory data analysis 11
  - Statistical prediction/modeling 17
  - Interpret results 18
  - Challenge Results 19
  - Synthesize/write-up results 19
- Organizing Your Analysis 20

Types of Data Analysis Files 20
Raw Data 20
Processed Data 20
Exploratory Figures 21
Final Figures 21
Raw Scripts 21
Final Scripts 21
R Markdown Files 21
README files 22
Text of the document 22
Further Resources 22

Markdown & knitr **22**
Coding Standards in R 22
Markdown 23
Syntax 23
R Markdown 25
What is Markdown? 25
What is R Markdown? 25
R Markdown Demo 26
knitr (Part 1) 26
Literate Statistical Programming 26
How do I Make My Work Reproducible? 27
Pros and Cons 27
knitr (Part 2) 27
What is knitr? 27
What is knitr Good For? 28
knitr (Part 3) 28
A few notes 28
knitr (Part 4) 29
Processing of knitr Documents 29
Inline Text Computations 29
Some calls to add in the begining of a code chunk: {r . . . , . . . } 29
Setting Global Options 31

Course Project 1 **31**

Reproducible Research Checklist & Evidence-based Data Analysis **31**
Communicating Results 31
RPubs 31
Reproducible Research Checklist (Part 1) 32
Reproducible Research Checklist (Part 2) 32
Reproducible Research Checklist (Part 3) 32
Evidence-based Data Analysis (Part 1) 32
Evidence-based Data Analysis (Part 2) 32
Evidence-based Data Analysis (Part 3) 32
Evidence-based Data Analysis (Part 4) 32
Evidence-based Data Analysis (Part 5) 32

Case Studies & Commentaries **32**
Caching Computations 32
Case Study: Air Pollution 32
Case Study: High Throughput Biology 32
Commentaries on Data Analysis 32

Course Project 2 **32**

Intro

- Reproducible Research applies to data analysis but also any sort of processing of data to help convey what has been done to the data so an analyis can be reproduced in the future.

- This course will cover the tools one can use in R to communicate what one has done with the data

Course Description

- "**In this course you will learn the ideas of reproducible research and reporting of statistical analyses. Topics covered include literate programming tools, evidence-based data analysis, and organizing data analyses. In this course you will learn to write a document using R markdown, integrate live R code into a literate statistical program, compile R markdown documents using knitr and related tools, publish reproducible documents to the web, and organize a data analysis so that it is reproducible and accessible to others.**"

Course Book

(The book can be downloaded as a pdf from leanpub)[https://leanpub.com/reportwriting]

What is Reproduciblity about?

- Peng makes an analogy between data science and music, he compares two songs: + **(Code Monkey)[https://www.youtube.com/watch?v=qYodWEKCuGg**]
  + **(Symphony No. 8)[https://www.youtube.com/watch?v=e7WgXhUBrps**]
- The second song is quite complex, it's even been nicknamed "Symphony of a Thousand" for the amount of people required to perform it. The score that comes with it gives detailed information of what every section is to be doing during the piece.
- In addition, *Mahler* was a conducter and often felt frustrated with scores that had complex parts but didn't convey enough information about what the composer wanted. So when he wrote his music he wrote detailed instructions with the score.
- In Data analysis there is no one unified way that the "score" of a data analysis is conveyed. As such everyone has thier own way from describing what was done to providing all the code. The first can sometimes be lacking and the second can seem to be an information overload.

Concepts, Ideas, & Structure

Concepts and Ideas (Part 1)

Replication

- The ulitimate standard for strengthening scientific evidence is replication of findings and conducting studies with independent: + Investigators
  + Data
  + Analytical methods
  + Laboratories
  + Instruments

- Replication is particularly important in studies that can impact broad policy or regulatory decisions

However, * Some studies cannot/can be challenging to be replicated
+ No time, studies nowadays require large sample sizes
+ No money, researchers gotta eat too
+ Unique, sometimes a study is of a particular subset (Air Pollution, 'rona)
* Reproducible Research makes analytic data and code available so that others may reproduce findgs; a middle ground between replication and nothing

Why Do We Need Reproducible Research?

- New technologies increasing data collection throughput; data are more complex and extremely high dimensional

- Existing data bases can be merged into new "megadatabases"

- Computing power is greatly increased, allowing more sophisticated analyses
  + Kinda like using DNA evidence for old cold cases

- For every field "X" there is a field "Computational X"
  + Reproducing the Computational X from the X will allows others to be confident the correct analysis was done

Example: Reproducible Air Pollution and Health Research

- Estimating small (but important) health effects in the presence of much stronger signals
  + Air pollution lightly impacts health but still effects it enough.. on occasion

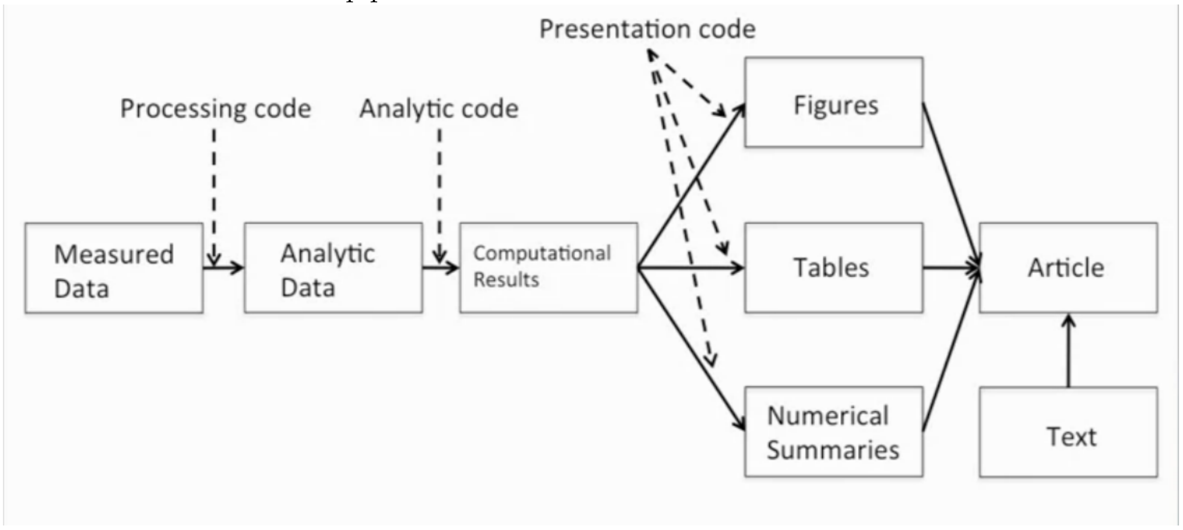
- Results inform substantial policy decisions, affect many stakeholders
  + EPA regulations can cost billions of dollars, so the research must be reproducible to convey the reason for the need of these new regulations

