ACCT 420: Course Logistics + R Refresh

Session 1

Dr. Richard M. Crowley

About Me

Teaching

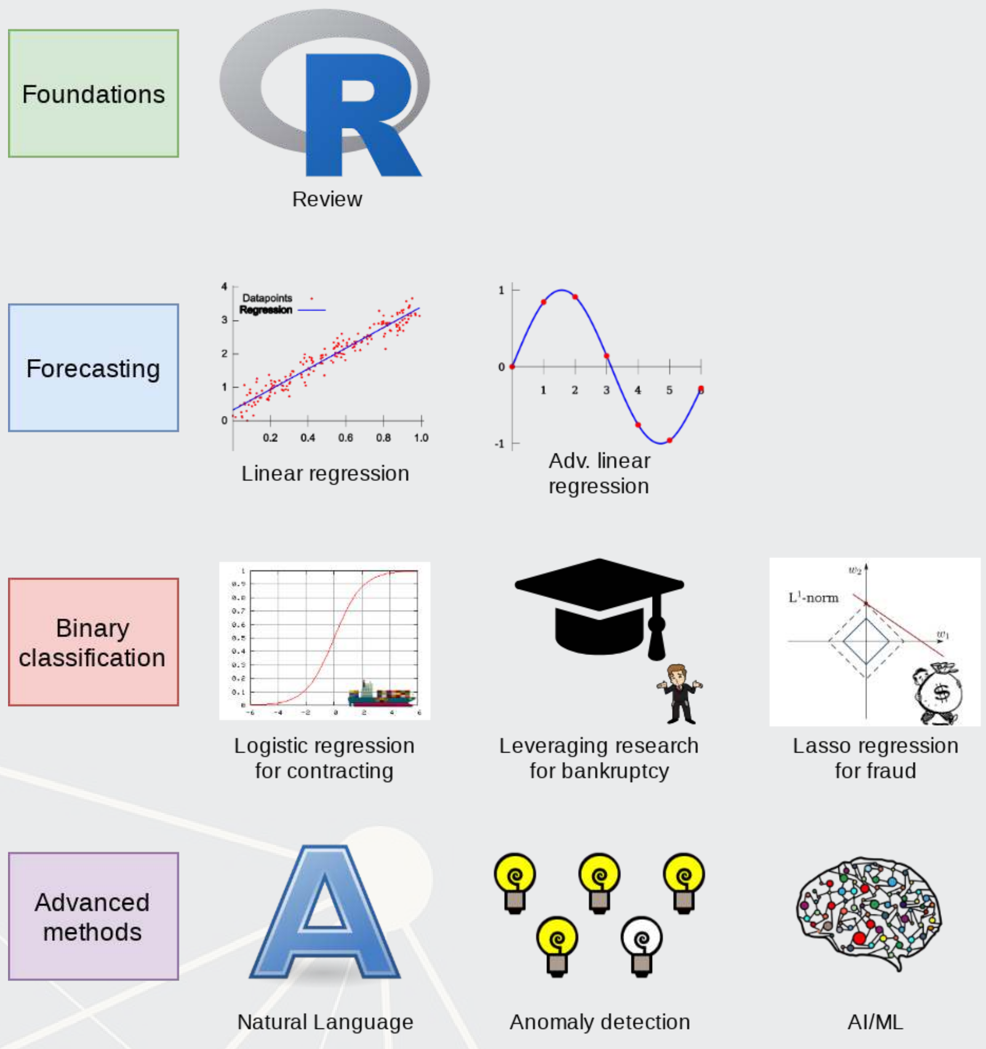
- Fourth year at SMU
  - Also teaching ACCT 101
- Before SMU: Taught at the University of Illinois Urbana-Champaign while completing my PhD

Research

- Accounting disclosure: What companies say, and why it matters
- Approach this using AI/ML techniques

About this course

What will this course cover?



Foundations

Review

Forecasting

Linear regression

Adv. linear regression

Binary classification

Logistic regression for contracting

Leveraging research for bankruptcy

Lasso regression for fraud

Advanced methods

Natural Language

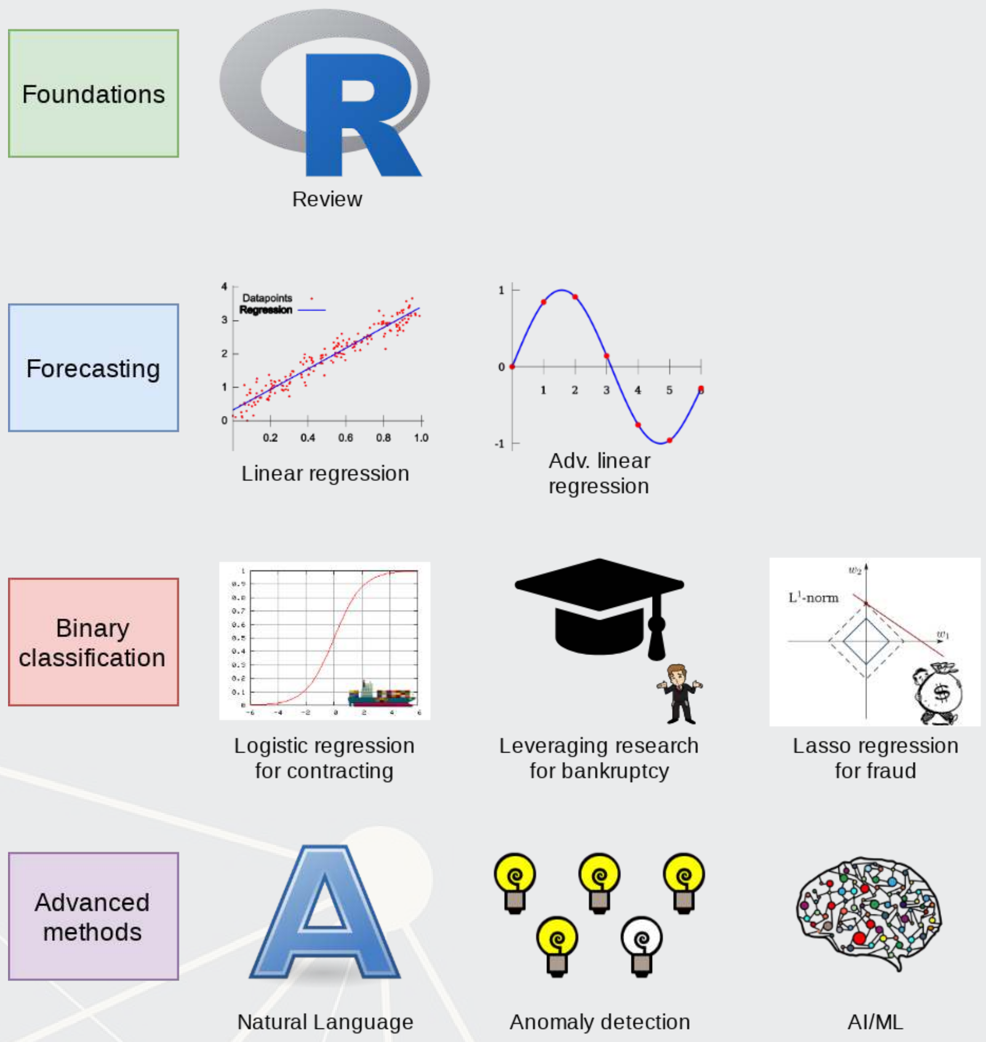
Anomaly detection

AI/ML

1. Foundations (*today*)
   - Thinking about analytics
   - In class: Setting a foundation for the course
   - Outside: Practice and refining skills on Datacamp
     - Pick any R course, any level, and try it out!
2. Financial forecasting
   - Predict financial outcomes
   - Linear models

> Getting familiar with forecasting using **real** data and R

What will this course cover?



Foundations

Review

Forecasting

Linear regression

Adv. linear regression

Binary classification

Logistic regression for contracting

Leveraging research for bankruptcy

Lasso regression for fraud

Advanced methods

Natural Language

Anomaly detection

AI/ML

3. Binary classification
   - Event prediction
     - Shipping delays
     - Bankruptcy
   - Classification & detection
4. Advanced methods
   - Non-numeric data (text)
   - Clustering
   - AI/Machine learning (ML)
     - 1 week on Ethics of AI
     - 2 weeks on current developments

> Higher level financial forecasting, detection, and AI/ML

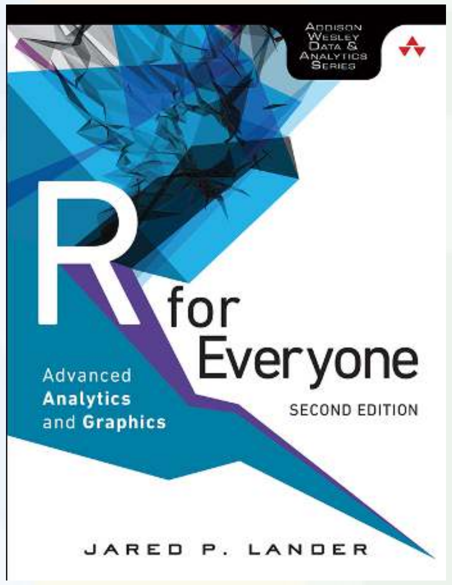
Datacamp

- Datacamp is providing *free* access to their *full* library of analytics and coding online tutorials
  - You will have free access for 6 months (Usually $25 USD/mo)
- Online tutorials include short exercises and videos to help you learn R
- I have assigned some limited materials via a Datacamp class
  - Counts towards participation
  - Check your email or eLearn for access (Sent on Aug 19)
  - Datacamp automatically records when you finish these
  - I have personally done any tutorial I assign to ensure its quality
- You are encouraged to go beyond the assigned materials – these will help you learn more about R and how to use it

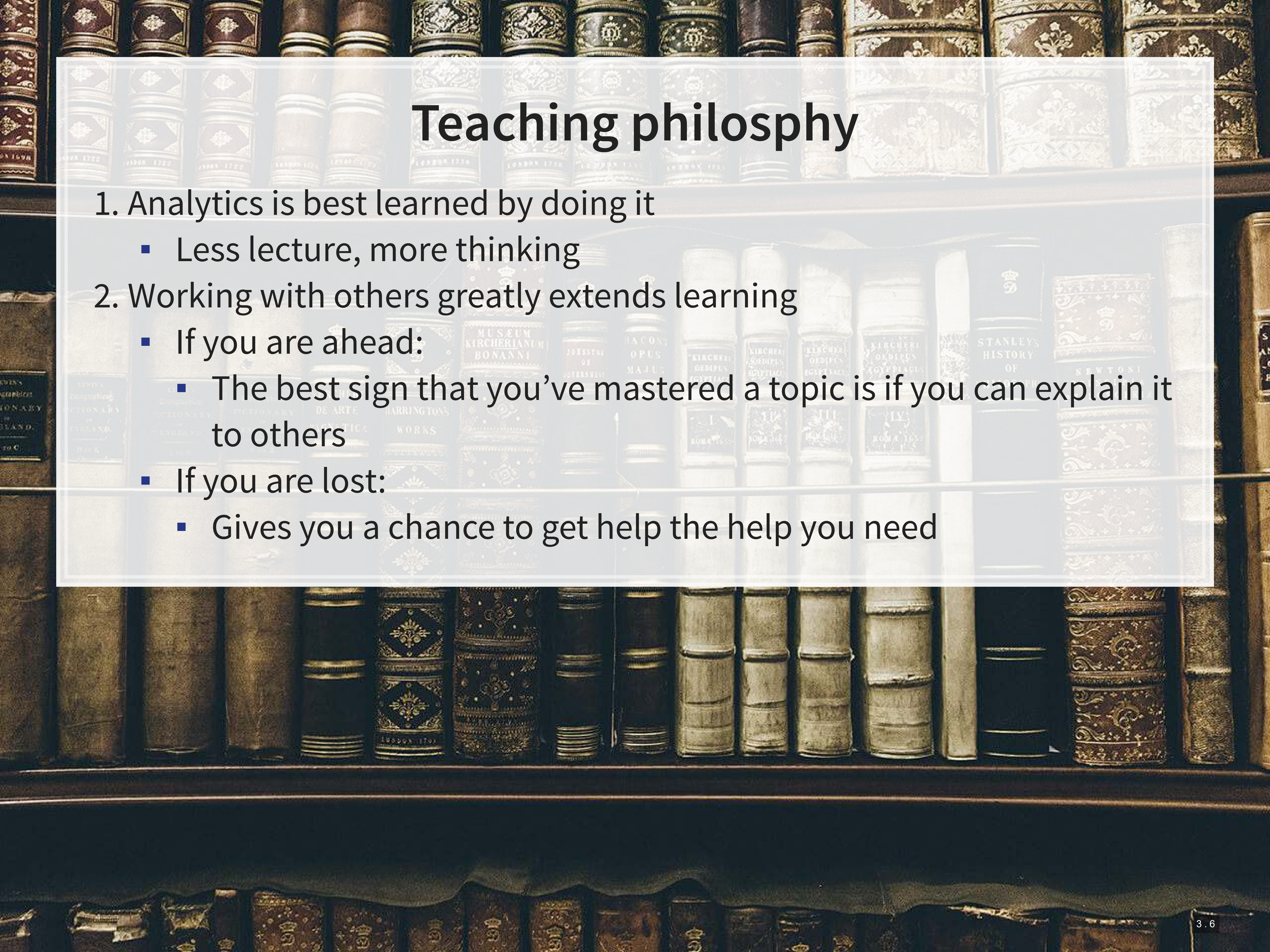
> Datacamp’s tutorials teach R from the ground up, and are mandatory unless you can already code in R.

Textbook

- There is no required textbook
  - Datacamp is taking the place of the textbook
- If you prefer having a textbook…
  - R for Everyone by Jared Lander is a good one on R



- Other course materials (slides and articles) are available at:
  - eLearn
  - https://rmc.link/acct420
    - Contains html versions of the slides with interactive content
- Announcements will be only on eLearn



Teaching philosphy

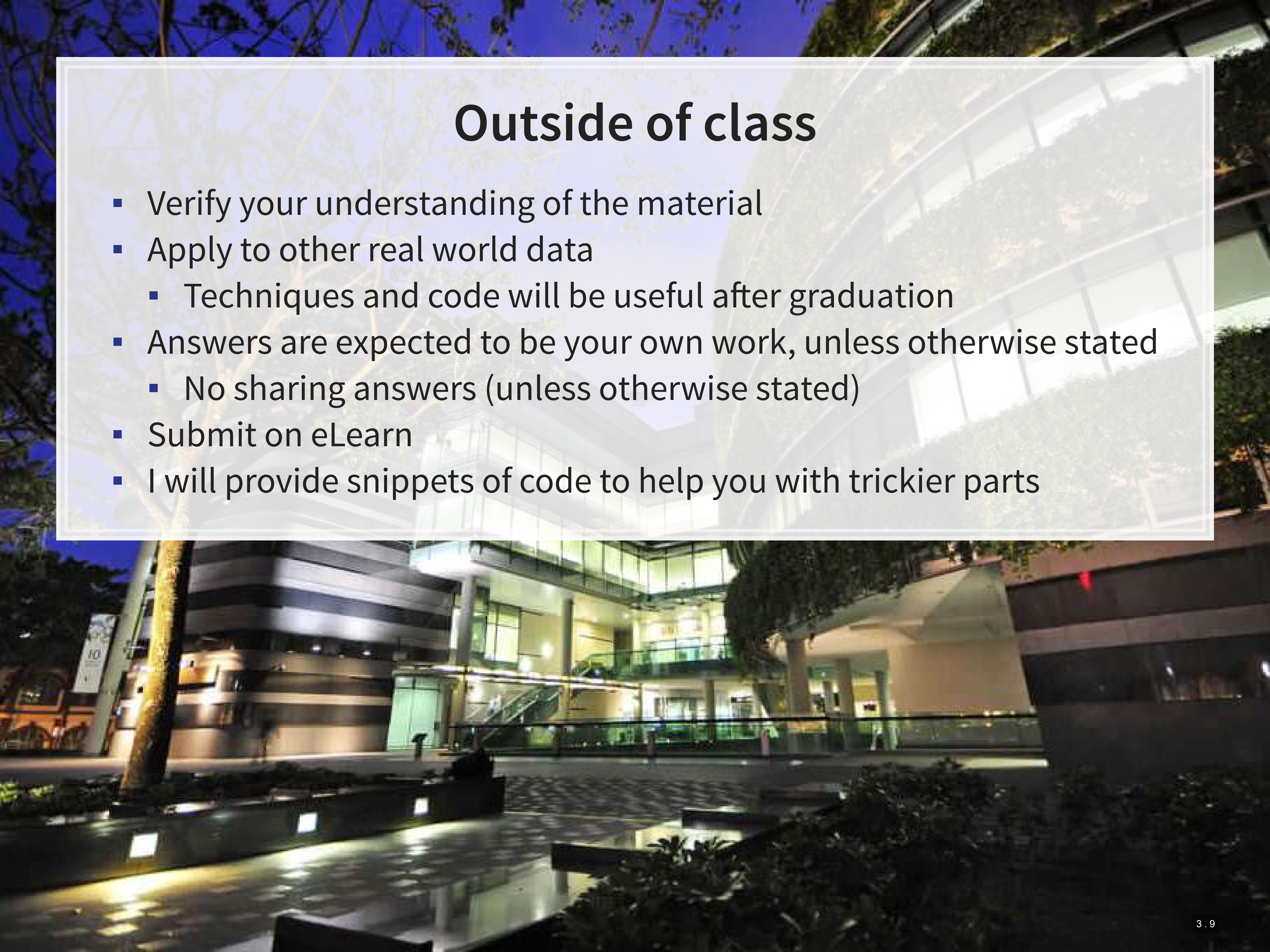
1. Analytics is best learned by doing it
   - Less lecture, more thinking
2. Working with others greatly extends learning
   - If you are ahead:
     - The best sign that you’ve mastered a topic is if you can explain it to others
   - If you are lost:
     - Gives you a chance to get help the help you need

Grading

- Standard SMU grading policy
- Participation @ 10%
- Individual work @ 20%
- Group project @ 30%
- Final exam @ 40%

Participation

- Come to class
  - If you have a conflict, email me
    - Excused classes do not impact your participation grade
- Ask questions to *extend* **or** *clarify*
- Answer questions and explain answers
  - Give it your best shot!
- Help those in your group to understand concepts
- Present your work to the class
- Do the online exercises on Datacamp



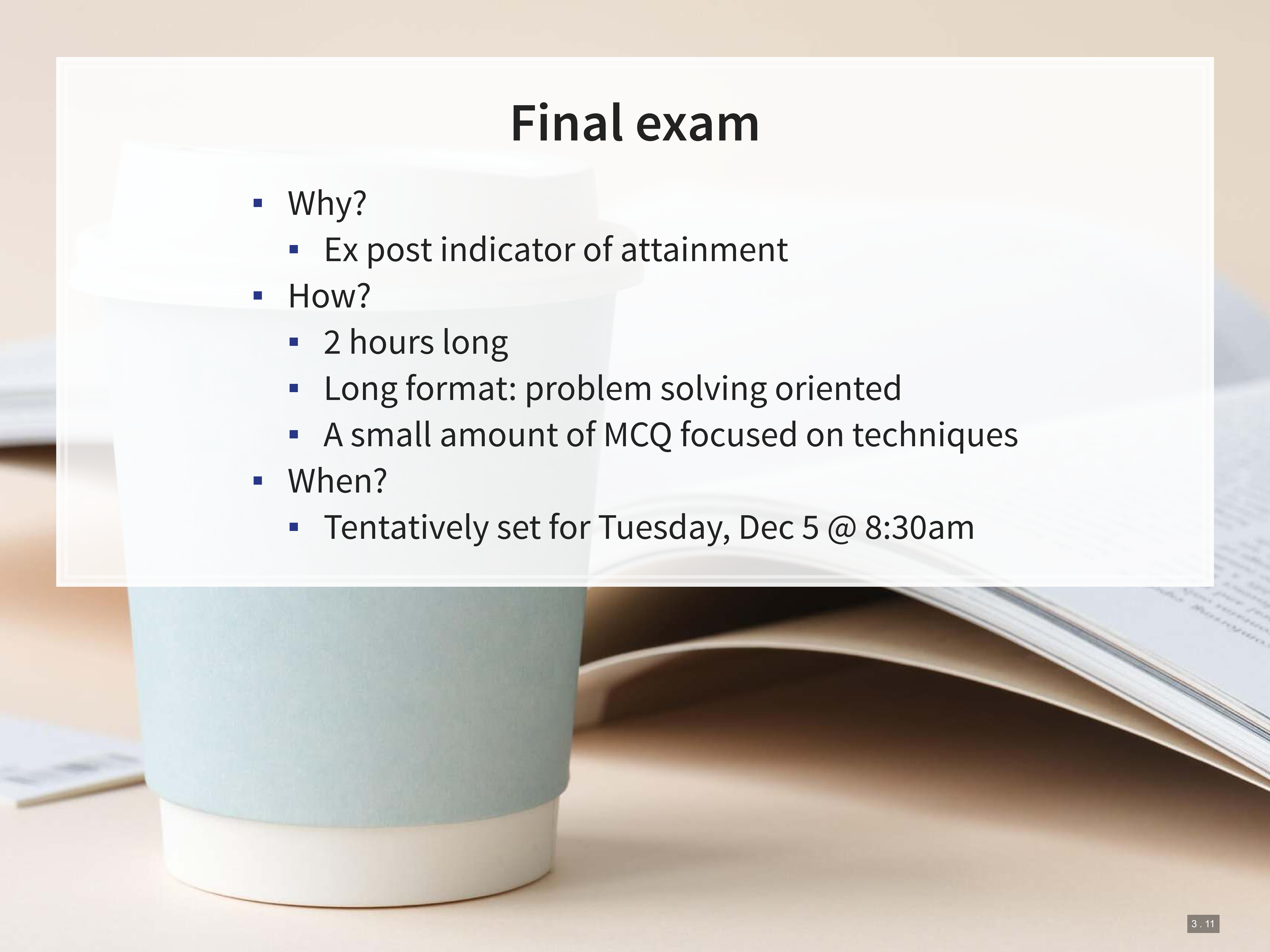
Outside of class

- Verify your understanding of the material
- Apply to other real world data
  - Techniques and code will be useful after graduation
- Answers are expected to be your own work, unless otherwise stated
  - No sharing answers (unless otherwise stated)
- Submit on eLearn
- I will provide snippets of code to help you with trickier parts

Group project

- Data science competition format, hosted on Kaggle
  - Multiple options for the project will be available
- The project will start on session 7
- The project will finish on session 12 with group presentations

kaggle



Final exam

- Why?
  - Ex post indicator of attainment
- How?
  - 2 hours long
  - Long format: problem solving oriented
  - A small amount of MCQ focused on techniques
- When?
  - Tentatively set for Tuesday, Dec 5 @ 8:30am

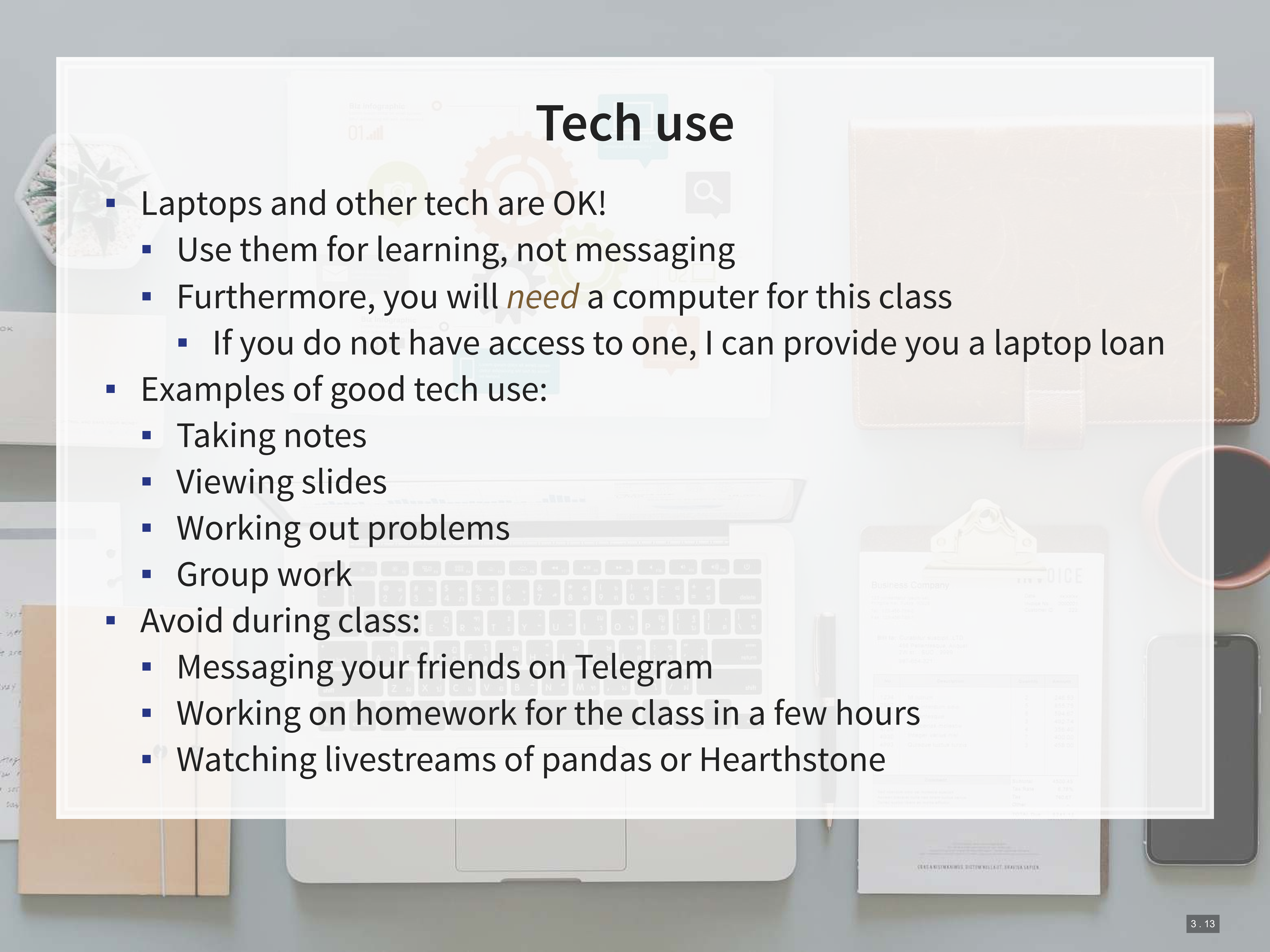
Expectations

In class

- Participate
  - Ask questions
    - Clarify
    - Add to the discussion
  - Answer questions
  - Work with classmates

Out of class

- Check eLearn for course announcements
- Do the assigned tutorials on Datacamp
  - This will make the course much easier!
- Do individual work on your own (unless otherwise stated)
  - Submit on eLearn
- Office hours are there to help!
  - Short questions can be emailed instead



Tech use

- Laptops and other tech are OK!
  - Use them for learning, not messaging
  - Furthermore, you will *need* a computer for this class
    - If you do not have access to one, I can provide you a laptop loan
- Examples of good tech use:
  - Taking notes
  - Viewing slides
  - Working out problems
  - Group work
- Avoid during class:
  - Messaging your friends on Telegram
  - Working on homework for the class in a few hours
  - Watching livestreams of pandas or Hearthstone

Office hours

- Walk-in hours **TBD**
  - Will be announced on eLearn
  - Or by appointment
- Short questions can be emailed
  - I try to respond within 24 hours

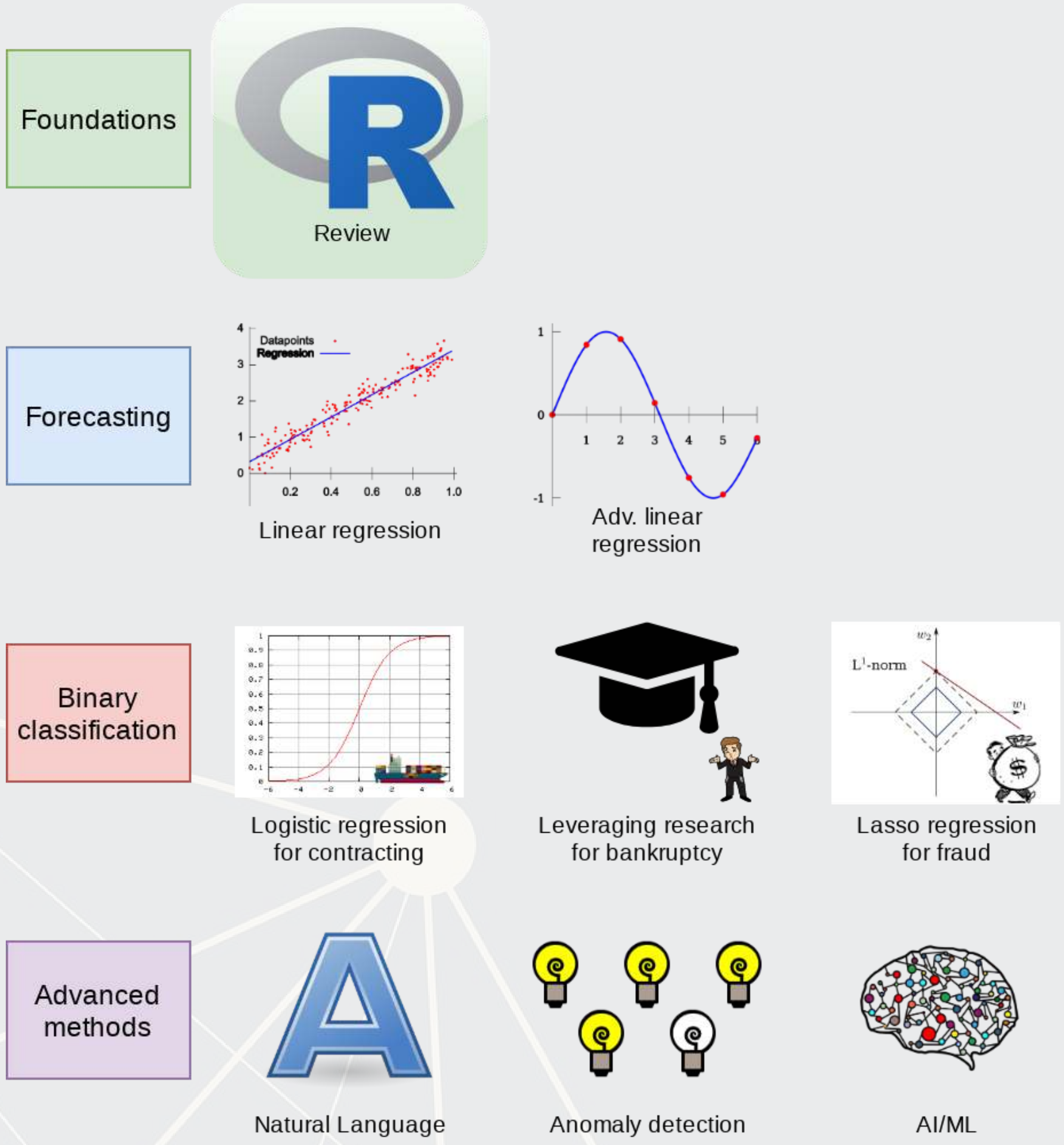
About you

About you

- Survey at rmc.link/aboutyou
- Results are anonymous
- We will go over the survey next week at the start of class

Analytics

Learning objectives



Foundations

Review

Forecasting

Linear regression

Adv. linear regression

Binary classification

Logistic regression for contracting

Leveraging research for bankruptcy

Lasso regression for fraud

Advanced methods

Natural Language

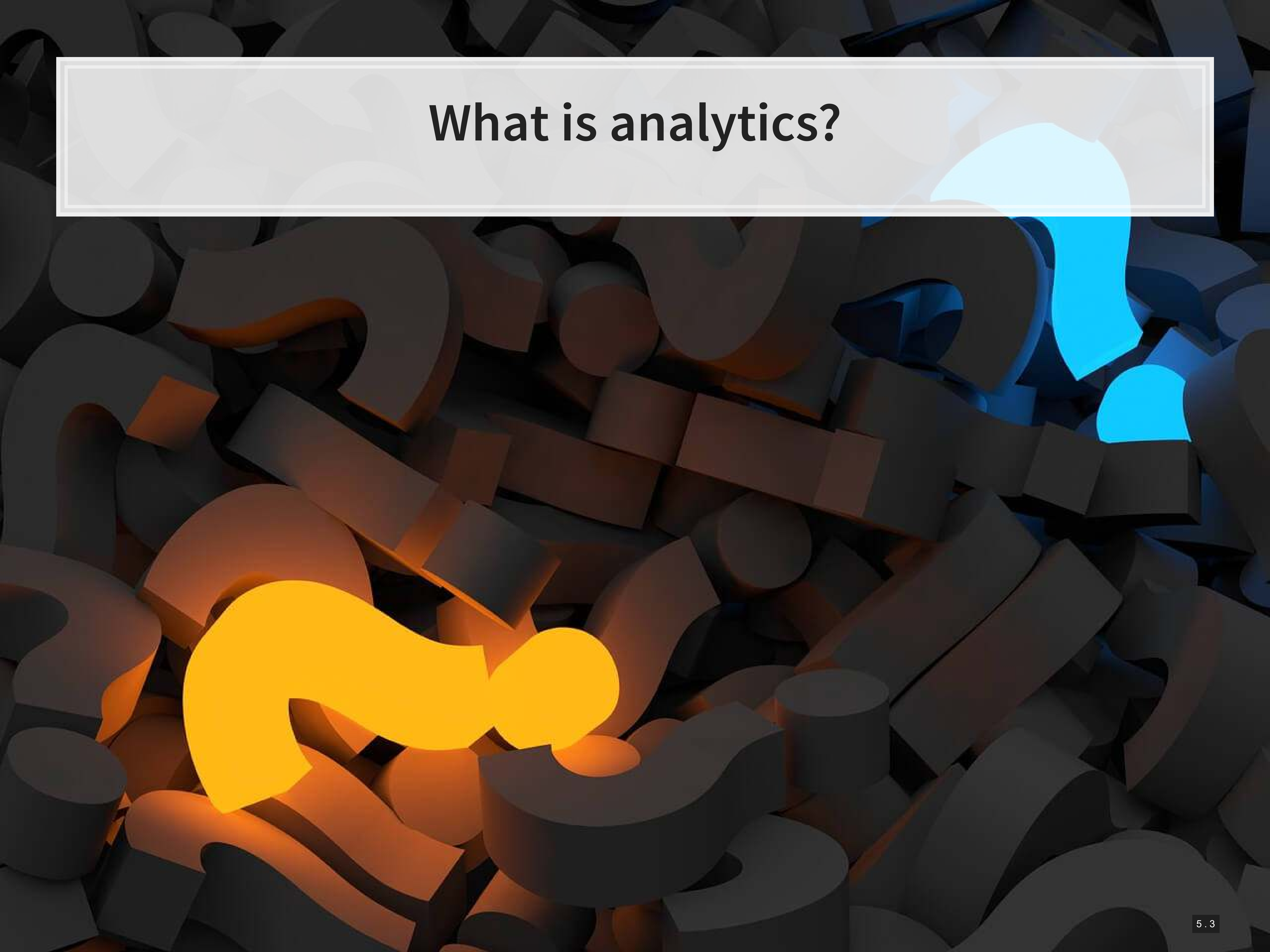
Anomaly detection

AI/ML

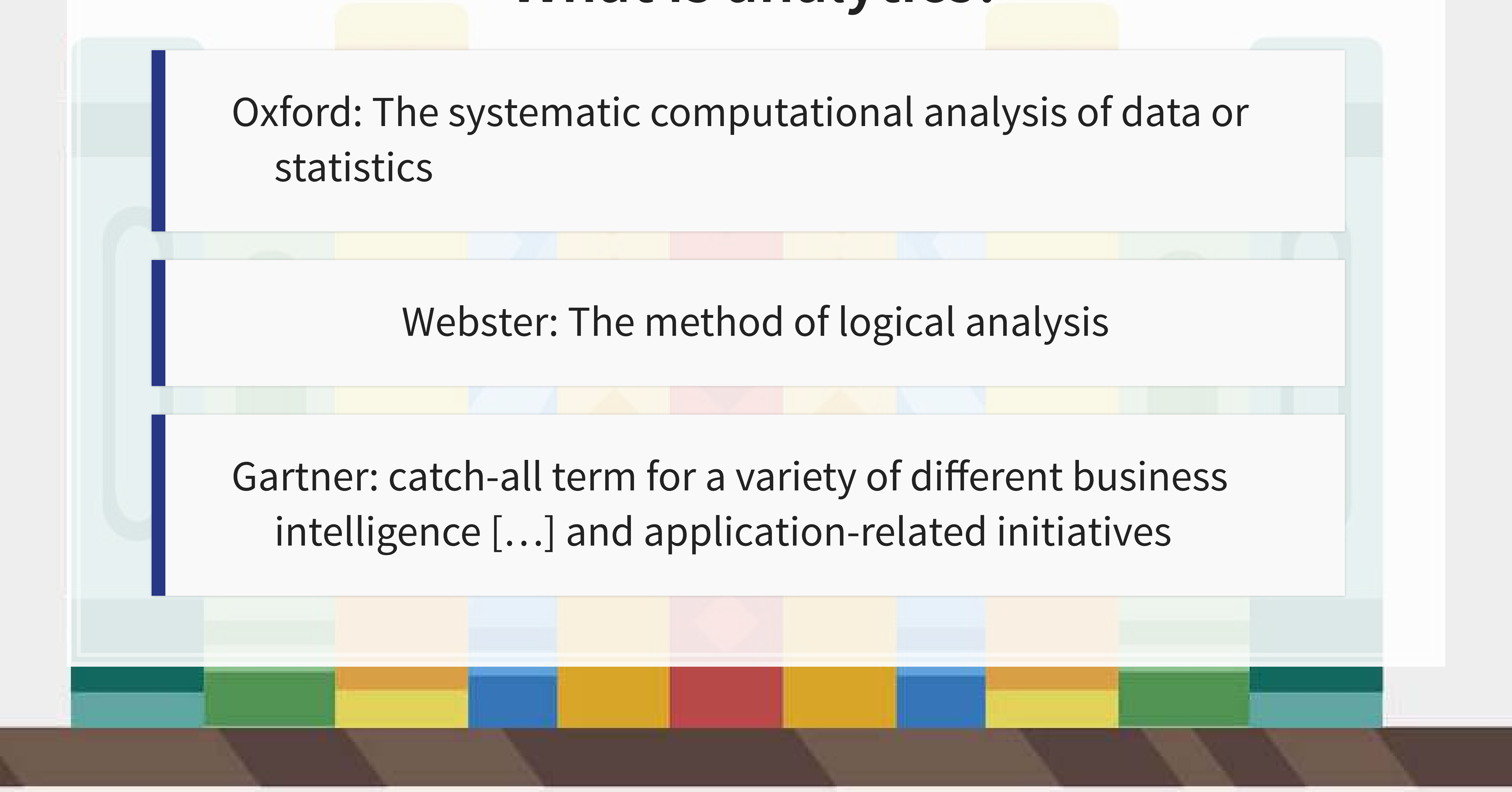
- **Theory:**
  - What is analytics?
- **Application:**
  - Who uses analytics? (and why?)
- **Methodology:**
  - Review of **R**

*Almost every class will touch on each of these three aspects

What is analytics?