- Complex statistical methods are needed and subjected to intense scrutiny
- See Also: Internet-based Health and Air Pollution Surveillance System (iHAPSS)

Concepts and Ideas (Part 2)

Research Pipeline

- When you read an article you only get the article, not the data that are behind it.
- This is where the research pipeline comes in..



Recent Developments in Reproducible Research

(The Duke Saga)[https://www.youtube.com/watch?v=eV9dcAGaVU8&feature=emb_err_watch_on_yt]
(Evolution of Translational Omics: Lessons Learned and the Path Forward)[https://www.nap.edu/catalog/13297/evolution-of-translational-omics-lessons-learned-and-the-path-forward]
* In the Discovery/Test Validation stage of omics-based tests:
+ **Data/metadata** used to develop test should be made publicly available
+ The **computer code** and fully specified computational procedures used for development of the candidate omics-based test should be made sustainably available
+ "Ideally, the computer code that is released will **encompass all of the steps of computational analysis**, including all data preprocessing steps, that have been described in this chapter. All aspects of the analysis need to be transparently reported."

What do We Need for Reproducible Research?

- Analytic data are available

- Analytic code are available
- Documentation of code and data
- Standard means of distribution

Who are the Players: * Authors
+ Want to make their research reproducible
+ Want tools for RR to make their lives easier (or at least not much harder)
* Readers
+ Want to reproduce (and perhaps expand upon) interesting findings
+ Want tools for RR to make their lives easier

Challenges

- Authors must undertake considerable effor to put data/results on the web (may not have resources like a web server)
- Readers must download data/results individually and piece together which data go with which code sections, etc.
- Readers may not have the same resources as authors
- Few tools to help authors/readers (although toolbox is growing!)

In Reality…
* Authors + Just put stuff on the web
+ (Infamous) Journal supplementary materials skewed about
+ There are some central databases for various fields (e.g. biology, ICPSR)
* Readers
+ Just download the data and (try to) figure it out
+ Piece together the software and run it

Concepts and Ideas (Part 3)

Literate (Statistical) Programming

- An article is a stream of **text** and **code**
- Analysis code si divided into text and code "chunks"
- Each code chunk loads data and computes results
- Presentation code formats results (tables, figures, etc.)

- Article text explains what is going on

- Literate programs can be **weaved** to produce human-readable documents and **tangled** to produce machine-readable documents
- Literate programming is a general concept that requires:

1) A documentation language (human readable)

2) A programing language (machine readable)

Sweave

- Pronounced S-weave
- Uses L[A]T_E_X (Pretend that worked) and R as the documentation and programming languages

- Sweave was developed by Friedrich Leisch (member of the R Core) and is maintained by R core

- **Website**

Limitations: * Focused primarily on LaTeX, a difficult to learn markup language used "only by weirdos"
* Lacks features like caching, multiple plots per chunk, mixing programming languages and many other technical items
* Not frequently updated or very actively developed

knitr

- knitr si an alternative (more recent) package

- Brings together many features added on to Sweave to address limitations

- knitr uses R as the programming language (although others are allowed) and variety of documentation languages
  + LaTeX, Markdown, HTML

- knitr was developed by Yihui Xie (while a graduate student in statistics at Iowa State)

- **Website**

Reminder to commit, delete this line *AFTER* committing

Summary

- Reproducible research is important as a **minimum standard**, particularly for studies that are difficult to replicate

- Infrastructure is needed for **creating** and **distributing** reproducible documents, beyhond what is currently available

- There is a growing number of tools for creating reproducible documents

Scipting Your Analysis

- Scripting everything helps make your work as reproducible as possible

- In the past one may have written everything down in a lab notebook, but now with computers we document everything with scripts in computers

- To make an analogy to music, the final paper/presentation is like the melody, but the exploratory work is like the supporting instruments in a song

- Simular to a score in music we need a way to document everything that's going on in an analysis, this is the script

Structure of a Data Analysis (Part 1): Defining to Cleaning

- General steps in a data analysis:
  - Define the question
  - Define the ideal data set
  - Determine what data you can access
  - Obtain the data
  - Clean the data
  - Exploratory data analysis
  - Statistical prediciton/modeling
  - Interpret results
  - Challenge results
  - Synthesize/write up results

    - Create reproducible code

"*Ask yourselves, what problem have you solved, ever, that was worth solving, where you knew all fo the given information in advance? Where you **didn't** have a surplus of information and have to filter it out, or you had insuffcent information and have to go find some?*" - Dan Myer, Maths Educator; **TED talk "Math class needs a makeover"**

- A lot of the process of data analysis is filtering through all the information

Defining a question

- The more effort you can put into coming up with a reasonable question, the less effort you'll have to spend sorting through a lot of stuff; most powerful dimension reduction tool you can employ.
- Narrowing down question will reduce potential noise in a large data set

- The science will determine the question, leading to the data, leading to the applied statistics, from here one could develop theoretical statistics should their skill allow

- The applied statistics have to be thoruhgly thought through to use the appropriate methods to make some conclusion

An example: * Start with a general question
+ Can I automatically detect emails that are SPAM and those that are not? (Side Note: SPAM email comes from a reference to **this Monty Python sketch**, as such legit email is classified as "HAM")

- Make it concrete
    - Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?