What is analytics?



Oxford: The systematic computational analysis of data or statistics

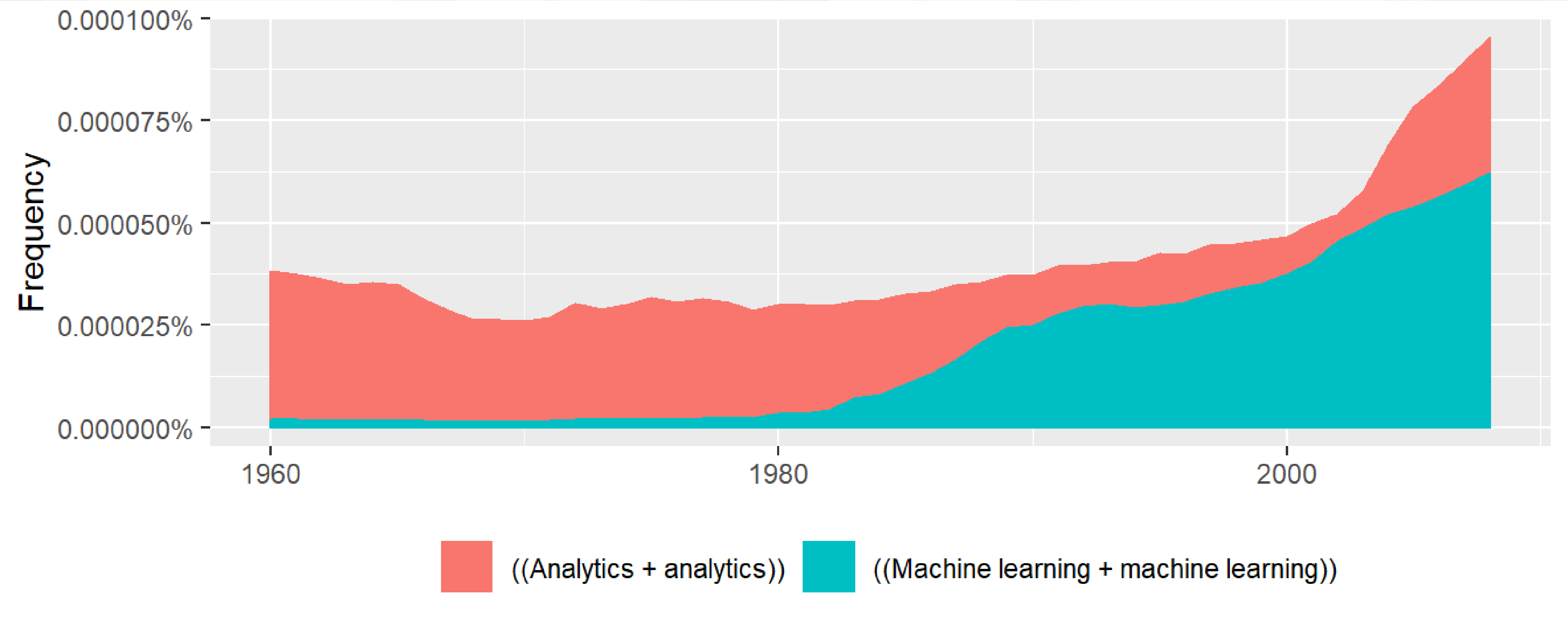
Webster: The method of logical analysis

Gartner: catch-all term for a variety of different business intelligence […] and application-related initiatives

What is analytics?

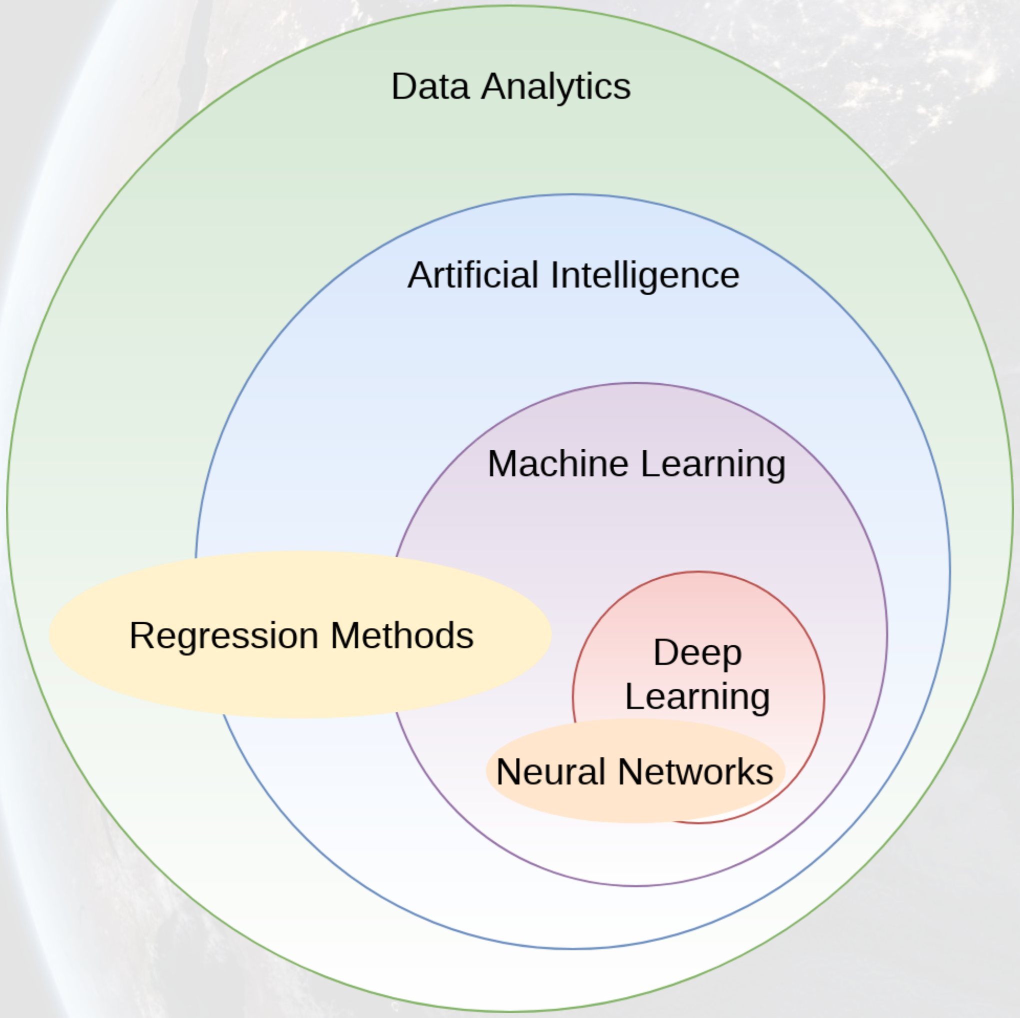
Simply put: Answering questions using data

- Additional layers we can add to the definition:
  - Answering questions using *a lot of* data
  - Answering questions using data *and statistics*
  - Answering questions using data *and computers*



Made using `ngramr`

Analytics vs AI/machine learning



- In class reading:
  - AI Will Enhance Us, Not Replace Us
  - By DataRobot's Senior Director of Product Marketing
  - Shortlink: *rmc.link/420class1*

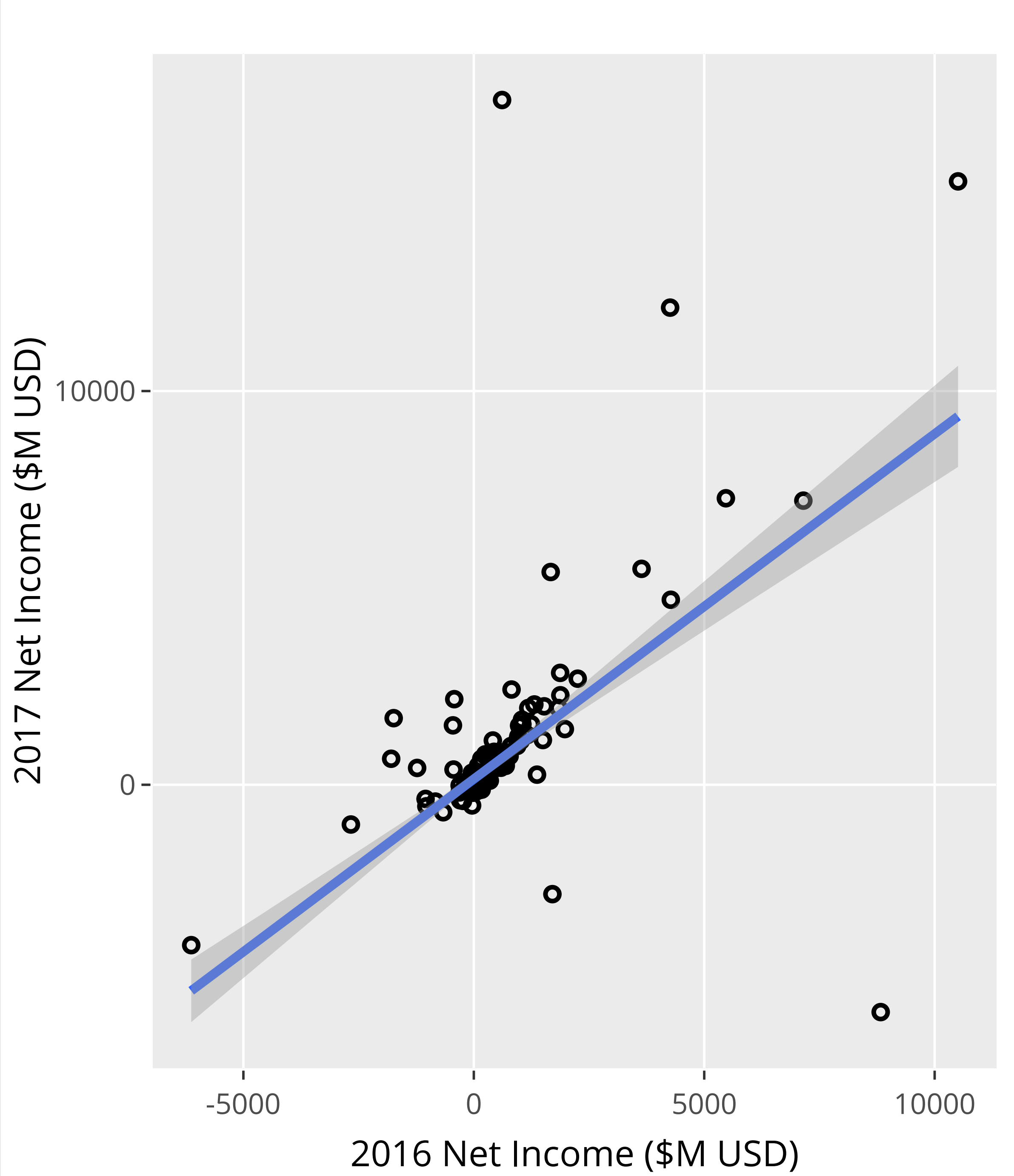
> How will Analytics/AI/ML change society and the accounting profession?

What are forecasting analytics?

- Forecasting is about making an educated guess of events to come in the future
  - Who will win the next soccer game?
  - What stock will have the best (risk-adjusted) performance?
  - What will Singtel’s earnings be next quarter?
- Leverage *past* information
  - Implicitly assumes that the past and the future predictably related

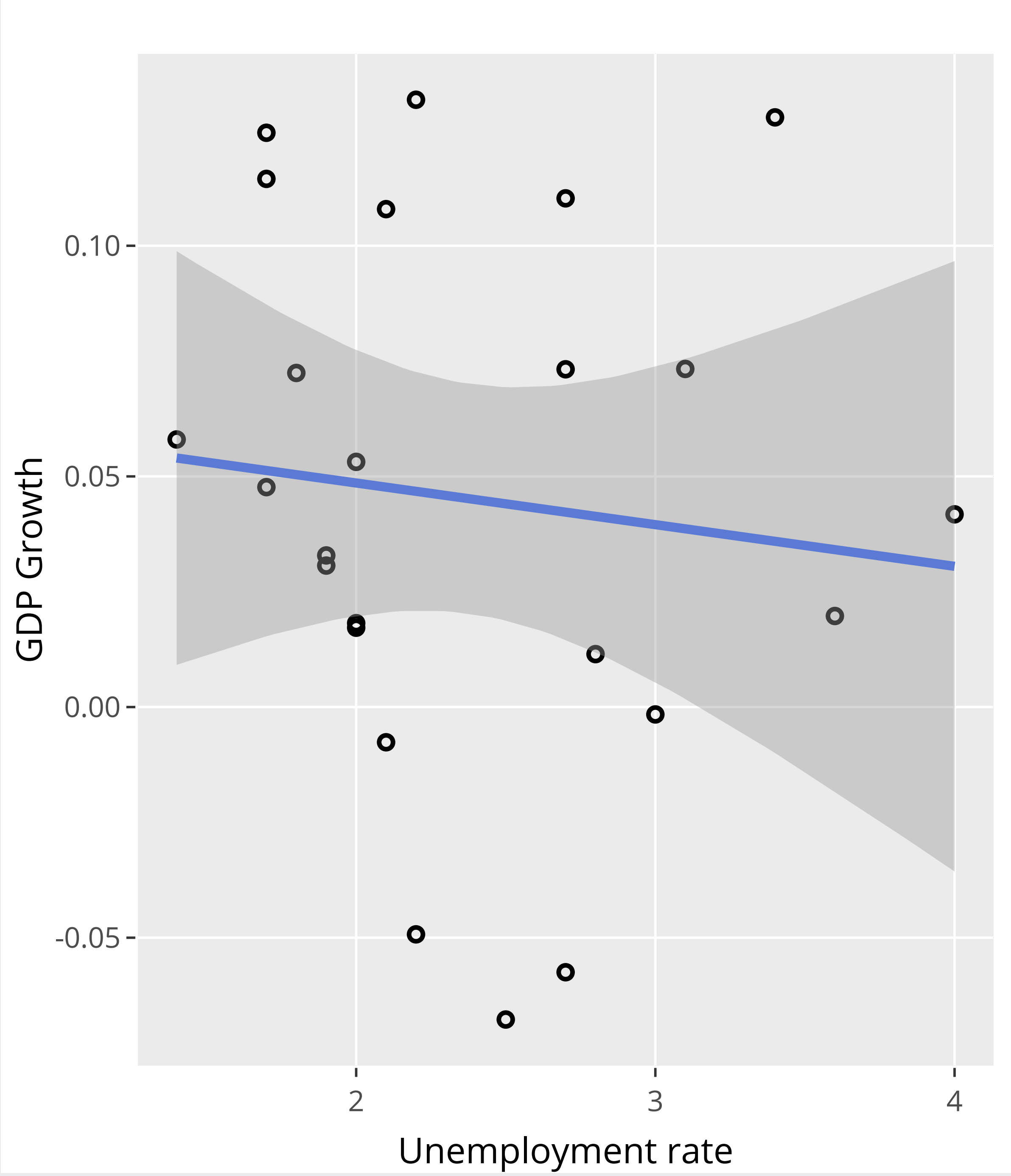
Past and future examples

- Past company earnings predicts future company earnings
    - Some earnings are stable over time (Ohlsson model)
    - Correlation: 0.7400142



Past and future examples

- Job reports predicts GDP growth in Singapore
  - Economic relationship
  - More unemployment in a year is related to lower GDP growth
    - Correlation of -0.1047259



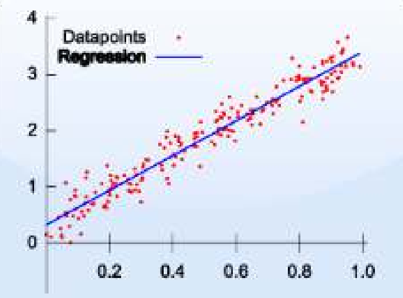
Past and future examples

- Ice cream revenue predicts pool drownings in the US
  - ???
  - Correlation is… only 0.0502886
  - What about units sold?
    - Correlation is negative!!!
    - -0.720783
  - What about price?
    - Correlation is 0.7872958

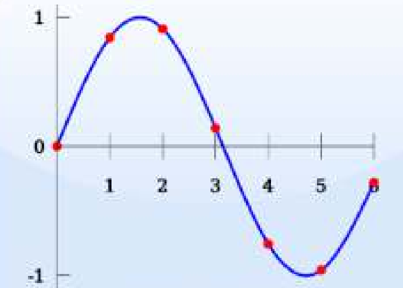
> This is where the “educated” comes in

Forecasting analytics in this class

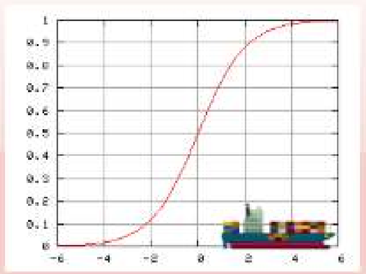
- Revenue/sales
- Shipping delays
- Bankruptcy
- Machine learning applications



Linear regression



Adv. linear regression



Logistic regression for contracting

Leveraging research for bankruptcy

AI/ML

What are forensic analytics?