Define the ideal data set

- The data set may depend on your goal
    - Descriptive - a whole population
        * Ex: All of the emails in the universe

    - Exploratory - a random sample with many variables measured

    - Inferential - the right population, randomly sampled
        * Have to be careful of the sampling mechanism and the population you're drawing from

    - Predictive - a training and test data set from the same population

    - Causal - data from a randomized study
        * "If I modify this compenet, something else will happen

    - Mechanistic - data about all components of the system

Our example:
* One could get all the emails **from googles datacenter**

Determine what data you can access

- Sometimes you can find data free on the web
- other times you may need to buy the data
- Be sure to respect the terms of use
- If the data don't exist, you may need to generate it yourself

Our example:
* Google's data center security is quite high and getting *everyone's* emails would be releasing some personal information so we'll probably have to go with something else, **since Google isn't evil**

- A possible solution is to use **the Spambase dataset**

Obtain the data

- Try to obtain the raw data
- Be sure to reference the source
- Polite emails go a long way if you need data from someone
- If you will load the data from an internet source, record the url and time accessed

(Data set for our example comes with the kernlab package.)[http://search.r-project.org/library/kernlab/html/spam.html] How the data was previously processed is also documented in that link.

Clean the data

- Raw data often needs to be processed
- If it is pre-processed, make sure you understand how
- Understand the source of the data (sensus, sample, convenience sample, etc.)
- May need reformating, subsampling - record these steps so they can be reproduced
- **Determine if the data are good enough** - if not, quit or change data

Out cleaned data set

```r
library(kernlab)
data(spam)
str(spam[, 1:5])
```

```
## 'data.frame':    4601 obs. of  5 variables:
##  $ make    : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ address: num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all     : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ num3d   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our     : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
```

Structure of a Data Analysis (Part 2): Exploring to Creating Reproducible Code

Subsampling our data into test & train sets

```r
library(kernlab)
data(spam)
set.seed(3435)
trainIndicator <- rbinom(nrow(spam), size = 1, prob = 0.5)
table(trainIndicator)
```

```
## trainIndicator
##    0    1
## 2314 2287
```

```r
trainSpam <- spam[trainIndicator == 1, ]
testSpam <-  spam[trainIndicator == 0, ]
```

Exploratory data analysis

- Look at summaries of the data
- Check for missing data
- Create exploratory plots
- Perform exploratory analyses (e.g. clustering)

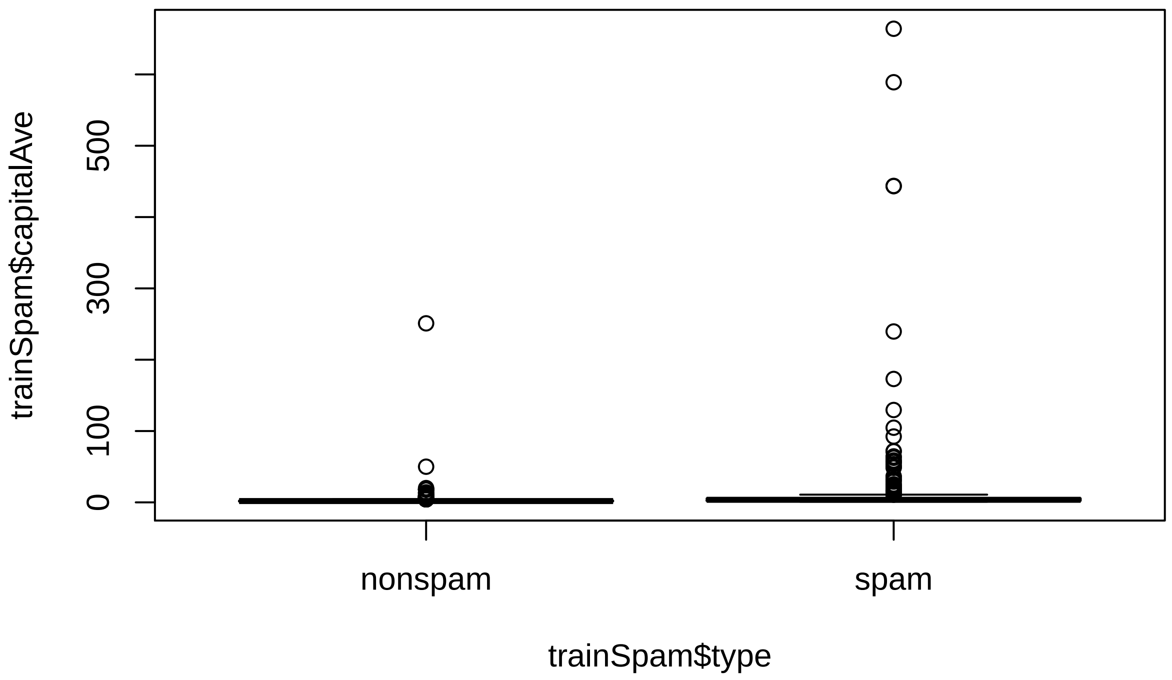
```r
#Checkin' out data
names(trainSpam)
```

```
##  [1] "make"              "address"           "all"
##  [4] "num3d"             "our"               "over"
##  [7] "remove"            "internet"          "order"
## [10] "mail"              "receive"           "will"
## [13] "people"            "report"            "addresses"
## [16] "free"              "business"          "email"
## [19] "you"               "credit"            "your"
## [22] "font"              "num000"            "money"
## [25] "hp"                "hpl"               "george"
## [28] "num650"            "lab"               "labs"
## [31] "telnet"            "num857"            "data"
## [34] "num415"            "num85"             "technology"
## [37] "num1999"           "parts"             "pm"
## [40] "direct"            "cs"                "meeting"
## [43] "original"          "project"           "re"
## [46] "edu"               "table"             "conference"
## [49] "charSemicolon"     "charRoundbracket"  "charSquarebracket"
## [52] "charExclamation"   "charDollar"        "charHash"
## [55] "capitalAve"        "capitalLong"       "capitalTotal"
## [58] "type"
```