- Forensic analytics focus on *detection*
  - Detecting crime such as bribery
  - Detecting fraud within companies
  - Looking at a lot of dog pictures to identify features unique to each breed

Forensic analytics in this class

- Fraud detection
- Working with textual data
- Detecting changes
- Machine learning applications

Leveraging research for bankruptcy

Lasso regression for fraud

Natural Language

Anomaly detection

AI/ML

Forecasting vs forensic analytics

- Forecasting analytics requires a time dimension
  - Predicting *future* events
- Forensic analytics is about understanding or detecting something
  - Doesn’t need a time dimension, but it can help

> These are not mutually exclusive. Forensic analytics can be used for forecasting!

Who uses analytics?

In general

- Governments
  - AI.Singapore
  - Big data office
  - “Smart” initiatives
- Academics
- Individuals!

- Companies
  - Finance
  - Manufacturing
  - Transportation
  - Computing
  - …

> 53% of companies were using big data in a 2017 survey!

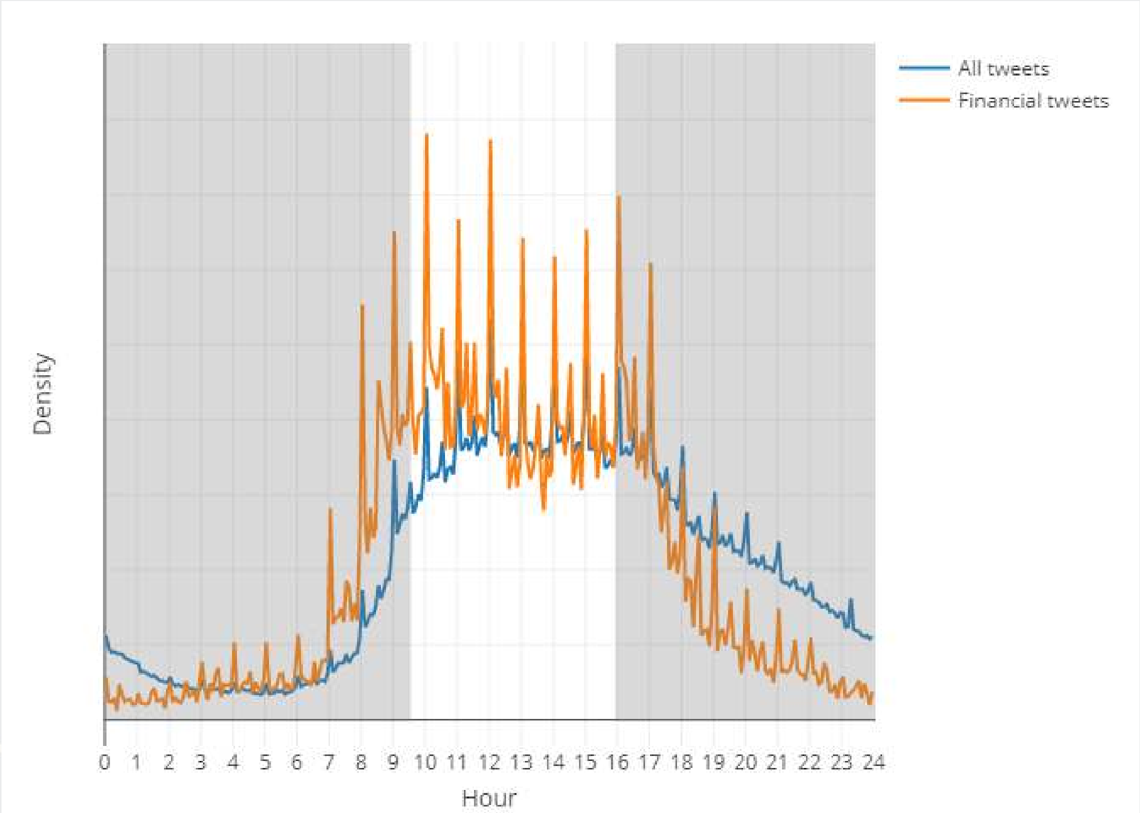
What do companies use analytics for?

- Customer service
  - Royal Bank of Scotland
    - Understanding customer complaints
- Improving products
  - Siemens’ Internet of Trains
    - Improving train reliability
- Their business
  - $18.3B USD market in 2017
    - Just a small portion of overall IT spending ($3.7T USD)

What do governments use analytics for?

- Govtech
  - Beeline
- Open data
  - Data.gov.sg
  - City of New York
- AI Singapore
  - Talent matching
    - 100 Experiments
  - AI in health Grand Challenge
  - AI research funding

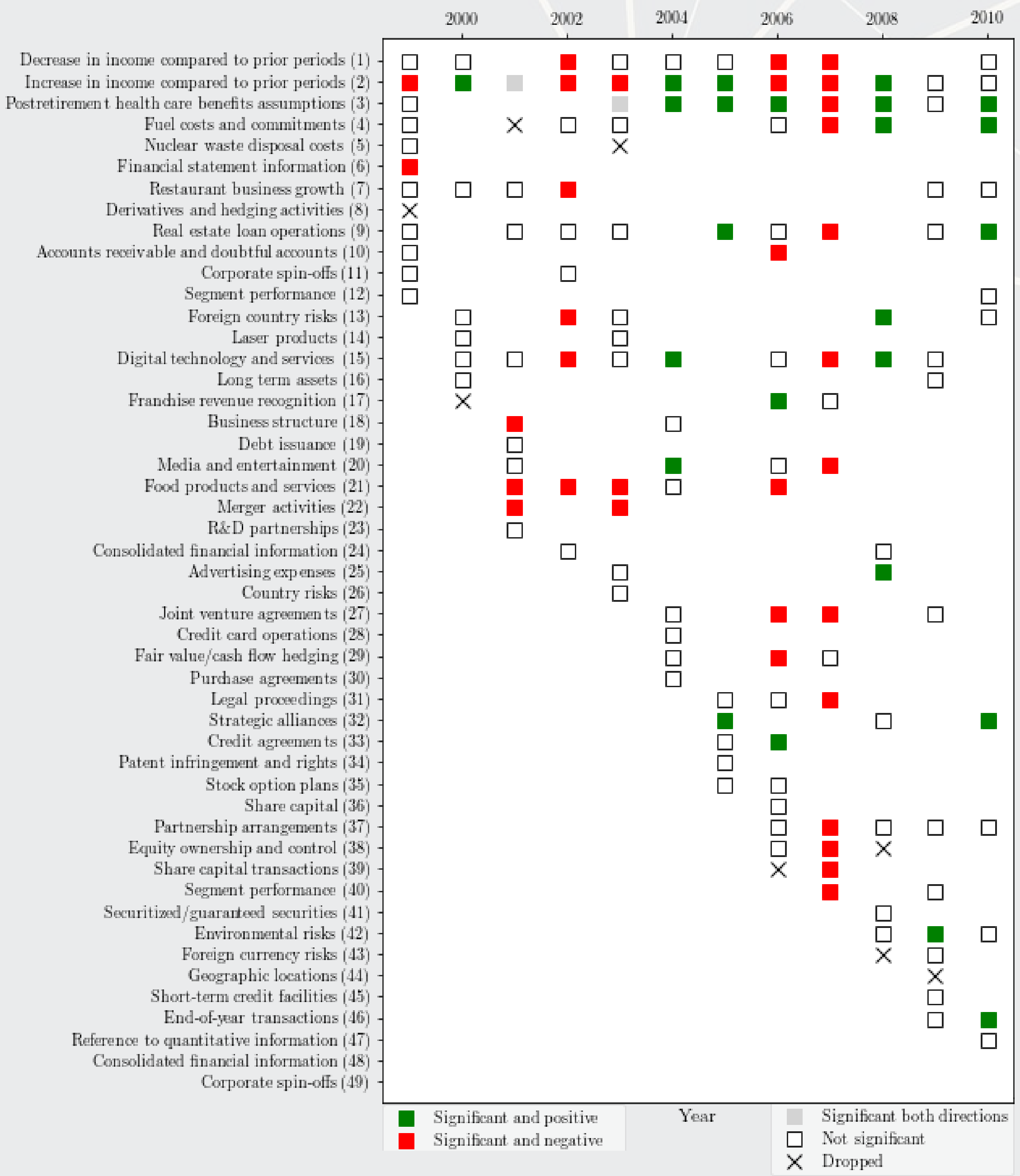
What do academics use analytics for?



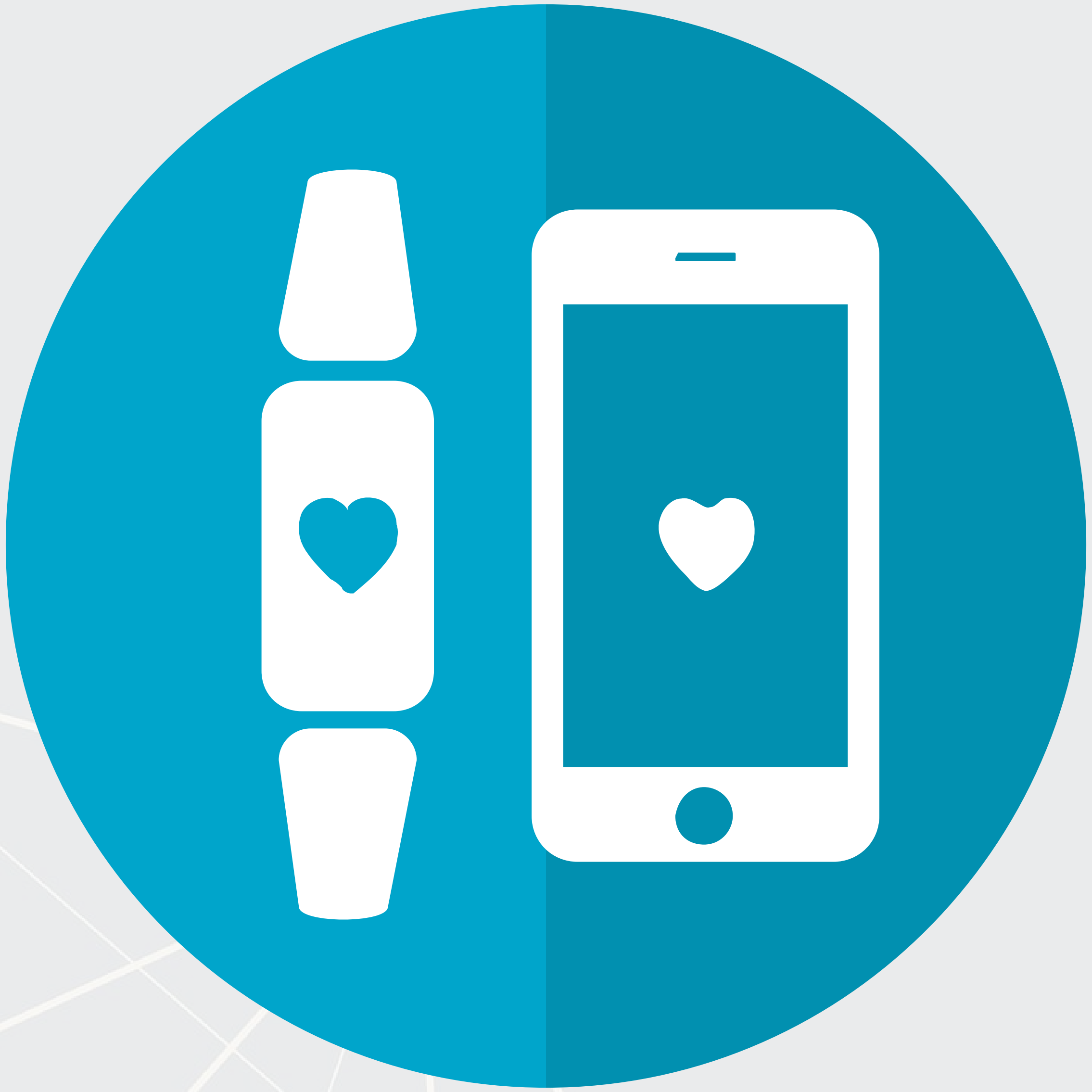
- Tweeting frequency by S&P 1500 companies (paper)
- Aggregates every tweet from 2012 to 2016
- Shows frequency in 5 minute chunks
  - Note the spikes every hour!
- The white part is the time the NYSE is open

What do academics use analytics for?

- Annual report content that predicts fraud (paper)
- For instance, discussing income is useful
  - first row is decreases, second is increases
  - But if it’s good or bad depends on the year
  - For instance, in 1999 it is a red flag
    - And one that Enron is flagged for



What do individuals use analytics for?



- Consulting
  - Radim Řehůřek: Maintainer of `gensim`, freelance consultant
- Investing
  - Quantnet discussions
- Health
  - Smart watches and other wearables

Why should you learn analytics?

- Important skill for understanding the world
  - Good timing to learn it, too!
- Gives you an edge over many others
  - Particularly useful for your career
- Jobs for “Management analysts” are expected to expand by 14% from 2016 to 2026
  - Accountants and auditors: 10%
  - Financial analysts: 11%
  - Average industry: 7%
  - All figures from US Bureau of Labor Statistics

Review of R

What is R?

- R is a “statistical programming language”
  - Focused on data handling, calculation, data analysis, and visualization
- We will use R for all work in this course

Why do we need R?

- Analytics deals with more data than we can process by hand
  - We need to ask a computer to do the work!
- R is one of the de facto standards for analytics work
  - Third most popular language for data analytics and machine learning (source)
  - Fastest growing of all mainstream languages
  - Free and open source, so you can use it anywhere
  - It can do most any analytics
  - Not a general programming language

> Programming in R provides a way of talking with the computer to make it do what you want it to do

Alternatives to R

python™

- Extremely popular
- Free and open source
- Very strong AI/ML support

TensorFlow

Microsoft Cognitive Toolkit
aka.ms/CognitiveToolkit
Microsoft

H_2O.ai

julia

- Fast and free
- Mathematics oriented
- Still young though

Scala

- Fast and free
- Focused on scalability, basis of Apache Spark

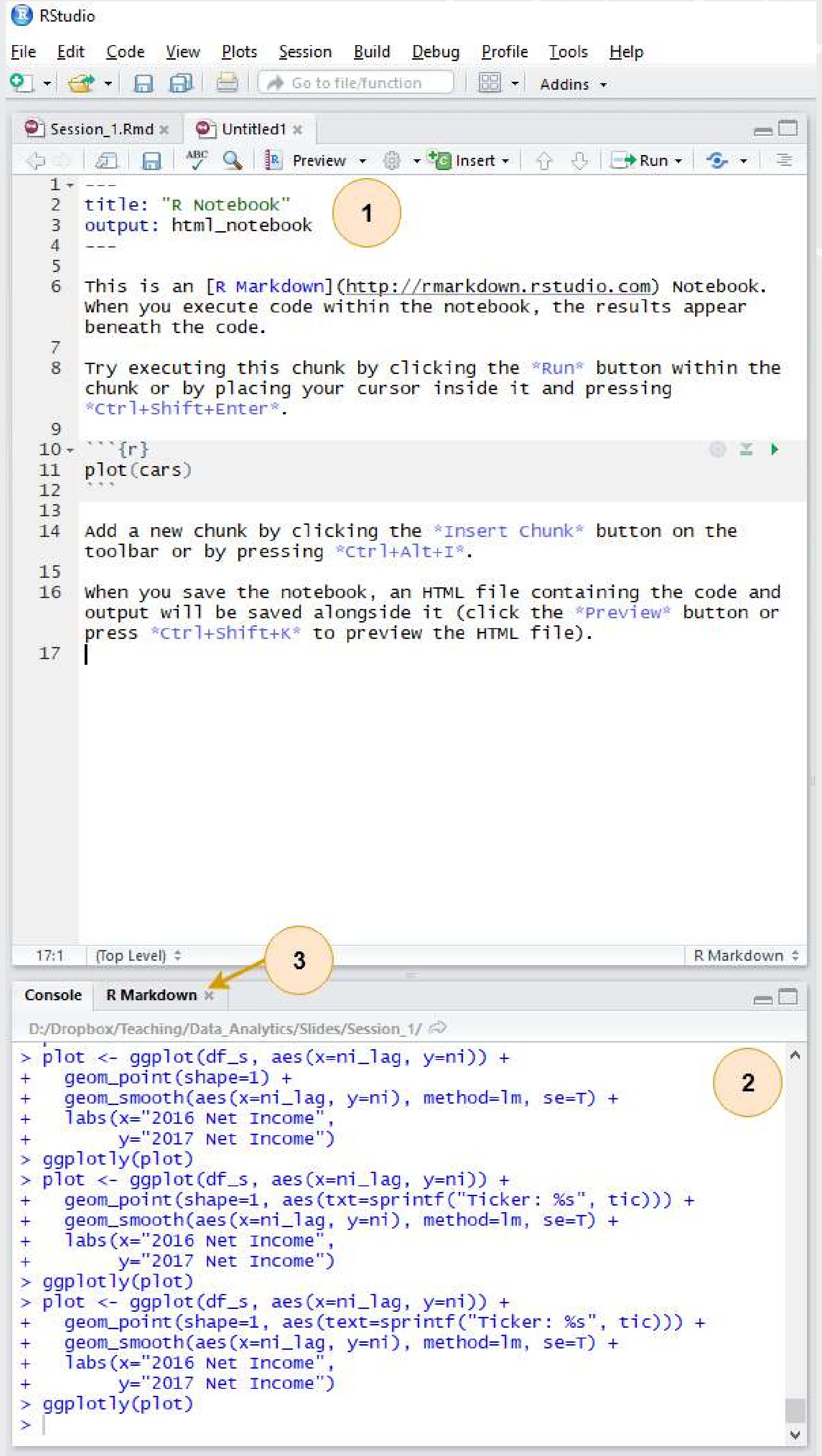
Setup for R

Setup

- For this class, I will assume you are using RStudio with the default R installation
  - RStudio downloads
  - R for Windows
  - R for (Max) OS X (Download R-3.6.1.pkg)
  - R for Linux
- For the most part, everything will work the same across all computer types
- Everything in these slides was tested on R 3.6.1 on Windows and Linux

How to use R Studio

1. R markdown file
   - You can write out reports with embedded analytics
2. Console
   - Useful for testing code and exploring your data
   - Enter your code one line at a time
3. R Markdown console
   - Shows if there are any errors when preparing your report

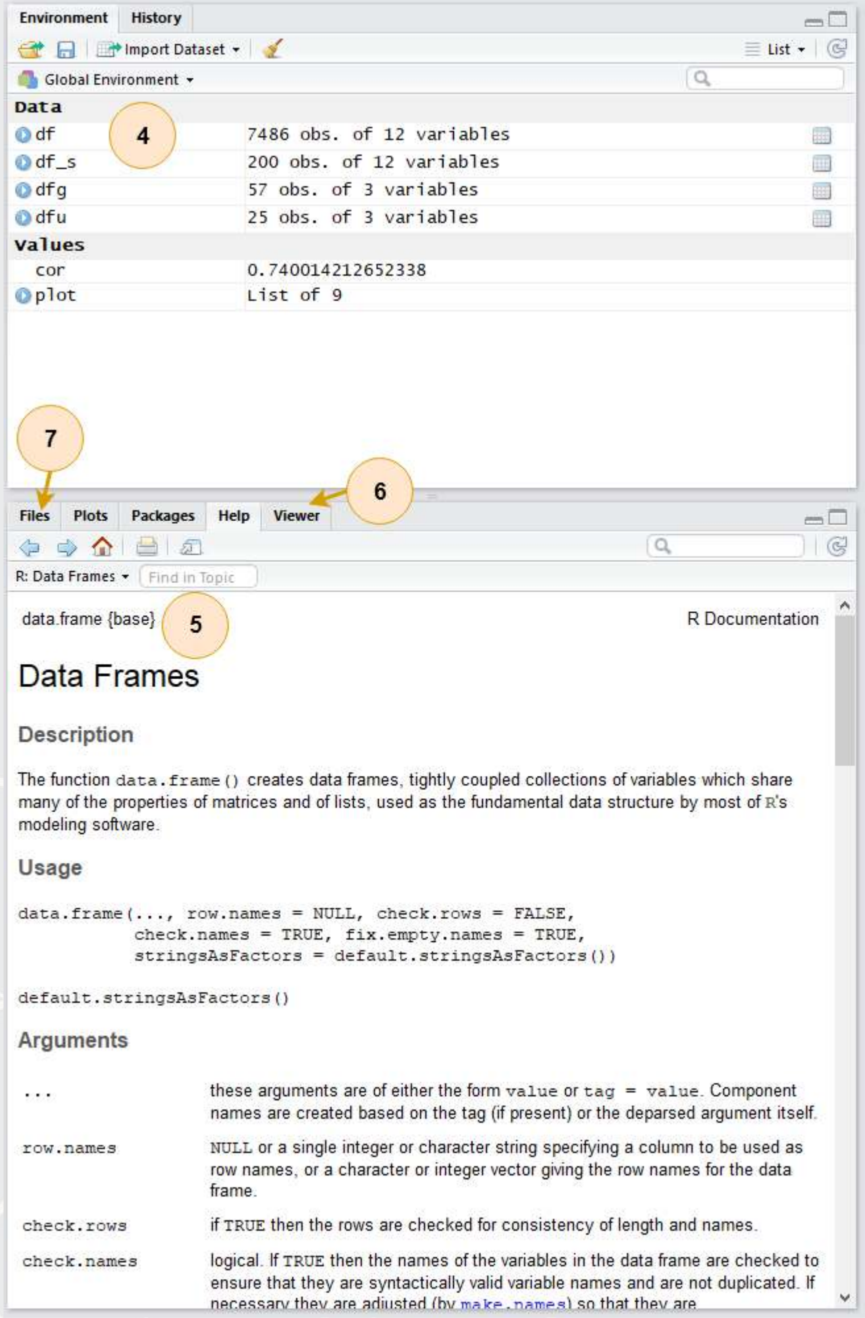


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Session_1.Rmd
Untitled1
Preview
Insert
Run
1
17:1
(Top Level)
R Markdown
3
Console
R Markdown
D:/Dropbox/Teaching/Data_Analytics/Slides/Session_1/
2

~~~
---
title: "R Notebook"
output: html_notebook
---

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook.
When you execute code within the notebook, the results appear
beneath the code.

Try executing this chunk by clicking the *Run* button within the
chunk or by placing your cursor inside it and pressing
*Ctrl+Shift+Enter*.

```{r}
plot(cars)
```


Add a new chunk by clicking the *Insert Chunk* button on the
toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and
output will be saved alongside it (click the *Preview* button or
press *Ctrl+Shift+K* to preview the HTML file).
~~~

```
> plot <- ggplot(df_s, aes(x=ni_lag, y=ni)) +
+   geom_point(shape=1) +
+   geom_smooth(aes(x=ni_lag, y=ni), method=lm, se=T) +
+   labs(x="2016 Net Income",
+        y="2017 Net Income")
> ggplotly(plot)
> plot <- ggplot(df_s, aes(x=ni_lag, y=ni)) +
+   geom_point(shape=1, aes(txt=sprintf("Ticker: %s", tic))) +
+   geom_smooth(aes(x=ni_lag, y=ni), method=lm, se=T) +
+   labs(x="2016 Net Income",
+        y="2017 Net Income")
> ggplotly(plot)
> plot <- ggplot(df_s, aes(x=ni_lag, y=ni)) +
+   geom_point(shape=1, aes(text=sprintf("Ticker: %s", tic))) +
+   geom_smooth(aes(x=ni_lag, y=ni), method=lm, se=T) +
+   labs(x="2016 Net Income",
+        y="2017 Net Income")
> ggplotly(plot)
>
```

How to use R Studio



```
Data
df        7486 obs. of 12 variables
df_s      200 obs. of 12 variables
dfg       57 obs. of 3 variables
dfu       25 obs. of 3 variables
Values
cor       0.740014212652338
plot      List of 9
```

data.frame {base} R Documentation

Data Frames

Description

The function `data.frame()` creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

Usage

```
data.frame(..., row.names = NULL, check.rows = FALSE,
           check.names = TRUE, fix.empty.names = TRUE,
           stringsAsFactors = default.stringsAsFactors())

default.stringsAsFactors()
```

Arguments

`...`	these arguments are of either the form `value` or `tag = value`. Component names are created based on the tag (if present) or the deparsed argument itself.
`row.names`	`NULL` or a single integer or character string specifying a column to be used as row names, or a character or integer vector giving the row names for the data frame.
`check.rows`	if `TRUE` then the rows are checked for consistency of length and names.
`check.names`	logical. If `TRUE` then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names and are not duplicated. If necessary they are adjusted (by `make.names`) so that they are

4. Environment
   - Shows all the values you have stored
5. Help
   - Can search documentation for instructions on how to use a function
6. Viewer
   - Shows any output you have at the moment.
7. Files
   - Shows files on your computer

Basic R commands

Arithmetic

- Anything in boxes like those on the right in my slides are R code
- The slides themselves are made in R, so you could copy and paste any code in the slides right into R to use it yourself
- Grey boxes: Code
  - Lines starting with `#` are comments
    - They only explain what the code does
- Blue boxes: Output

```r
# Addition uses '+'
1 + 1
```

```
## [1] 2
```

```r
# Subtraction uses '-'
2 - 1
```

```
## [1] 1
```

```r
# Multiplication uses '*'
3 * 3
```

```
## [1] 9
```

```r
# Division uses '/'
4 / 2
```

```
## [1] 2
```

Arithmetic

- Exponentiation
  - Write x^y as `x ^ y`
- Modulus
  - The remainder after division
  - Ex.: $46 \bmod 6 = 4$
    1. $6 \times 7 = 42$
    2. $46 - 42 = 4$
    3. $4 < 6$, so 4 is the remainder
- Integer division (not used often)
  - Like division, but it drops any decimal

```
# Exponentiation uses '^'
5 ^ 5
```

```
## [1] 3125
```

```
# Modulus (aka the remainder) uses '%
46 %% 6
```

```
## [1] 4
```

```
# Integer division uses '%/%'
46 %/% 6
```

```
## [1] 7
```

Variable assignment

- Variable assignment lets you give something a name
  - This lets you easily reuse it
- In R, we can name almost anything that we create
  - Values
  - Data
  - Functions
  - etc…
- We will name things using the `<-` command

```
# Store 2 in 'x'
x <- 2

# Check the value of x
x
```

```
## [1] 2
```

```
# Store arithmetic in y
y <- x * 2

# Check the value of y
y
```

```
## [1] 4
```

Variable assignment

- Note that values are calculated at the time of assignment
- We previously set `y <- 2 * x`
- If we change the values of `x` and `y` remain unchanged!

```
# Previous value of x and y
x
```

```
## [1] 2
```

```
y
```

```
## [1] 4
```

```
# Change x, then recheck the value
# of x and y
x <- 200

x
```

```
## [1] 200
```

```
y
```

```
## [1] 4
```

Application: Singtel’s earnings growth

> Set a variable `growth` to the amount of Singtel’s earnings growth percent in 2018

```r
# Data from Singtel's earnings reports, in Millions of SGD
singtel_2017 <- 3831.0
singtel_2018 <- 5430.3

# Compute growth
growth <- singtel_2018 / singtel_2017 - 1

# Check the value of growth
growth
```

```
## [1] 0.4174628
```

Recap

- So far, we are using R as a glorified calculator
- The key to using R is that we can scale this up with little effort
  - Calculating *every* public companies’ earnings growth isn’t much harder than calculating Singtel’s!

> Scaling this up will give use a lot more value

- How to scale up:
  1. Use data structures to hold collections of data
     - Could calculate growth for **all** companies instead of just Singtel, using the same basic structure
  2. Leverage **functions** to automate more complex operations
     - There are many functions built in, and **many** more freely available

Data structures

Data types

- Numeric: Any number
  - Positive or negative
  - With or without decimals
- Boolean: `TRUE` or `FALSE`
  - Capitalization matters!
  - Shorthand is `T` and `F`
- Character: “text in quotes”
  - More difficult to work with
  - You can use either single or double quotes
- Factor: Converts text into numeric data
  - Categorical data

```
company_name <- "Google"   # character
company_name
```

```
## [1] "Google"
```

```
company_name <- 'Google'   # character
company_name
```

```
## [1] "Google"
```

```
tech_firm <- TRUE   # boolean
tech_firm
```

```
## [1] TRUE
```

```
earnings <- 12662   # numeric, $M USD
earnings
```

```
## [1] 12662
```

Scaling up…

- We already have some data entered, but it’s only a small amount
- We can scale this up using …
  - Vectors using `c()` – holds only 1 type
  - Matrices using `matrix()`! – holds only 1 type
  - Lists using `list()`! – holds anything (including other structures)
  - Data frames using `data.frame()`! – holds different types by column

Vectors: What are they?

- Remember back to linear algebra…

Examples:

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$$

A row (or column) of data

Vector example: Profit margin for tech firms

```
# Calculating proit margin for all public US tech firms
# 715 tech firms in Compustat with >1M sales in 2017

# Data:
#    earnings_2017: vector of earnings, $M USD
#    revenue_2017: vector of revenue, $M USD
#    names_2017: a vector of tickers (strings)

# Namining the vectors
names(earnings_2017) <- names_2017
names(revenue_2017) <- names_2017

earnings_2017[1:6]
```

```
##               AVX CORP        BK TECHNOLOGIES ADVANCED MICRO DEVICES
##                  4.910                 -3.626                 43.000
##   ASM INTERNATIONAL NV SKYWORKS SOLUTIONS INC         ANALOG DEVICES
##                543.878               1010.200                727.259
```

```
revenue_2017[1:6]
```

```
##               AVX CORP        BK TECHNOLOGIES ADVANCED MICRO DEVICES
##               1562.474                 39.395               5329.000
##   ASM INTERNATIONAL NV SKYWORKS SOLUTIONS INC         ANALOG DEVICES
##                886.503               3651.400               5107.503
```

Vector example: Profit margin for tech firms

```r
# Summarizing vectors
summary(earnings_2017)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
## -4307.49   -15.98     1.84   296.84    91.36 48351.00 
```

```r
summary(revenue_2017)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
##      1.06    102.62    397.57   3023.78   1531.59 229234.00 
```

```r
# Calculating profit margin
margin <- earnings_2017 / revenue_2017
summary(margin)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
## -13.97960  -0.10253   0.01353  -0.10967   0.09295   1.02655 
```

```r
# Worst, midpoint, and best profit margin firms in 2017. Our names carried over :)
margin[order(margin)][c(1,length(margin)/2,length(margin))]
```

```
## HELIOS AND MATHESON ANALYTIC                   NLIGHT INC 
##                 -13.97960161                   0.01325588 
##            CCUR HOLDINGS INC 
##                   1.02654899 
```

Matrices: What are they?

- Remember back to linear algebra…

Example:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

> A rows *and* columns of data

Selecting from matrices

- Select using 2 indexes instead of 1:
  - `matrix_name[rows,columns]`
  - To select all rows or columns, leave that index blanks

```
columns <- c("Google", "Microsoft",
             "Goldman")
rows <- c("Earnings","Revenue")

firm_data <- matrix(data=
  c(12662, 21204, 4286, 110855,
    89950, 42254), nrow=2)
# Equivalent:
# matrix(data=c(12662, 21204, 4286,
#   110855, 89950, 42254), ncol=3)

# Apply names
rownames(firm_data) <- rows
colnames(firm_data) <- columns

# Print the matrix
firm_data
```

```
##            Google Microsoft Goldman
## Earnings  12662      4286   89950
## Revenue   21204    110855   42254
```

```
firm_data[2, 3]
```

```
## [1] 42254
```

```
firm_data[, c("Google", "Microsoft")]
```

```
##            Google Microsoft
## Earnings  12662      4286
## Revenue   21204    110855
```

```
firm_data[1,]
```

```
##    Google Microsoft   Goldman
##     12662      4286     89950
```

```
firm_data["Revenue", "Goldman"]
```

```
## [1] 42254
```

Combining matrices

- Matrices are combined top to bottom as rows with `rbind()`
- Matrices are combined side-by-side as columns with `cbind()`

```r
# Preloaded: industry codes as indcode (vector)
#     - GICS codes: 40=Financials, 45=Information Technology
#     - See: https://en.wikipedia.org/wiki/Global_Industry_Classification_Standarc
# Preloaded: JPMorgan data as jpdata (vector)

mat <- rbind(firm_data,indcode)  # Add a row
rownames(mat)[3] <- "Industry"  # Name the new row
mat
```

```
##           Google Microsoft Goldman
## Earnings  12662      4286   89950
## Revenue   21204    110855   42254
## Industry     45        45      40
```

```r
mat <- cbind(firm_data,jpdata)  # Add a column
colnames(mat)[4] <- "JPMorgan"  # Name the new column
mat
```

```
##           Google Microsoft Goldman JPMorgan
## Earnings  12662      4286   89950    17370
## Revenue   21204    110855   42254   115475
```

Lists: What are they?

- Like vectors, but with mixed types
- Generally not something we will create
- Often returned by analysis functions in R
  - Such as the linear models we will look at next week

```r
# Ignore this code for now...
model <- summary(lm(earnings ~ revenue, data=tech_df))
#Note that this function is hiding something...
model
```

```
##
## Call:
## lm(formula = earnings ~ revenue, data = tech_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16045.0     20.0    141.6    177.1  12104.6
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.837e+02  4.491e+01  -4.091 4.79e-05 ***
## revenue      1.589e-01  3.564e-03  44.585  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1166 on 713 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7356
## F-statistic:  1988 on 1 and 713 DF,  p-value: < 2.2e-16
```

Looking into lists

- Lists generally use double square brackets, `[[index]]`
  - Used for pulling individual elements out of a list
- `[[c()]]` will drill through lists, as opposed to pulling multiple values
- Single square brackets pull out elements as is
- Double square brackets extract just the element
- For 1 level, we can also use `$`

```
model["r.squared"]
```

```
## $r.squared
## [1] 0.7360059
```

```
model[["r.squared"]]
```

```
## [1] 0.7360059
```

```
model$r.squared
```

```
## [1] 0.7360059
```

```
earnings <- c(12662, 21204, 4286)
company <- c("Google", "Microsoft", "
names(earnings) <- company
earnings["Google"]
```

```
## Google
##  12662
```

```
earnings[["Google"]]
```

```
## [1] 12662
```

```
#Can't use $ with vectors
```

Structure of a list

- `str()` will tell us what’s in this list

```r
str(model)
```

```
## List of 11
##  $ call         : language lm(formula = earnings ~ revenue, data = tech_df)
##  $ terms        :Classes 'terms', 'formula'  language earnings ~ revenue
##   .. ..- attr(*, "variables")= language list(earnings, revenue)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "earnings" "revenue"
##   .. .. .. ..$ : chr "revenue"
##   .. ..- attr(*, "term.labels")= chr "revenue"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(earnings, revenue)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "earnings" "revenue"
##  $ residuals    : Named num [1:715] -59.7 173.8 -620.2 586.7 613.6 ...
##   ..- attr(*, "names")= chr [1:715] "1" "2" "3" "4" ...
##  $ coefficients : num [1:2, 1:4] -1.84e+02 1.59e-01 4.49e+01 3.56e-03 -4.09 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "(Intercept)" "revenue"
```

What are data frames?

- Data frames are like a hybrid between lists and matrices

Like a matrix:

- 2 dimensional like matrices
- Can access data with `[ ]`
- All elements in a column must be the same data type

Like a list:

- Can have different data types for different columns
- Can access data with `$`

Columns ≈ variables, e.g., earnings

Rows ≈ observations, e.g., Google in 2017

Dealing with data frames

> There are three schools of thought on this

1. Use *Base R* functions (i.e., what's built in)
   - Tends to be tedious
2. Use *tidy* methods (from `tidyverse`)
   - Almost always cleaner and more readable
   - Sometimes faster, sometimes slower
   - This creates a structure called a `tibble`
3. Use *data.table* (from `package:data.table`)
   - Very structured syntax, but difficult to read
   - Almost always fastest – use when speed is needed
   - This creates a structure called a `data.table`

> Cast either to a `data.frame` using `as.data.frame()`

Data in Base R

> Note: Base R methods are explained in the R Supplement

```r
library(tidyverse)  # Imports most tidy packages
# Base R data import -- stringsAsFactors is important here
df <- read.csv("../../Data/Session_1-2.csv", stringsAsFactors=FALSE)
df <- subset(df, fyear == 2017 & !is.na(revt) & !is.na(ni) &
             revt > 1 & gsector == 45)
df$margin = df$ni / df$revt
summary(df)
```

```
##      gvkey           datadate            fyear         indfmt         
##  Min.   :  1072   Min.   :20170630   Min.   :2017   Length:715        
##  1st Qu.: 20231   1st Qu.:20171231   1st Qu.:2017   Class :character  
##  Median : 33232   Median :20171231   Median :2017   Mode  :character  
##  Mean   : 79699   Mean   :20172029   Mean   :2017                     
##  3rd Qu.:148393   3rd Qu.:20171231   3rd Qu.:2017                     
##  Max.   :315629   Max.   :20180430   Max.   :2017                     
##                                                                       
##     consol             popsrc            datafmt         
##  Length:715         Length:715         Length:715        
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
##                                                          
##      tic                conm              curcd          
##  Length:715         Length:715         Length:715        
```

Data the tidy way

```r
# Tidy import
df <- read_csv("../../Data/Session_1-2.csv") %>%
  filter(fyear == 2017,       # fiscal year
         !is.na(revt),        # revenue not missing
         !is.na(ni),          # net income not missing
         revt > 1,            # at least 1M USD in revenue
         gsector == 45) %>%   # tech firm
  mutate(margin = ni/revt)    # profit margin
summary(df)
```

```
##     gvkey              datadate            fyear          indfmt         
##  Length:715         Min.   :20170630   Min.   :2017   Length:715        
##  Class :character   1st Qu.:20171231   1st Qu.:2017   Class :character  
##  Mode  :character   Median :20171231   Median :2017   Mode  :character  
##                     Mean   :20172029   Mean   :2017                     
##                     3rd Qu.:20171231   3rd Qu.:2017                     
##                     Max.   :20180430   Max.   :2017                     
##     consol             popsrc            datafmt         
##  Length:715         Length:715         Length:715        
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
##      tic                conm              curcd          
##  Length:715         Length:715         Length:715        
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
```

Other important tidy methods

- Sorting: use `arrange()`
- Grouping for calculations:
  - Group using `group_by()`
  - Ungroup using `ungroup()` once you are done
- Keep only a subset of variables using `select()`
- We’ll see many more along the way!