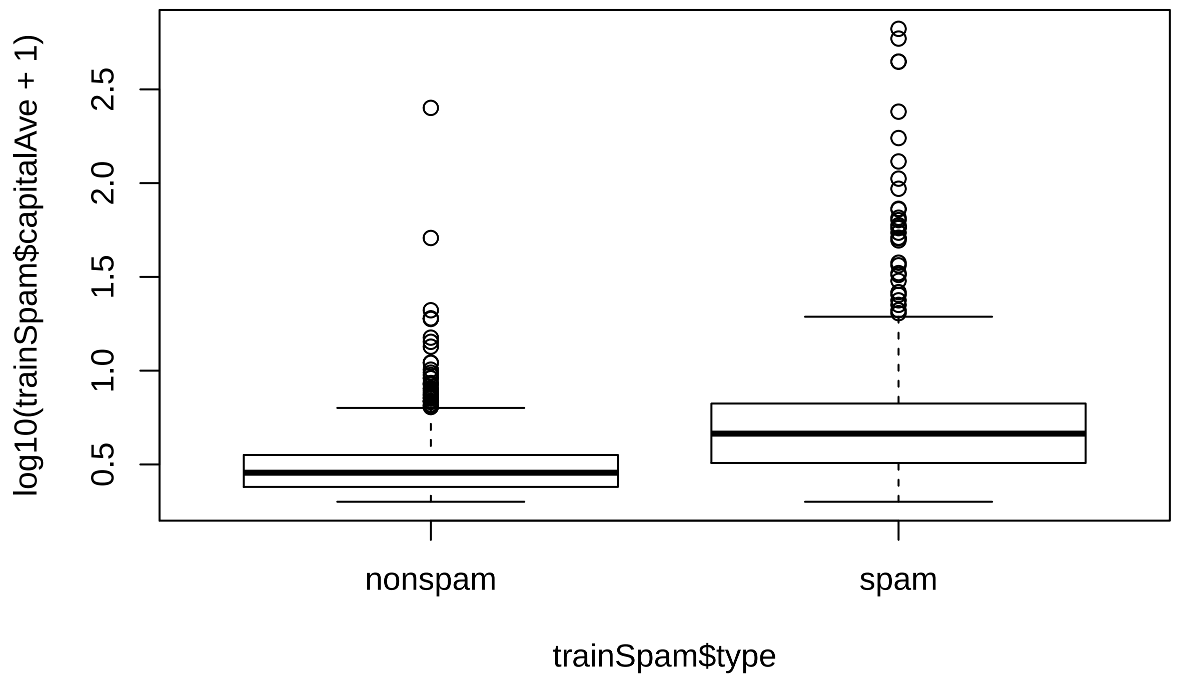
```
head(trainSpam) #freq of words in emails
```

```
##    make address  all num3d  our over remove internet order mail receive will
## 1  0.00    0.64 0.64     0 0.32 0.00   0.00        0  0.00 0.00    0.00 0.64
## 7  0.00    0.00 0.00     0 1.92 0.00   0.00        0  0.00 0.64    0.96 1.28
## 9  0.15    0.00 0.46     0 0.61 0.00   0.30        0  0.92 0.76    0.76 0.92
## 12 0.00    0.00 0.25     0 0.38 0.25   0.25        0  0.00 0.00    0.12 0.12
## 14 0.00    0.00 0.00     0 0.90 0.00   0.90        0  0.00 0.90    0.90 0.00
## 16 0.00    0.42 0.42     0 1.27 0.00   0.42        0  0.00 1.27    0.00 0.00
##    people report addresses free business email  you credit your font num000
## 1    0.00      0         0 0.32        0  1.29 1.93   0.00 0.96    0      0
## 7    0.00      0         0 0.96        0  0.32 3.85   0.00 0.64    0      0
## 9    0.00      0         0 0.00        0  0.15 1.23   3.53 2.00    0      0
## 12   0.12      0         0 0.00        0  0.00 1.16   0.00 0.77    0      0
## 14   0.90      0         0 0.00        0  0.00 2.72   0.00 0.90    0      0
## 16   0.00      0         0 1.27        0  0.00 1.70   0.42 1.27    0      0
##    money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1   0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 7   0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 9   0.15  0   0      0      0   0    0      0      0 0.15      0     0
## 12  0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 14  0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 16  0.42  0   0      0      0   0    0      0      0 0.00      0     0
##    technology num1999 parts pm direct cs meeting original project re edu table
## 1           0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 7           0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 9           0    0.00     0  0   0.00  0       0      0.3       0  0   0     0
## 12          0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 14          0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 16          0    1.27     0  0   0.42  0       0      0.0       0  0   0     0
##    conference charSemicolon charRoundbracket charSquarebracket charExclamation
## 1           0         0.000            0.000                 0           0.778
## 7           0         0.000            0.054                 0           0.164
## 9           0         0.000            0.271                0            0.181
## 12          0         0.022            0.044                0            0.663
## 14          0         0.000            0.000                0            0.000
## 16          0         0.000            0.063                0            0.572
##     charDollar charHash capitalAve capitalLong capitalTotal type
## 1        0.000    0.000      3.756          61          278 spam
## 7        0.054    0.000      1.671           4          112 spam
## 9        0.203    0.022      9.744         445         1257 spam
## 12       0.000    0.000      1.243          11          184 spam
## 14       0.000    0.000      2.083           7           25 spam
## 16       0.063    0.000      5.659          55          249 spam
```

```r
#Look at some plots
plot(trainSpam$capitalAve ~ trainSpam$type)#Avg capital letters
```



```r
#Data is hard to see so looking at the log will help
plot(log10(trainSpam$capitalAve + 1) ~ trainSpam$type)
```

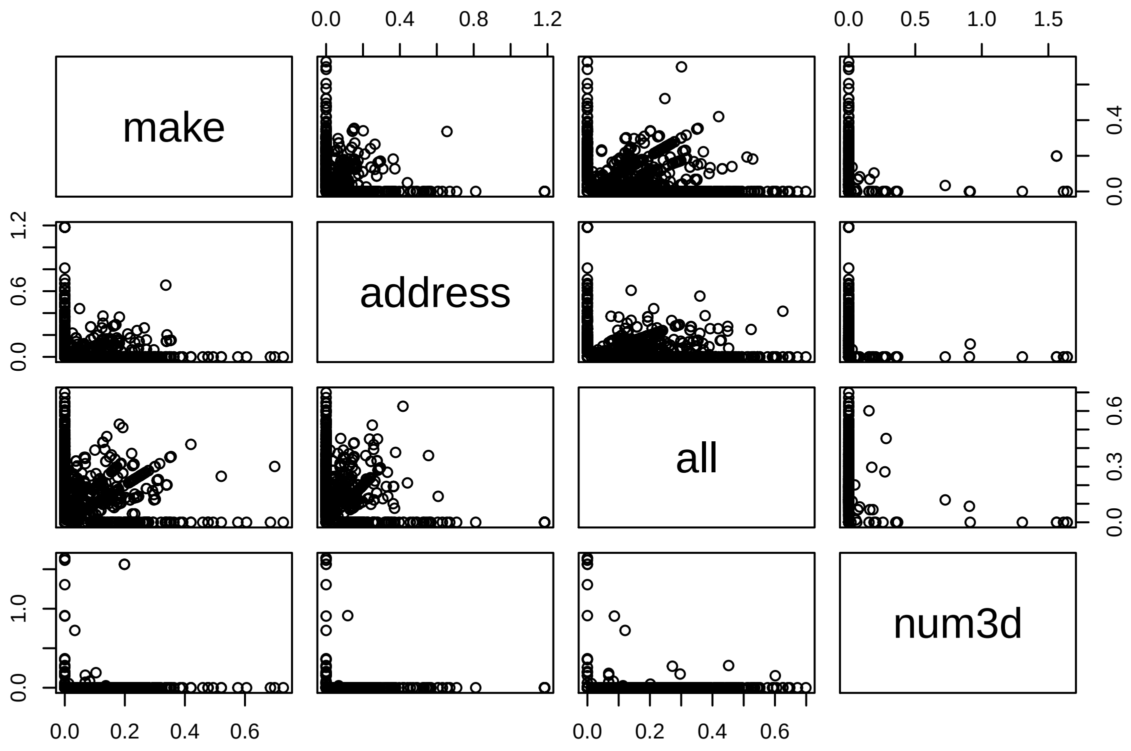


```
# add 1 to evaluate 0s - ok for exploring, not reports
```

- The second plot helps us see the freq of capitals in spam is higher than in nonspam

Relationships between predictors:

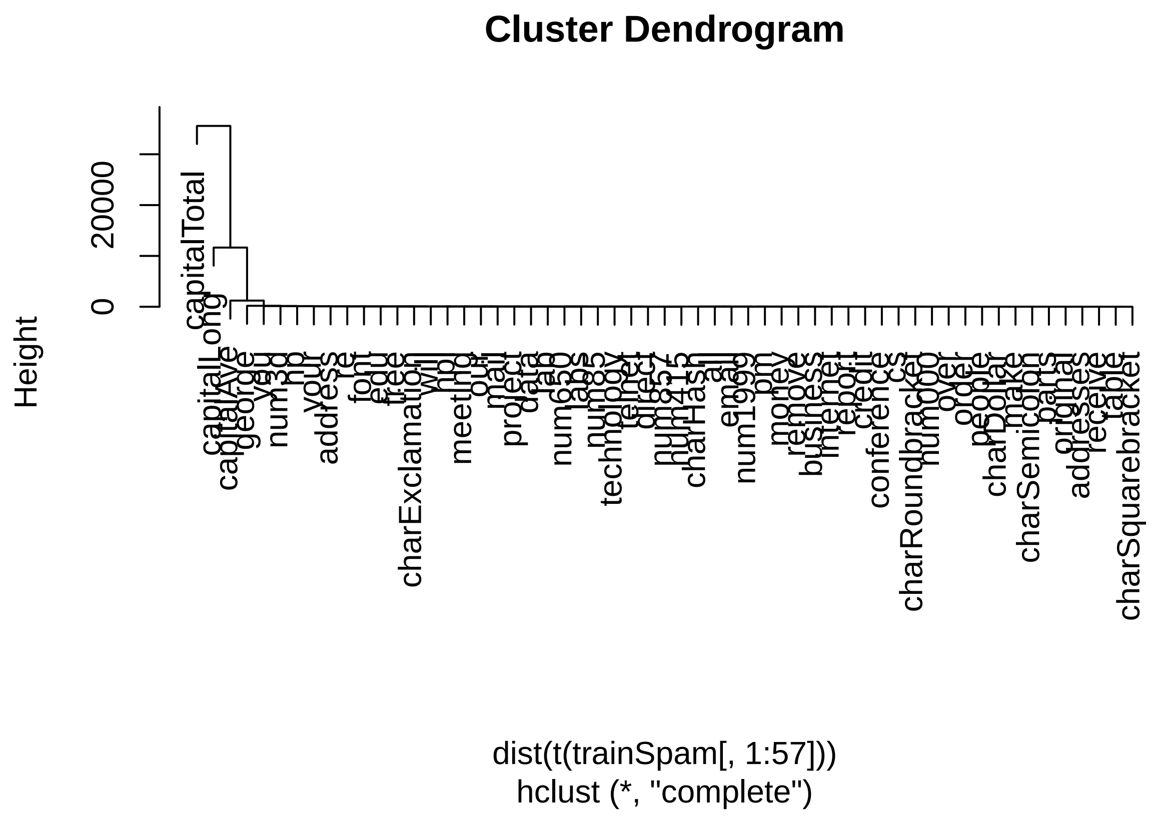
```
plot(log10(trainSpam[, 1:4] + 1))
```



* Words in the diagonal tell what predictor those respective rows & cols are looking at
+ Referred to as a "Paris plot"