A note on syntax: Piping

> Pipe notation is never necessary and not built in to R

- Piping comes from `magrittr`
  - The `%>%` pipe is loaded with `tidyverse`
- Pipe notation is done using `%>%`
  - `Left %>% Right(arg2, ...)` is the same as `Right(Left, arg2, ...)`

> Piping can drastically improve code readability

- `magrittr` has other interesting pipes, such as `%<>%`
  - `Left %<>% Right(arg2, ...)` is the same as `Left <- Right(Left, arg2, ...)`

Tidy example without piping

> Note how unreadable this gets (but output is the same)

```r
df <- mutate(
        filter(
          read_csv("../../Data/Session_1-2.csv"),
          fyear == 2017,      # fiscal year
          !is.na(revt),       # revenue not missing
          !is.na(ni),         # net income not missing
          revt > 1,           # at least 1M USD in revenue
          gsector == 45),  # tech firm
        margin = ni/revt)   # profit margin
summary(df)
```

```
##     gvkey              datadate             fyear        indfmt
##  Length:715         Min.   :20170630   Min.   :2017   Length:715
##  Class :character   1st Qu.:20171231   1st Qu.:2017   Class :character
##  Mode  :character   Median :20171231   Median :2017   Mode  :character
##                     Mean   :20172029   Mean   :2017
##                     3rd Qu.:20171231   3rd Qu.:2017
##                     Max.   :20180430   Max.   :2017
##     consol             popsrc            datafmt
##  Length:715         Length:715         Length:715
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      tic                conm              curcd
```

Practice: Data types and structures

- This practice is to make sure you understand data types
- Do exercises 1 through 3 on today's R practice file:
  - R Practice
  - Shortlink: rmc.link/420r1

Useful functions

Reference

> Many useful functions are highlighted in the R Supplement

1. Installing and loading packages

```
# Install the tidyverse package from inside R
install.packages("tidyverse")

# Load the package
library(tidyverse)
```

2. Help functions

```
# To see a help page for a function (such as data.frame()) run either of:
help(data.frame)
?data.frame
```

```
# To see the arguments a function takes, run:
args(data.frame)
```

```
## function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE,
##     fix.empty.names = TRUE, stringsAsFactors = default.stringsAsFactors())
## NULL
```

Making your own functions!

- Use the `function()` function!
  - `my_func <- function(agruments) {code}`

> Simple function: Add 2 to a number

```
add_two <- function(n) {
  n + 2
}

add_two(500)
```

```
## [1] 502
```

Slightly more complex function example

```
mult_together <- function(n1, n2=0, square=FALSE) {
  if (!square) {
    n1 * n2
  } else {
    n1 * n1
  }
}

mult_together(5,6)
```

```
## [1] 30
```

```
mult_together(5,6,square=TRUE)
```

```
## [1] 25
```

```
mult_together(5,square=TRUE)
```

```
## [1] 25
```

Example: Currency conversion function

```r
FXRate <- function(from="USD", to="SGD", dt=Sys.Date()) {
  options("getSymbols.warning4.0"=FALSE)
  require(quantmod)
  data <- getSymbols(paste0(from, "/", to), from=dt-1, to=dt, src="oanda", auto.as
  return(data[[1]])
}
date()
```

```
## [1] "Sun Aug 18 16:31:56 2019"
```

```r
FXRate(from="USD", to="SGD")   # Today's SGD to USD rate
```

```
## [1] 1.38463
```

```r
FXRate(from="SGD", to="CNY")   # Today's SGD to CNY rate
```

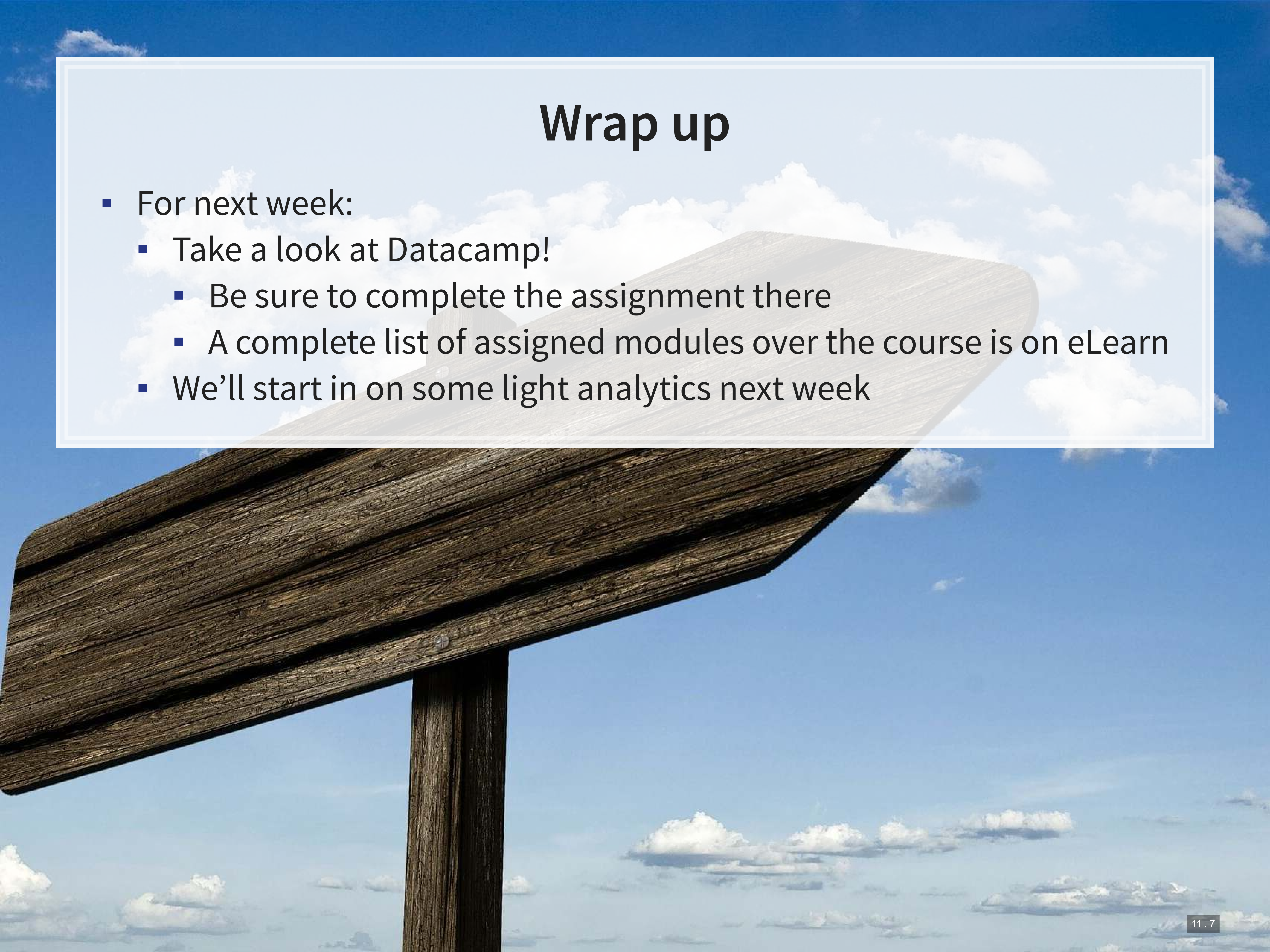
```
## [1] 5.086488
```

```r
FXRate(from="USD", to="SGD", dt=Sys.Date()-90)   # Last quarter's SGD to USD rate
```

```
## [1] 1.378014
```

Practice: Functions

- This practice is to make sure you understand functions and their construction
- Do exercises 4 and 5 on today's R practice file:
  - R Practice
  - Shortlink: rmc.link/420r1



Wrap up

- For next week:
  - Take a look at Datacamp!
    - Be sure to complete the assignment there
    - A complete list of assigned modules over the course is on eLearn
  - We’ll start in on some light analytics next week

Packages used for these slides

- `DT`
- `kableExtra`
- `knitr`
- `ngramr`
- `plotly`
- `quantmod`
- `revealjs`
- `RColorBrewer`
- `tidyverse`

Custom functions

```r
# Custom code to use Google Ngrams data
library('ngramr')
ngd <- c("(Analytics + analytics)", "(Machine learning + machine learning)")
ggram(ngd, year_start=1960, geom = "area", google_theme=F, smoothing = 3) + theme(legend.position="bottom", legend.direction="horizontal"
```

Appendix: Getting data from WRDS

Data Sources

- WRDS
  - WRDS is a provider of business data for academic purposes
  - Through your class account, you can access vast amounts of data
  - We will be particularly interested in:
    - Compustat (accounting statement data since 1950)
    - CRSP (stock price data, daily since 1926)
- We will use other public data from time to time
  - Singapore's big data repository
  - US Government data
  - Other public data collected by the Prof

Wharton UNIVERSITY of PENNSYLVANIA wrds WHARTON RESEARCH DATA SERVICES

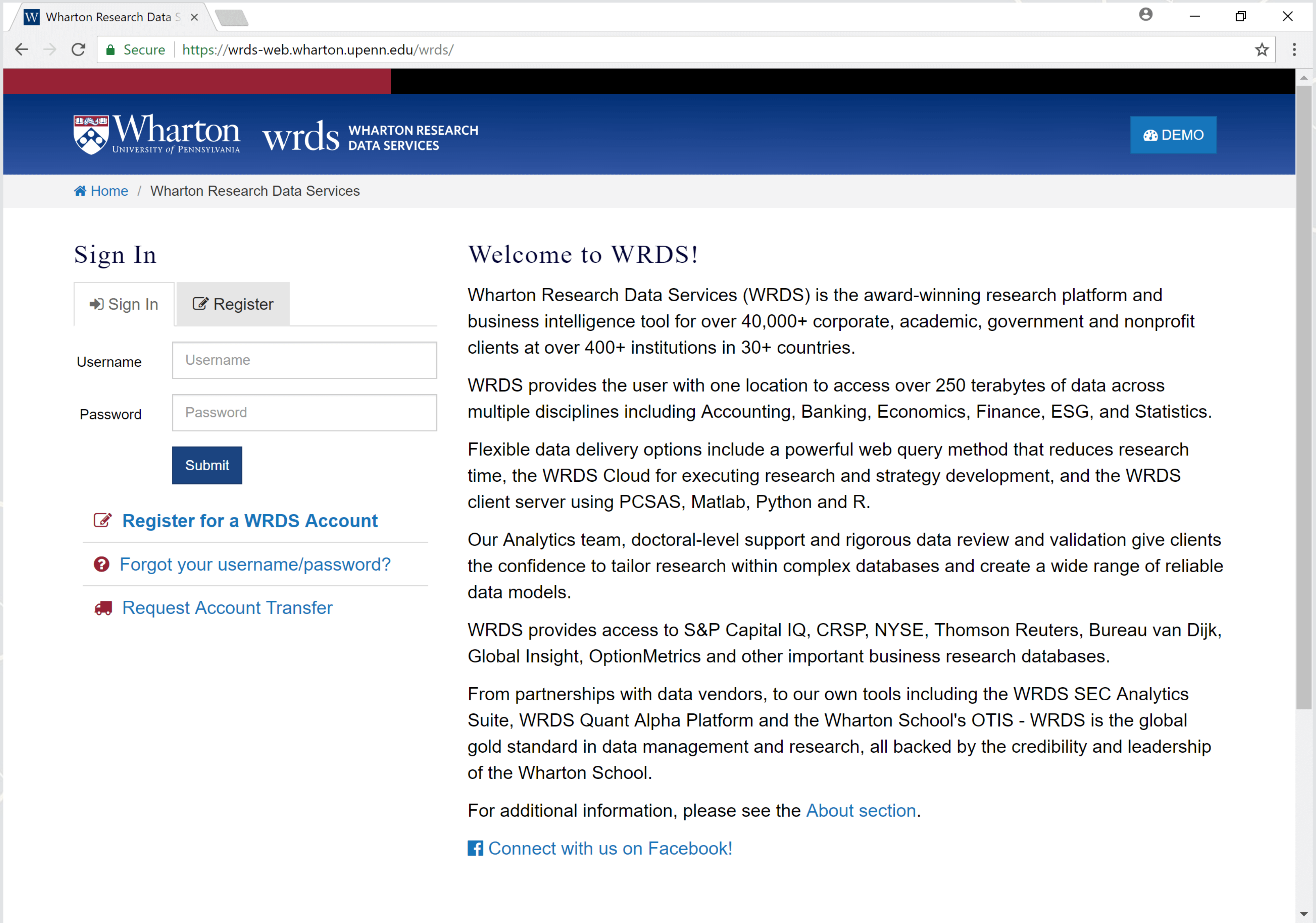
DATA.GOV

How to download from WRDS

1. Log in using a class account (posted on eLearn)
2. Pick the data provider that has your needed data
3. Select the data set you would like (some data sets only)
4. Apply any needed conditional restrictions (years, etc.)
    - These can help keep data sizes manageable
        - CRSP without any restrictions is >10 GB
5. Select the specific variables you would like export
6. Export as a csv file, zipped csv file (or other format)

Picture walkthrough for WRDS

Go to WRDS and sign in



Wharton Research Data S ×
Secure https://wrds-web.wharton.upenn.edu/wrds/
Wharton University of Pennsylvania
wrds WHARTON RESEARCH DATA SERVICES
DEMO
Home / Wharton Research Data Services

Sign In

Username: Username

Password: Password

Register for a WRDS Account

Forgot your username/password?

Request Account Transfer

Welcome to WRDS!

Wharton Research Data Services (WRDS) is the award-winning research platform and business intelligence tool for over 40,000+ corporate, academic, government and nonprofit clients at over 400+ institutions in 30+ countries.

WRDS provides the user with one location to access over 250 terabytes of data across multiple disciplines including Accounting, Banking, Economics, Finance, ESG, and Statistics.

Flexible data delivery options include a powerful web query method that reduces research time, the WRDS Cloud for executing research and strategy development, and the WRDS client server using PCSAS, Matlab, Python and R.

Our Analytics team, doctoral-level support and rigorous data review and validation give clients the confidence to tailor research within complex databases and create a wide range of reliable data models.

WRDS provides access to S&P Capital IQ, CRSP, NYSE, Thomson Reuters, Bureau van Dijk, Global Insight, OptionMetrics and other important business research databases.

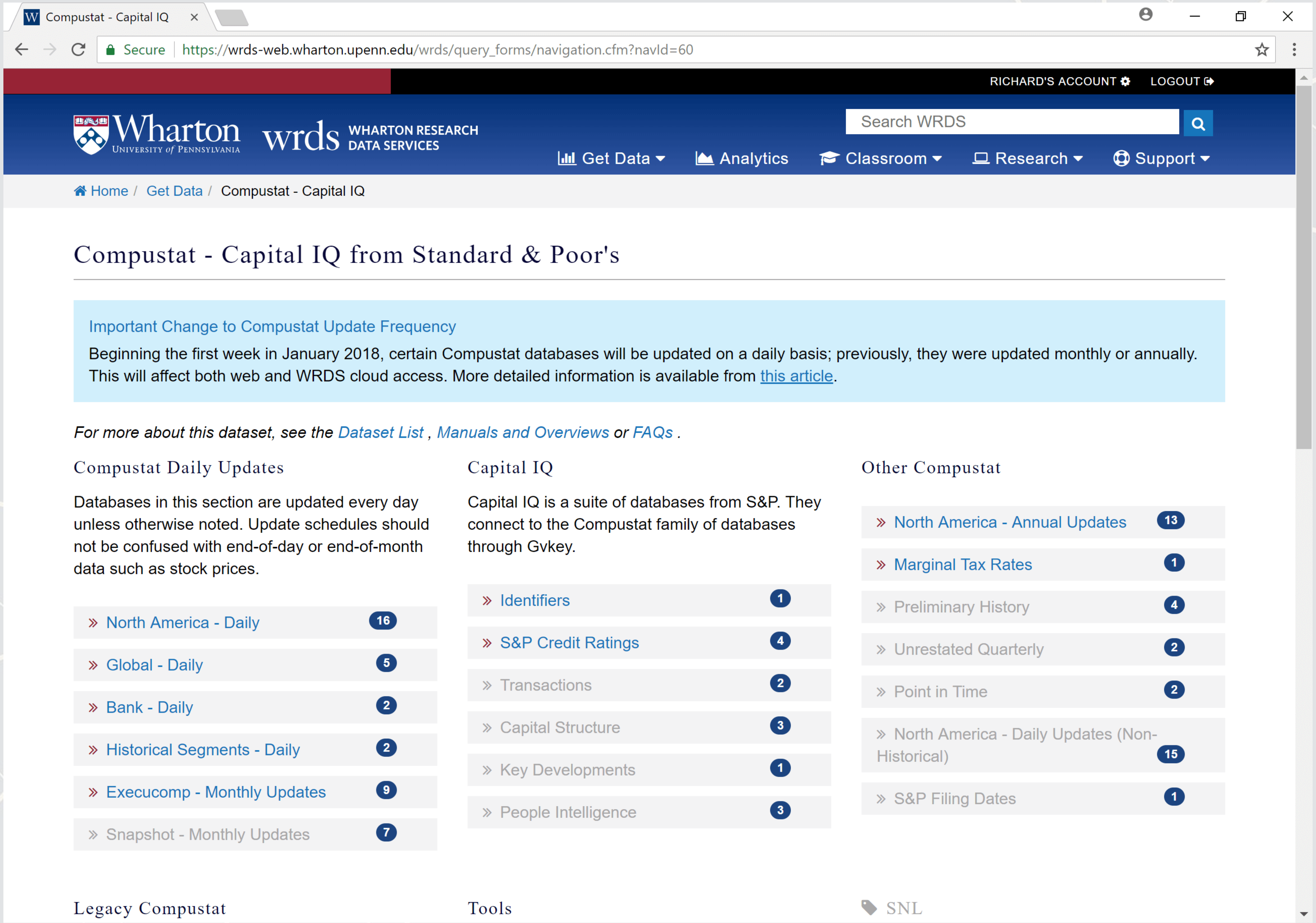
From partnerships with data vendors, to our own tools including the WRDS SEC Analytics Suite, WRDS Quant Alpha Platform and the Wharton School's OTIS - WRDS is the global gold standard in data management and research, all backed by the credibility and leadership of the Wharton School.

For additional information, please see the About section.

Connect with us on Facebook!

Pick a data provider, e.g. “Compustat - Capital IQ”

Pick a data set, e.g. "North America - Daily"



Compustat - Capital IQ
Secure
https://wrds-web.wharton.upenn.edu/wrds/query_forms/navigation.cfm?navId=60
RICHARD'S ACCOUNT
LOGOUT
Wharton
UNIVERSITY of PENNSYLVANIA
wrds WHARTON RESEARCH DATA SERVICES
Search WRDS
Get Data
Analytics
Classroom
Research
Support
Home / Get Data / Compustat - Capital IQ

Compustat - Capital IQ from Standard & Poor's

Important Change to Compustat Update Frequency

Beginning the first week in January 2018, certain Compustat databases will be updated on a daily basis; previously, they were updated monthly or annually. This will affect both web and WRDS cloud access. More detailed information is available from this article.

For more about this dataset, see the Dataset List , Manuals and Overviews or FAQs .

Compustat Daily Updates

Databases in this section are updated every day unless otherwise noted. Update schedules should not be confused with end-of-day or end-of-month data such as stock prices.

- » North America - Daily 16
- » Global - Daily 5
- » Bank - Daily 2
- » Historical Segments - Daily 2
- » Execucomp - Monthly Updates 9
- » Snapshot - Monthly Updates 7

Capital IQ

Capital IQ is a suite of databases from S&P. They connect to the Compustat family of databases through Gvkey.

- » Identifiers 1
- » S&P Credit Ratings 4
- » Transactions 2
- » Capital Structure 3
- » Key Developments 1
- » People Intelligence 3

Other Compustat

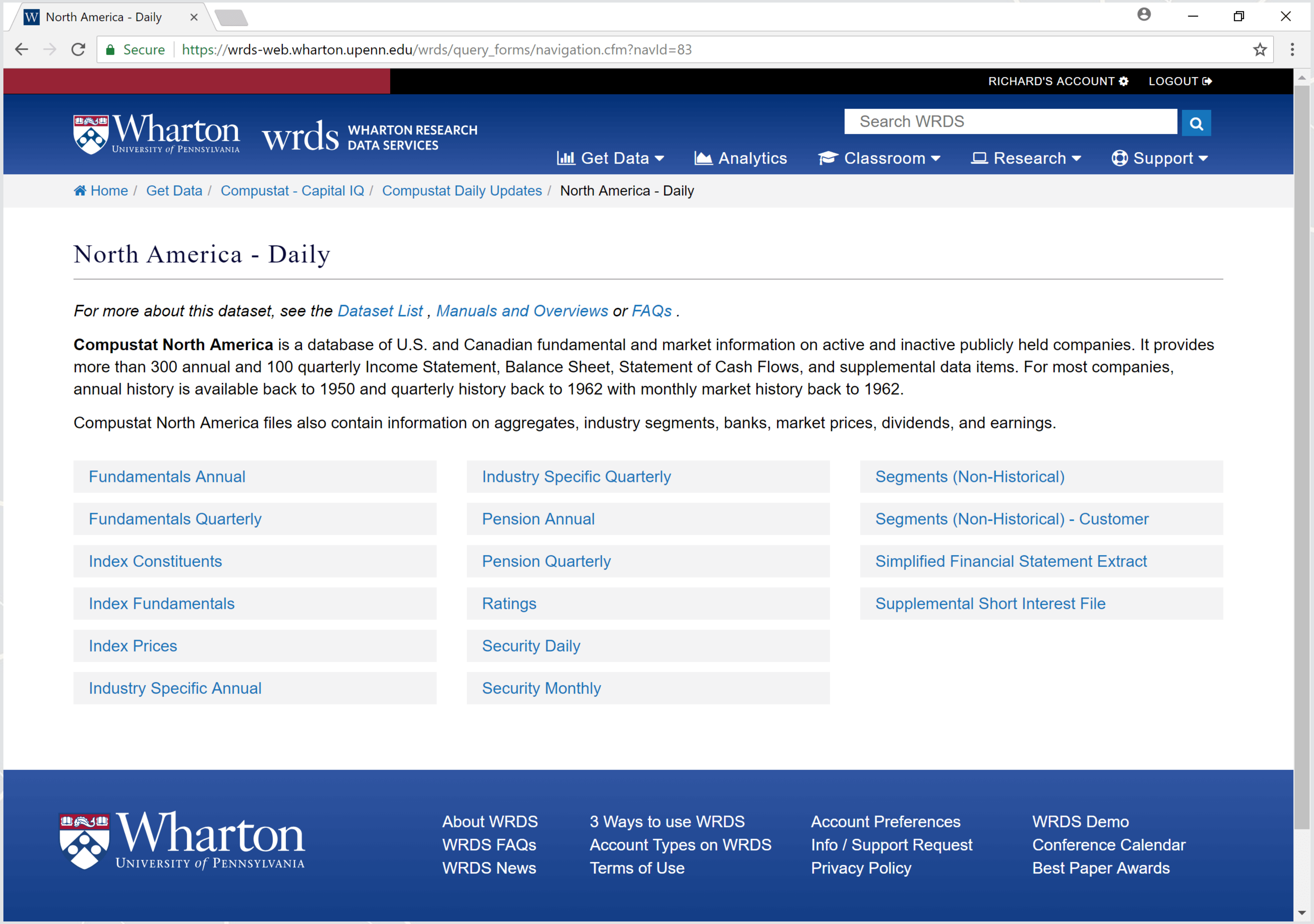
- » North America - Annual Updates 13
- » Marginal Tax Rates 1
- » Preliminary History 4
- » Unrestated Quarterly 2
- » Point in Time 2
- » North America - Daily Updates (Non-Historical) 15
- » S&P Filing Dates 1

Legacy Compustat

Tools

SNL

Pick a data set, e.g. “Fundamentals Annual”



North America - Daily
Secure
https://wrds-web.wharton.upenn.edu/wrds/query_forms/navigation.cfm?navId=83
RICHARD'S ACCOUNT
LOGOUT
Wharton University of Pennsylvania
wrds WHARTON RESEARCH DATA SERVICES
Search WRDS
Get Data
Analytics
Classroom
Research
Support

Home / Get Data / Compustat - Capital IQ / Compustat Daily Updates / North America - Daily

North America - Daily

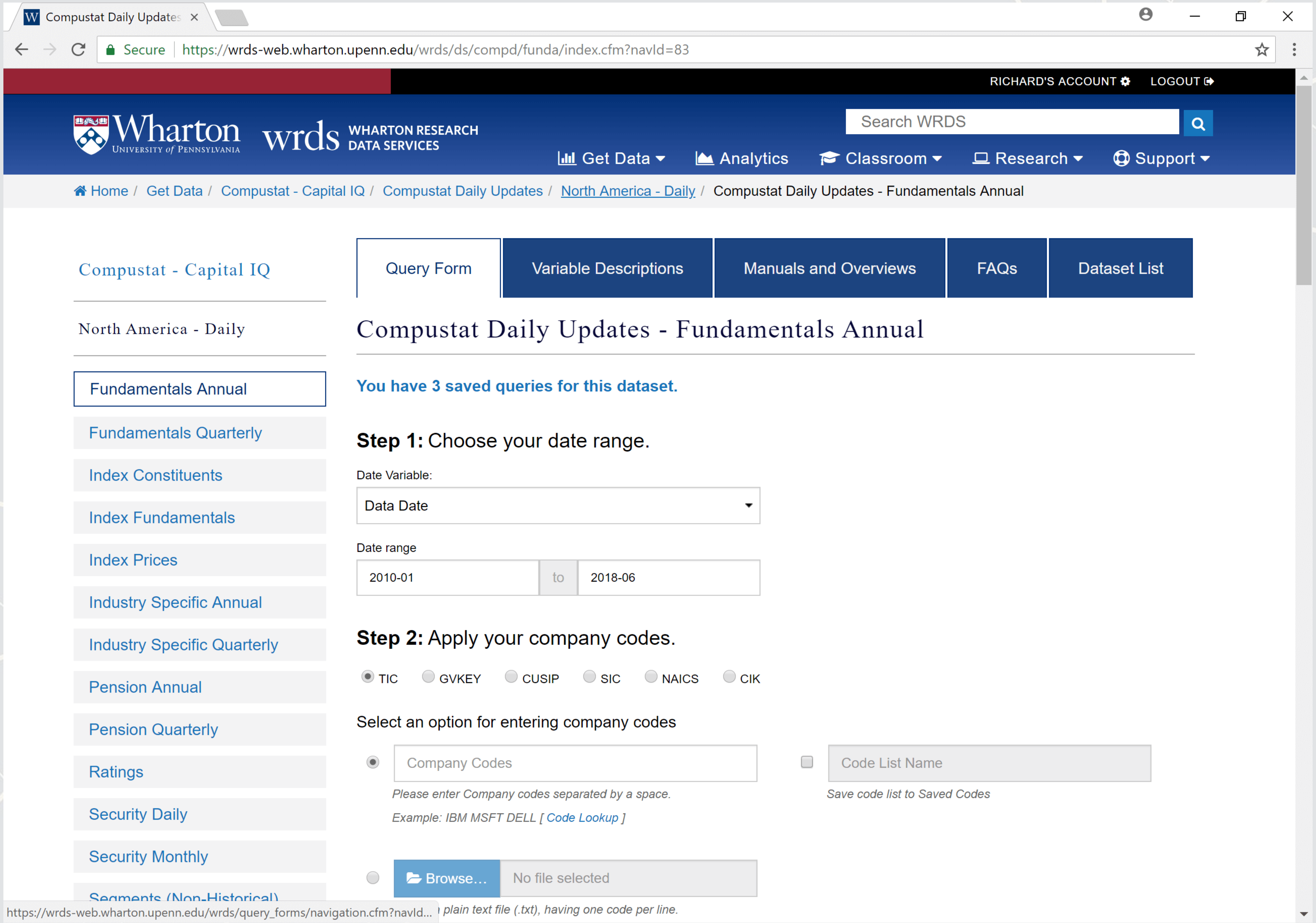
For more about this dataset, see the Dataset List , Manuals and Overviews or FAQs .

Compustat North America is a database of U.S. and Canadian fundamental and market information on active and inactive publicly held companies. It provides more than 300 annual and 100 quarterly Income Statement, Balance Sheet, Statement of Cash Flows, and supplemental data items. For most companies, annual history is available back to 1950 and quarterly history back to 1962 with monthly market history back to 1962.

Compustat North America files also contain information on aggregates, industry segments, banks, market prices, dividends, and earnings.

- Fundamentals Annual
- Fundamentals Quarterly
- Index Constituents
- Index Fundamentals
- Index Prices
- Industry Specific Annual
- Industry Specific Quarterly
- Pension Annual
- Pension Quarterly
- Ratings
- Security Daily
- Security Monthly
- Segments (Non-Historical)
- Segments (Non-Historical) - Customer
- Simplified Financial Statement Extract
- Supplemental Short Interest File

Selecting data: Time range



Compustat Daily Updates ×
Secure | https://wrds-web.wharton.upenn.edu/wrds/ds/compd/funda/index.cfm?navId=83
RICHARD'S ACCOUNT LOGOUT
Wharton University of Pennsylvania
wrds WHARTON RESEARCH DATA SERVICES
Search WRDS
Get Data
Analytics
Classroom
Research
Support

Home / Get Data / Compustat - Capital IQ / Compustat Daily Updates / North America - Daily / Compustat Daily Updates - Fundamentals Annual

Compustat Daily Updates - Fundamentals Annual

You have 3 saved queries for this dataset.

Step 1: Choose your date range.

Date Variable: Data Date

Date range: 2010-01 to 2018-06

Step 2: Apply your company codes.

● TIC ○ GVKEY ○ CUSIP ○ SIC ○ NAICS ○ CIK

Select an option for entering company codes

● Company Codes □ Code List Name

Please enter Company codes separated by a space.

Example: IBM MSFT DELL [Code Lookup]

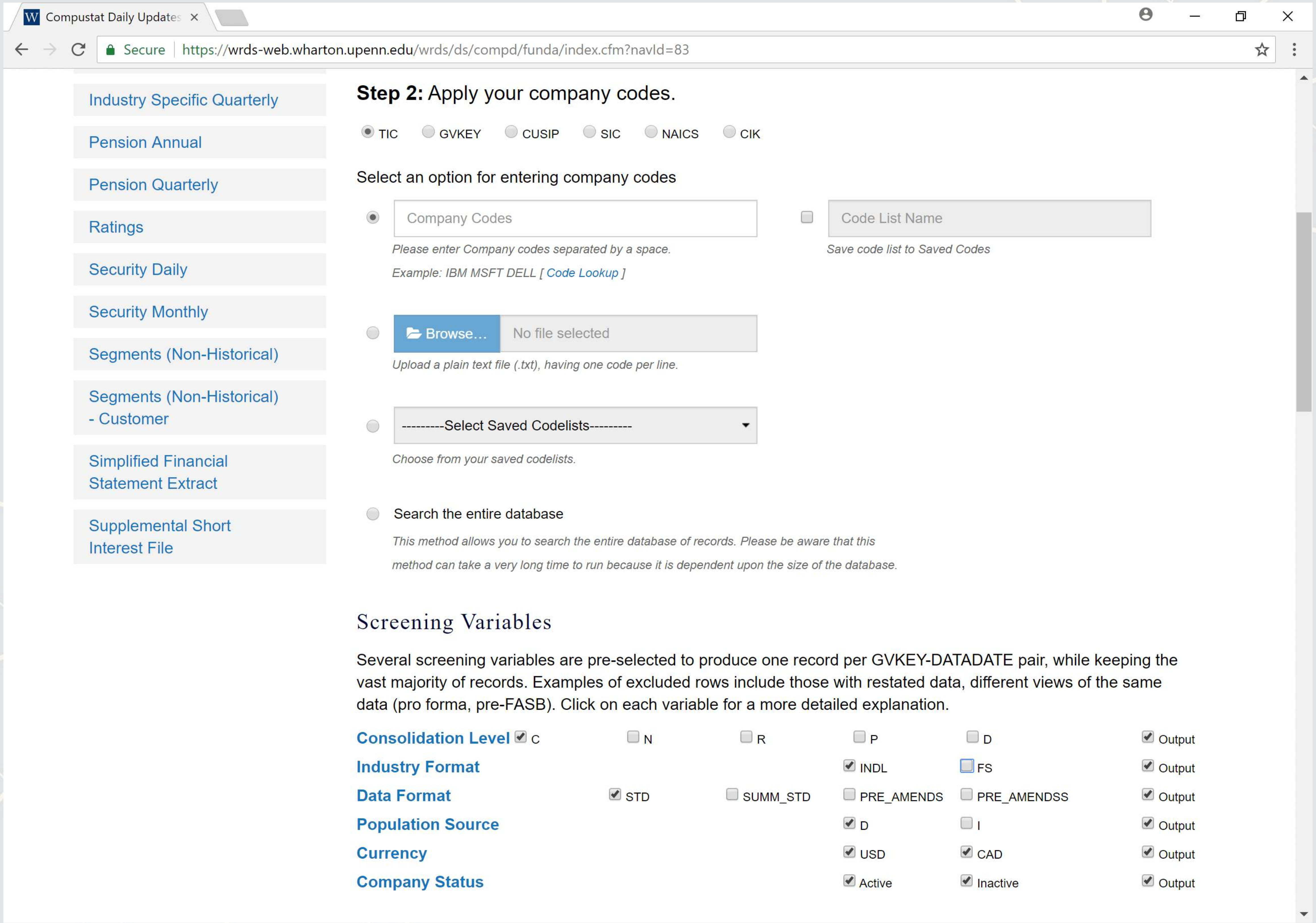
Save code list to Saved Codes

○ Browse… No file selected

plain text file (.txt), having one code per line.

https://wrds-web.wharton.upenn.edu/wrds/query_forms/navigation.cfm?navId...

Selecting data: Companies and data format



Compustat Daily Updates ×
← → C 🔒 Secure | https://wrds-web.wharton.upenn.edu/wrds/ds/compd/funda/index.cfm?navId=83

- Industry Specific Quarterly
- Pension Annual
- Pension Quarterly
- Ratings
- Security Daily
- Security Monthly
- Segments (Non-Historical)
- Segments (Non-Historical) - Customer
- Simplified Financial Statement Extract
- Supplemental Short Interest File

Step 2: Apply your company codes.

● TIC ○ GVKEY ○ CUSIP ○ SIC ○ NAICS ○ CIK

Select an option for entering company codes

● Company Codes □ Code List Name

Please enter Company codes separated by a space.

Example: IBM MSFT DELL [Code Lookup]

Save code list to Saved Codes

○ Browse… No file selected

Upload a plain text file (.txt), having one code per line.

○ ---------Select Saved Codelists---------

Choose from your saved codelists.

○ Search the entire database

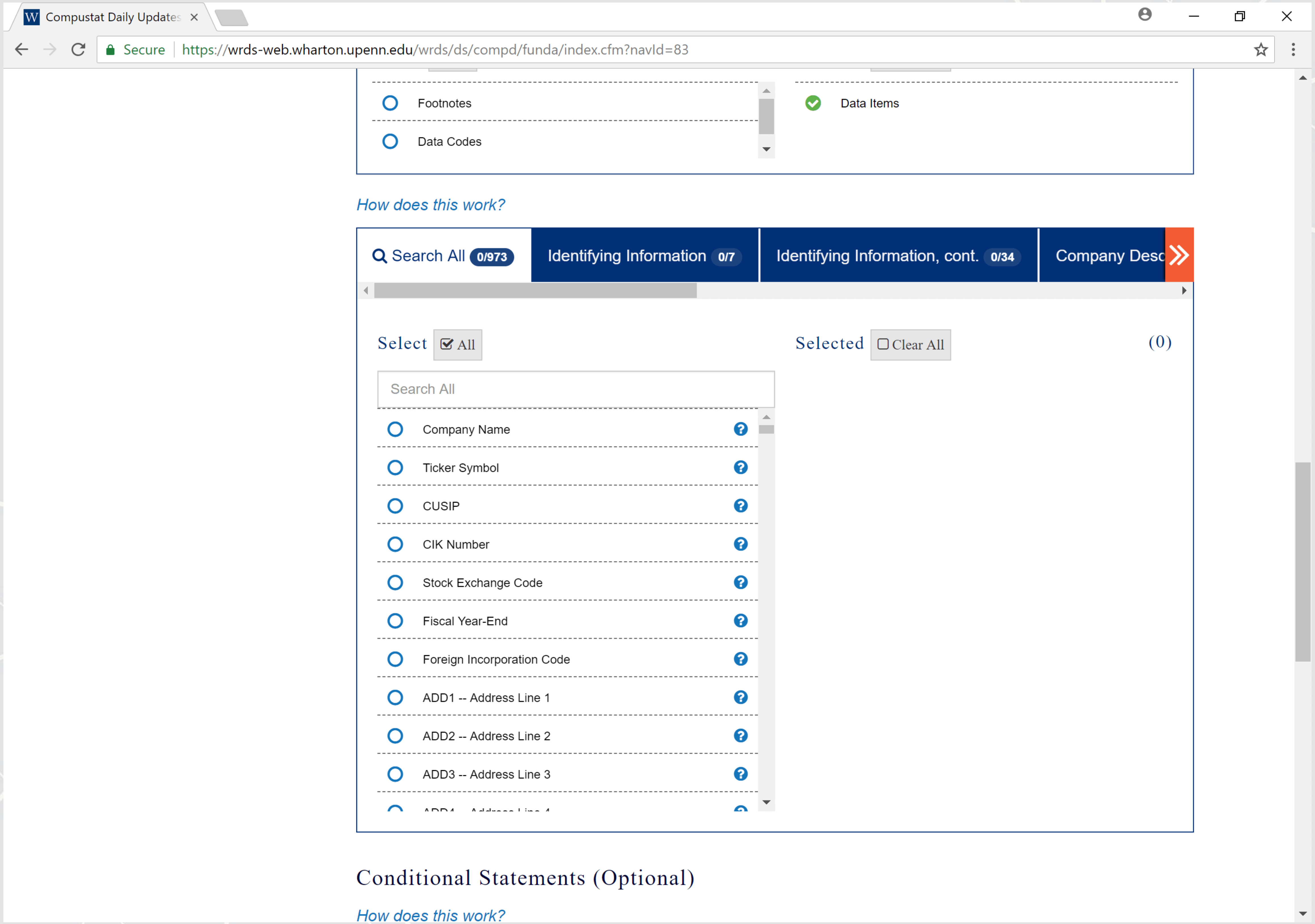
This method allows you to search the entire database of records. Please be aware that this method can take a very long time to run because it is dependent upon the size of the database.