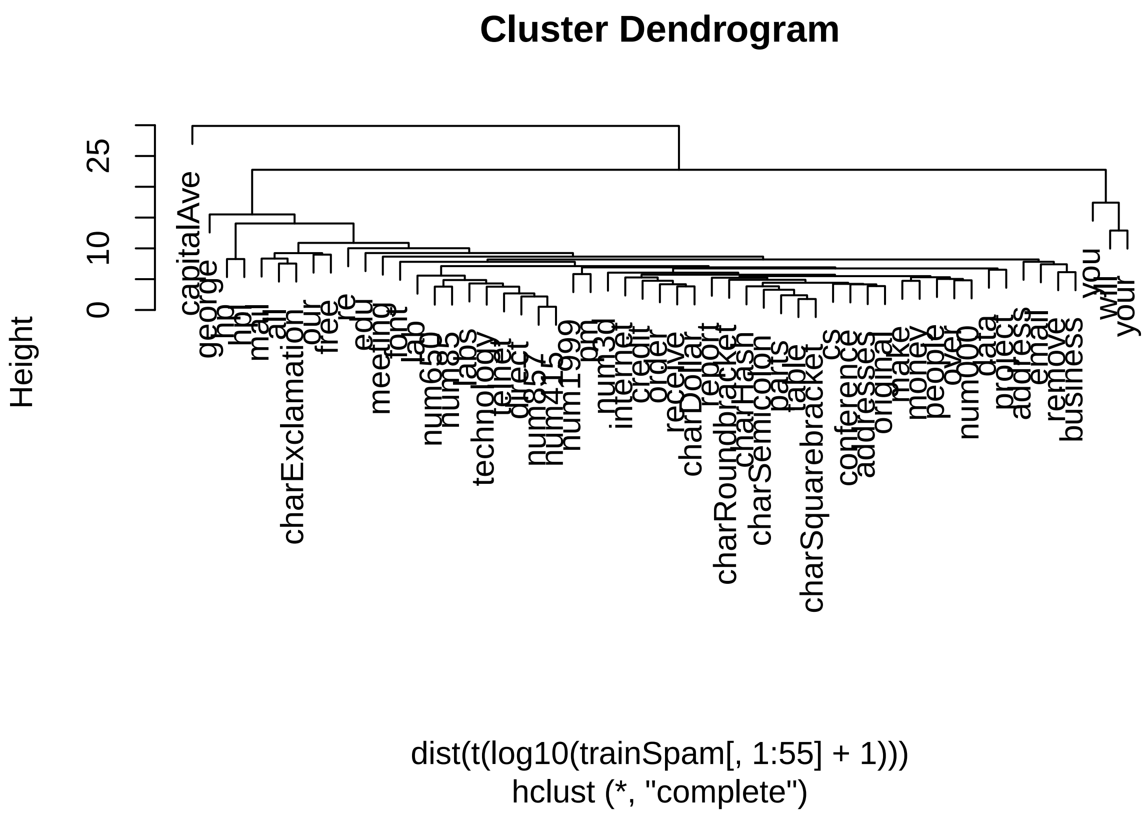
Clustering:

```
hCluster = hclust(dist(t(trainSpam[, 1:57])))
plot(hCluster)
```



* not very informative since info is skewed, so clustering a log might give a better insight

```r
hClusterUpdated <- hclust(dist(t(log10(trainSpam[, 1:55] + 1))))
plot(hClusterUpdated)
```



Statistical prediction/modeling

- Should be informed by the sefults of your exploratory analysis
- Exact methods depend on the question of interest
- Transformations/processing should be accounted for when necessary
- Measures of uncertainty should be reported
  Example:

```
#Which predictor has minimum cross-validated error?
names(trainSpam)[which.min(cvError)]
```

```
## [1] "charDollar"
```

Get a measure of uncertainty

```
## Use the best model from the group
predictionModel = glm(numType ~ charDollar,
                      family = "binomial", data = trainSpam)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
## Get predictions on the test set
predictionTest = predict(predictionModel, testSpam)
predictedSpam = rep("nonspam", dim(testSpam)[1])

## Arb. Classify as 'spam' for those with prob > 0.5
predictedSpam[predictionModel$fitted > 0.5] = "spam"
head(predictedSpam)
```

```
## [1] "nonspam" "nonspam" "spam"    "nonspam" "nonspam" "nonspam"
```

Get a measure of uncertainty

```r
## Classification table
result <- table(predictedSpam, testSpam$type)
result
```

```
##
## predictedSpam nonspam spam
##       nonspam    1346  458
##       spam         61  449
```

```r
## Error rate
errors <- result[1,2] + result[2,1]
total <- sum(result[1:2,1:2])
print(paste0(errors, " errors occured out of ",
      total, " readings resulting in an error rate of ",
      round(errors/total, 4)))
```

```
## [1] "519 errors occured out of 2314 readings resulting in an error rate of 0.2243"
```

Interpret results

- Use the appropriate language
  - "describes"
  - "correlates with/associated with"
  - "leads to/causes"
  - "predicts"
- Give an explanation
- Interpret coefficients
- Interpret measures of uncertainty

In our example: * The fraction of chacters that are dollar signs can be used to predict if an email is Spam
* Anything with more than 6.6% dollar signs is classified as Spam

* More dollar signs always means more Spam under our prediction
* Our test set error rate was 22.4%

Challenge Results

- Challenge all steps:
  - Question
  - Data source
  - Processing
  - Analysis
  - Conclusions
- Challenge measures of uncertainty
- Challenge choices of terms to include in models
- Think of potential alternative analyses

Synthesize/write-up results

- Lead with the question
- Summarize the analyses into the story
- Don't include every analysis, include it:
  - If it is needed for the story
  - If it is needed to address a challenge
- Order analyses according to the story, rather than chronologically
- Include "pretty" figures that contribute to the story

In our example: * Lead with the question
+ "*Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?*"
* Describe the approach
+ Collected data from UCI -> created training/test sets
+ Explored relationships
+ Choose logistic model on training set by cross validation
+ Applied to test, 78% test set accuracy
* Interpret results
+ Number of dollar signs seems reasonable, e.g. "Make money with Viagra $ $ $ $!" * Challenge results

+ 78% isn't that great
+ I could use more variables
+ Why logistic regression?

Organizing Your Analysis

- No universal way for every data analysis, however this lecture aims to give some useful tips

Types of Data Analysis Files

- Data
  - Raw data
  - Processed data
- Figures
  - Exploratory figures - not very polished
  - Final figures
- R code
  - Raw / unused scripts
  - Final scripts - easier to read, commented
  - R Markdown files
- Text
  - README files
  - Text of analysis / report

Raw Data

- Should be stored in your analysis folder
- If accessed fromt he web, include url, description, and date accessed in README
- Adding raw data to Git repo is good, however sometimes these files are too large to be stored on GitHub

Processed Data

- Processed data should be named files so it is easy to see which script generated the data

- The processing script - processed data mapping should occur in the README
- Processed data should be **tidy**

Exploratory Figures

- Figures made during the course of your analysis, not necessarily part of your final report
- They do not need to be "pretty"

Final Figures

- Usually a small subset of the original, exploratory figures
- Axes/colors set to make the figure clear
- Possibly multiple panels (helps to condence related info)
- Labeled well and annotated to help readers understand what's going on with the data

Raw Scripts

- May be less commented (although comments do help you)
- May be multiple versions
- May include analyses that are later discarded since they lead to a dead-end

Final Scripts

- Clearly commented
  - Small comments liberally - aim to answer the "what, when, why, and how"s
  - Bigger commented blocks for whole sections of code
- Include processing details
- Only analyses that appear in the final write-up, helps others view the process and reproduce

R Markdown Files

(Is this where I say something about all these notes being in R Markdown?)
* R markdown files (`.Rmd`) can be used to generate repoducible reports
* Text and R code are integrated in one document
* Very easy to create in `Rstudio`

README files

- Explain what's going on in the directory

- Not necessary if you use R markdown files, as those ussually will be stating what's going on as code is executed

- Should contain step-by-step instructions for analysis

- **Here is an example**

Text of the document

- It should include a title, introduction (motivation), methods (statistics you used), results (including measures of uncertainty), and conclusions (including potential problems)

- It should tell a story

- *It should not include every analysis you performed*

- References should be incldued for statistical methods

Further Resources

- Information about a non-reproducible study that led to cancer patients being mistreated: **The Duke Saga Starter Set**

- **Reproducible research and Biostatistics**

- **Managing a statistical analysis project guidelines and best practices**

- **Project template** - a pre-organized set of files for data analysis

Markdown & knitr

Coding Standards in R

- Help make code readable so both you and others can read what your code does

- Just like any other style, such as clothing, not everyone will agree on the basic ideas but this lecture will cover some of the standards

1) Save code as text files

- Easily interpertable by all devices
- RStudio does this by default

2) Indent your code

- Separates sections of code, such as loops & functions
- Amount a tab width is is up for debate, but old-schoolers like a width of 8, but a width of 4 is considered a minimum

3) Limit the width of your code

- 80 columns is standard
- Code can be concisely viewed without annoying horizontal scrolling
- Also helps avoid issues with code readability when combined with indenting standards, a 4 nested for loop will start hitting the right margin

4) Limit the length of individual functions

- Each data should do one basic activity
  - `readData(filename)` should just read the data and return the data.table
  - `readData(filename)` should **NOT** read, process, fit a model, and print some output
- Nice to have a function written on a single page of the code to be able to evaluate what it does
- Helps with finding bugs within a function

Markdown

- Simplified markup language
- Easy to integrate with R Code and other programming languages

"*Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).*" - John Gruber, creator of Markdown

Syntax

- Italics
  - *This text will appear italicized*
  - *This text will appear italicized*

- Bold
  - **This text will appear bold!**

  - **This text will appear bold!**
- Italics & Bold
  - ***This text will appear both italicized & bold***

  - ***This text will appear both italicized & bold***
- Headings
  - ## This is a secondary heading

  - ### This is a tertiary heading
  - (Example has been ommitted as to not mess up TOC)
- Unordered Lists
  - Character doesn't matter, as long as it's consist

  - * I use this for first bullet(Line above)

  - + These for second bullet(This line)

  - - These for a third bullet
    - Third Bullet
- Ordered Lists
  - 1) first item

  - 2) second item

  - 3) third item

  1) first item

  2) second item

  3) third item

  - If you want to add something it just has to be a number followed by the same character, then markdown will order the numbers when it executes based on the inital number

  1. What if I like periods instead

  2. I forgot to add this line earlier and it starts with "`34.`"

  3. Yeah that's fine just use those

  4. Whatever you type shows up as is
- Links (Ignore the \ characters)
  - \[Johns Hopkin Bloomberg School of Public Health\]\(http://www.jhsph.edu/ \)

  - Johns Hopkin Bloomberg School of Public Health

  - Underline isn't supported in pdfs so I developed the tecnique of bolding links
    * **\[Download R\]\(http://www.r-project.org/ \)**

    * **Download R**
- Newlines require two spaces

R Markdown

What is Markdown?

- Created by John Gruber and Aaron Swartz

- A simplified version of "markup" languages

- Allows one to focus on writing as opposed to formatting

- Simple/minimal intuitive formatting elements

- Easily converted to valid HTML (and other formats) using existing tools

- Complete information is available at **this site**

- **Some background information**

What is R Markdown?

- R markdown is the integration of R code with markdown

- Allows one to create documents containing "live" R code

- R code is evaluated as part of the processing of the markdown

- Results from R code are inserted into markdown document

- A core tool in *literate statistical programming*

- R markdown can be converted to standard markdown using the `knitr` package in R

- Markdown can be converted to HTML using the `markdown` package in R

- Any basic text editor can be used to create a markdown document; no special editing tools are needed

- The R markdown –> markdown –> HTML

- Work flow can be easily managed using RStudio (but not required)
- Slides can be written in R markdown and converted using the `slidify` package

R Markdown Demo

This entire thing is made in R Markdown so just look at that related code for a demo.

knitr (Part 1)

- Helps make analysis reproducible through literate statistical programming

The problems that `knitr` aims to solve
* Authors must undertake considerable effor to put data/results on the web
* Readers must download data/results individually and piece together which data go with whcih code sections, etc.
* Authors/readers must manually interact with websites
* There is no single document to integrate data snalysis with textual representations; i. e. data, code, and text are not linked

Literate Statistical Programming

- Literate Programming orginally came from Don Knuth
- An article is to be a stream of **text** and **code**
- Analysis code is divided into text and code "chunks"
- Presentation code formats results (tables, figures, etc.)
- Article text explains what is going on in the code sections
- Literate programs are **weaved** to produced human-readable documents and **tangled** to produce machine-readable documents
- Literate programming as a general concept needs:
  - A documentation language
  - A programming language
- The original **Sweave** system developed by Friedrich Leisch used LaTeX and R
- **knitr** supports a variety of documentation languages

How do I Make My Work Reproducible?

- Decide to do it (ideally from the start)
  - Deciding at the end requries a lot of backtracking and reformatting

- Keep track fo things, perhaps with a version control system to track snapshots/changes

- Use software whose operation can be coded
  - This usually rules out and GUI software, unless it tracks what you click on

- Don't save output
  - Don't preprocess data and only keep clean data, as you won't have a record of how you created the clean data

- Save data in non-proprietary formats
  - formats where the layout of the data isn't publiclly known

  - makes difficult for others to access the data

Pros and Cons

Pros
* Text and code are all in once place, in a logical order
* Data and results are automatically updated to reflect external changes
* Code is live/automatic when building a document. Helps you know if an error in the analysis has appeared

Cons
* Text and code all in one place; can make documents difficult to read, especially if there is a **lot** of code
* Can substantially slow down processing of documents (although there are tools to help)

knitr (Part 2)

What is knitr?

- An R package written by Yihui Xie (while he was a grad student at Iowa State)
  - Available on CRAN

  - Built into RStudio

- Supports RMarkdown, LaTeX, and HTML as documentation languages

- Can export to PDF, HTML

- Built right into RStudio for your convenience
  Requirements

- A recent version of R

- A text editor (the one that comes with RStudio is fine)

- Some support packages also available on CRAN
  - auto downloaded with `install.package()` function
- Some knowledge of Markdown, LaTeX, or HTML

What is knitr Good For?

- Manuals

- Short/medium-length technical documents

- Tutorials

- Reports (esp. if generated periodically)

- Data preprocessing documents/summaries

What it's not good for
* Very long research articles
+ Hard to edit
* Complex, time-consuming computations
+ has to rerun everytime you generate the document
* Documents that require precise formatting

knitr (Part 3)

This lecture covers how to make a knitr document
* Create a R Markdown document
* Right up Rmd code you want to use to generate the document
* Hit `Knit` in RStudio
* If not in RStudio:

```
library(knitr)
setwd(<working directory>)
knit2html("document.Rmd")
browseURL("document.html")
```

A few notes

- knitr will fill a new document with filler text; normally you just want to delete this

- Code chucks begin with “‘{r} and end with “‘
    - All R code goes in between these markers

- Code chunks can have **names**, which is useful when we start making graphics
  “‘{r firstchunk}
  ## R code goes here
  “‘
- By default, code in a code chunk is echoed, as will the results of the computation (if there are results to print)

knitr (Part 4)

Processing of knitr Documents

(What happens under the hood)

- You write the RMarkdown document (`.Rmd`)

- knitr produces a Markdown document (`.md`)

- knitr converts the Markdown document into HTML (by default)

- You should **NOT** edit (or save) the `.md` or `.html` documents until you are finished

Inline Text Computations

```
time <- format(Sys.time(), "%a %b %d %X %Y")
rand <- rnorm(1)
```

- You can integrate values of object into a sentance by using *the grave key* followed by `r` to call to that object.
    - The current time is Fri May 08 05:25:20 AM 2020. A random number is 1.1542945.

Some calls to add in the begining of a code chunk: {r ..., ...}

- `echo` - Logical indicating if code should appear in output doc

- `results` - Options:
    - `"markup"`- default

    - `"asis"` - (as is) passthrough results, helpful for showing nice tables

```
library(datasets)
data(airquality)
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
```

```r
library(xtable)
xt <- xtable(summary(fit))
print(xt, type = "html")
```

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

-64.3421

23.0547

-2.79

0.0062

Wind

-3.3336

0.6544

-5.09

0.0000

Temp

1.6521

0.2535

6.52

0.0000

Solar.R

0.0598

0.0232

2.58

0.0112

- `"hide"` - do not display results
- `"hold"` - put all results below all code
- `warning` - logical indicating if warnings should appear in doc

- `eval` - logical indicating if code should be ran
- `fig.` - `height` & `width` will adjust the respective dimensions
  - for figures knitr will encode the image in HTML, so it is embedded in the actual HTML file
- `cache` - Logical indicating if results of computation should be cached
  - Helpful if a code chunk takes a long time to run
  - If set to `TRUE`, after the first run results are loaded from the cache rather than being re-computed every time; if nothing has changed in the chunk
  - `FALSE` by default
  - Dependencies are not checked explicitly, so if one code chunk depends on results from a previous and the previous is changed the old output will still be recalled from the cache
  - Chunks with significant *side effects* may not be cacheable, that is an effect outside the document

Setting Global Options

- Sometimes we want to set options for **every** code chunk that are different from the defaults
- For example, we may want to suppress all code echoing and results output
- we do this with `opts_chunk$set(<put new defaults in here>)`

Course Project 1

My project can be found on GitHub

Reminder to commit, delete this line *AFTER* committing

Reproducible Research Checklist & Evidence-based Data Analysis

Communicating Results

RPubs

Reminder to commit, delete this line *AFTER* committing

Reproducible Research Checklist (Part 1)

Reproducible Research Checklist (Part 2)

Reproducible Research Checklist (Part 3)

Reminder to commit, delete this line *AFTER* committing

Evidence-based Data Analysis (Part 1)

Evidence-based Data Analysis (Part 2)

Evidence-based Data Analysis (Part 3)

Evidence-based Data Analysis (Part 4)

Evidence-based Data Analysis (Part 5)

Reminder to commit, delete this line *AFTER* committing

Case Studies & Commentaries

Caching Computations

Case Study: Air Pollution

Reminder to commit, delete this line *AFTER* committing

Case Study: High Throughput Biology

Commentaries on Data Analysis

Reminder to commit, delete this line *AFTER* committing

Course Project 2

Reminder to commit, delete this line *BEFORE* committing