Screening Variables

Several screening variables are pre-selected to produce one record per GVKEY-DATADATE pair, while keeping the vast majority of records. Examples of excluded rows include those with restated data, different views of the same data (pro forma, pre-FASB). Click on each variable for a more detailed explanation.

Consolidation Level ☑ C	□ N	□ R	□ P	□ D		☑ Output
Industry Format			☑ INDL	□ FS		☑ Output
Data Format	☑ STD	□ SUMM_STD	□ PRE_AMENDS	□ PRE_AMENDSS		☑ Output
Population Source			☑ D	□ I		☑ Output
Currency			☑ USD	☑ CAD		☑ Output
Company Status			☑ Active	☑ Inactive		☑ Output

Selecting data fields



Compustat Daily Updates ×
Secure
https://wrds-web.wharton.upenn.edu/wrds/ds/compd/funda/index.cfm?navId=83

○ Footnotes
○ Data Codes
✓ Data Items

How does this work?

Search All 0/973 | Identifying Information 0/7 | Identifying Information, cont. 0/34 | Company Desc

Select ☑ All

Selected □ Clear All (0)

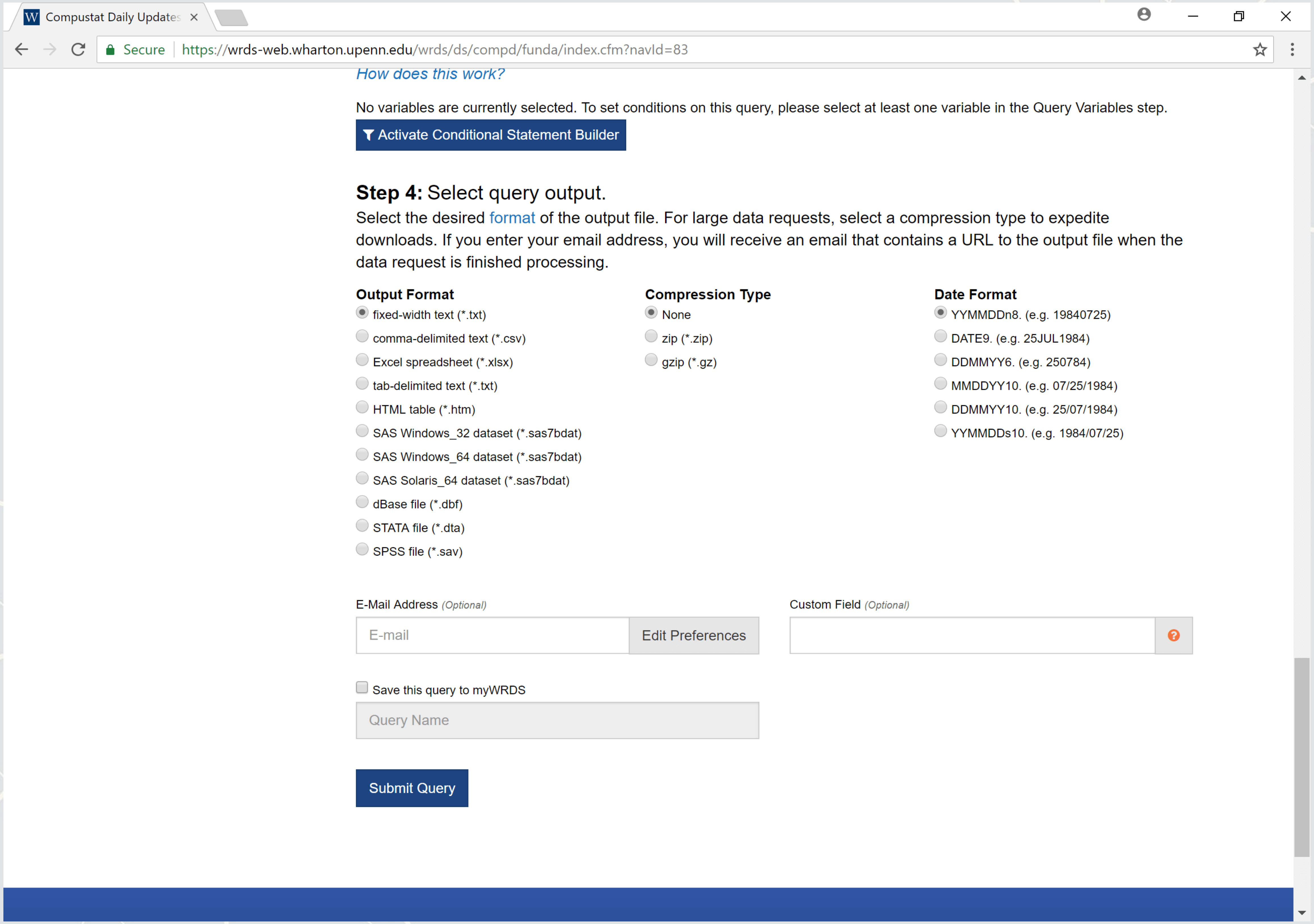
Search All

- ○ Company Name
- ○ Ticker Symbol
- ○ CUSIP
- ○ CIK Number
- ○ Stock Exchange Code
- ○ Fiscal Year-End
- ○ Foreign Incorporation Code
- ○ ADD1 -- Address Line 1
- ○ ADD2 -- Address Line 2
- ○ ADD3 -- Address Line 3
- ○ ADD4 -- Address Line 4

Conditional Statements (Optional)

How does this work?

Select output formats



Compustat Daily Update
https://wrds-web.wharton.upenn.edu/wrds/ds/compd/funda/index.cfm?navId=83
Secure

How does this work?

No variables are currently selected. To set conditions on this query, please select at least one variable in the Query Variables step.

Activate Conditional Statement Builder

Step 4: Select query output.

Select the desired format of the output file. For large data requests, select a compression type to expedite downloads. If you enter your email address, you will receive an email that contains a URL to the output file when the data request is finished processing.

Output Format

- ● fixed-width text (*.txt)
- ○ comma-delimited text (*.csv)
- ○ Excel spreadsheet (*.xlsx)
- ○ tab-delimited text (*.txt)
- ○ HTML table (*.htm)
- ○ SAS Windows_32 dataset (*.sas7bdat)
- ○ SAS Windows_64 dataset (*.sas7bdat)
- ○ SAS Solaris_64 dataset (*.sas7bdat)
- ○ dBase file (*.dbf)
- ○ STATA file (*.dta)
- ○ SPSS file (*.sav)

Compression Type

- ● None
- ○ zip (*.zip)
- ○ gzip (*.gz)

Date Format

- ● YYMMDDn8. (e.g. 19840725)
- ○ DATE9. (e.g. 25JUL1984)
- ○ DDMMYY6. (e.g. 250784)
- ○ MMDDYY10. (e.g. 07/25/1984)
- ○ DDMMYY10. (e.g. 25/07/1984)
- ○ YYMMDDs10. (e.g. 1984/07/25)

E-Mail Address *(Optional)*

E-mail | Edit Preferences

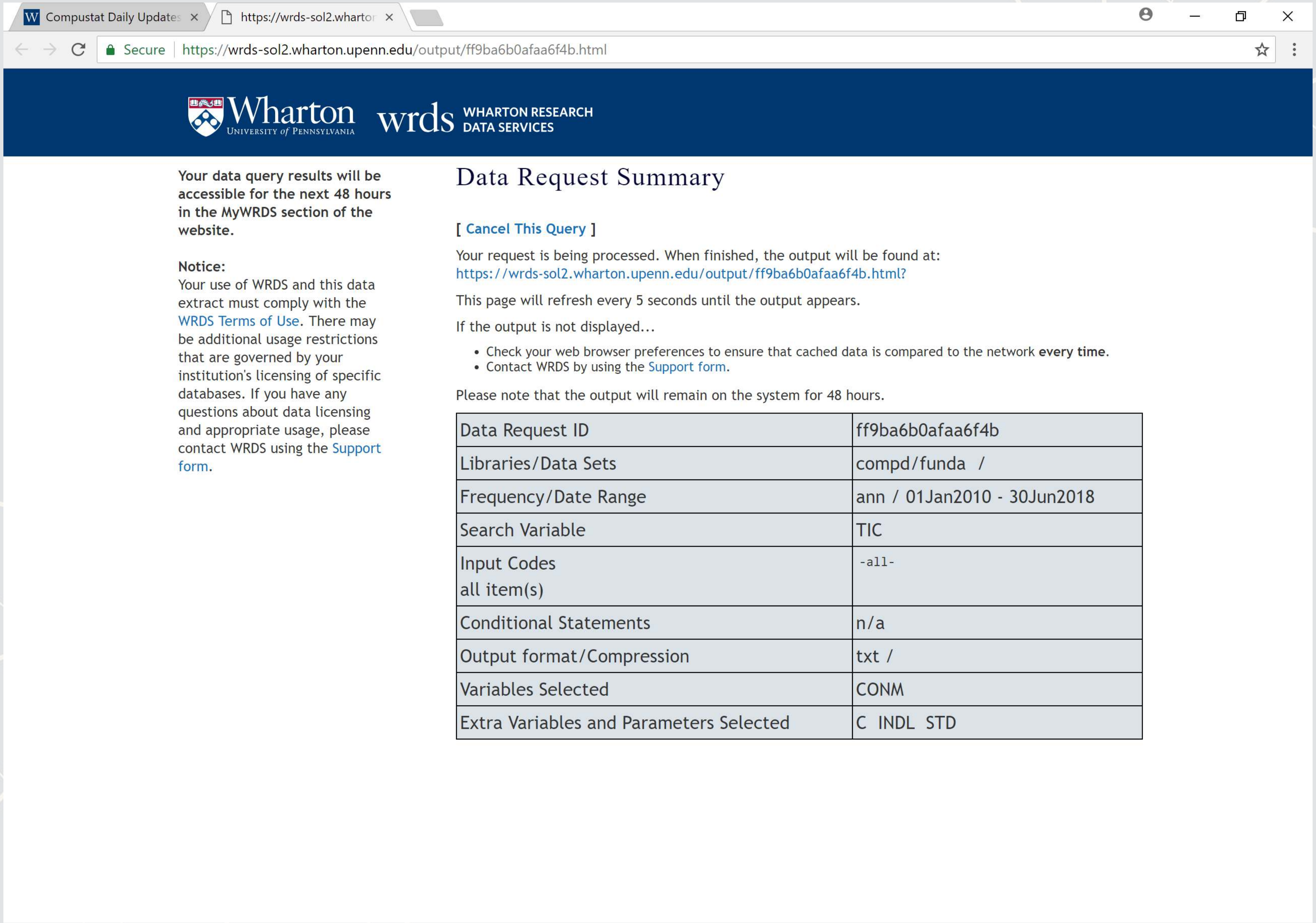
Custom Field *(Optional)*

□ Save this query to myWRDS

Query Name

Submit Query

Wait for the data to be prepared



Compustat Daily Updates
https://wrds-sol2.wharton
Secure
https://wrds-sol2.wharton.upenn.edu/output/ff9ba6b0afaa6f4b.html
Wharton
UNIVERSITY of PENNSYLVANIA
wrds
WHARTON RESEARCH
DATA SERVICES

Your data query results will be accessible for the next 48 hours in the MyWRDS section of the website.

Notice:
Your use of WRDS and this data extract must comply with the WRDS Terms of Use. There may be additional usage restrictions that are governed by your institution's licensing of specific databases. If you have any questions about data licensing and appropriate usage, please contact WRDS using the Support form.

Data Request Summary

[Cancel This Query]

Your request is being processed. When finished, the output will be found at:
https://wrds-sol2.wharton.upenn.edu/output/ff9ba6b0afaa6f4b.html?

This page will refresh every 5 seconds until the output appears.

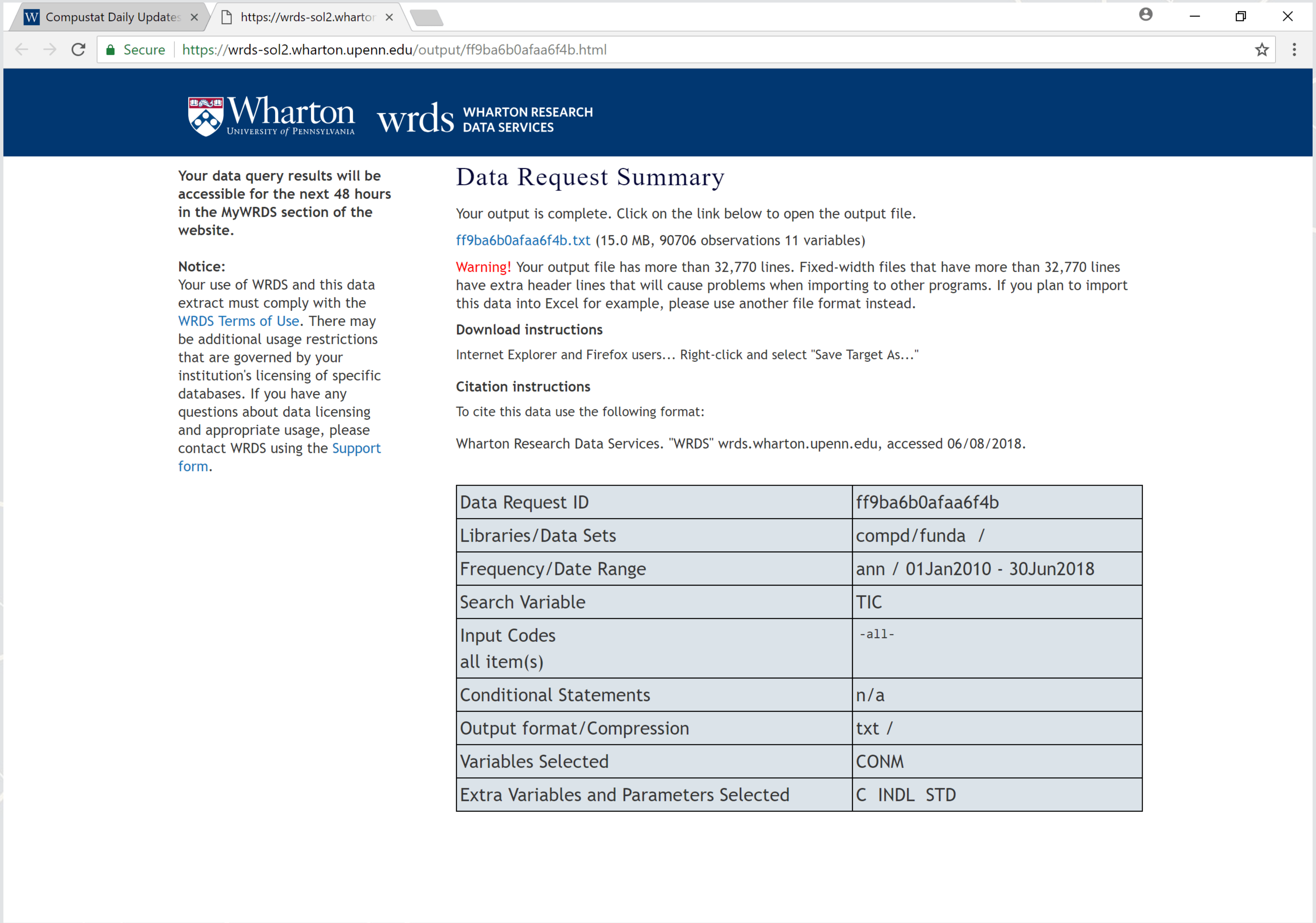
If the output is not displayed...

- Check your web browser preferences to ensure that cached data is compared to the network **every time**.
- Contact WRDS by using the Support form.

Please note that the output will remain on the system for 48 hours.

Data Request ID	ff9ba6b0afaa6f4b
Libraries/Data Sets	compd/funda /
Frequency/Date Range	ann / 01Jan2010 - 30Jun2018
Search Variable	TIC
Input Codes all item(s)	-all-
Conditional Statements	n/a
Output format/Compression	txt /
Variables Selected	CONM
Extra Variables and Parameters Selected	C INDL STD

Download the data!


Compustat Daily Update
https://wrds-sol2.wharton
Secure
https://wrds-sol2.wharton.upenn.edu/output/ff9ba6b0afaa6f4b.html
Wharton University of Pennsylvania
wrds WHARTON RESEARCH DATA SERVICES

Your data query results will be accessible for the next 48 hours in the MyWRDS section of the website.

Notice:
Your use of WRDS and this data extract must comply with the WRDS Terms of Use. There may be additional usage restrictions that are governed by your institution's licensing of specific databases. If you have any questions about data licensing and appropriate usage, please contact WRDS using the Support form.

Data Request Summary

Your output is complete. Click on the link below to open the output file.

ff9ba6b0afaa6f4b.txt (15.0 MB, 90706 observations 11 variables)

Warning! Your output file has more than 32,770 lines. Fixed-width files that have more than 32,770 lines have extra header lines that will cause problems when importing to other programs. If you plan to import this data into Excel for example, please use another file format instead.

Download instructions

Internet Explorer and Firefox users... Right-click and select "Save Target As..."

Citation instructions

To cite this data use the following format:

Wharton Research Data Services. "WRDS" wrds.wharton.upenn.edu, accessed 06/08/2018.

Data Request ID	ff9ba6b0afaa6f4b
Libraries/Data Sets	compd/funda /
Frequency/Date Range	ann / 01Jan2010 - 30Jun2018
Search Variable	TIC
Input Codes all item(s)	-all-
Conditional Statements	n/a
Output format/Compression	txt /
Variables Selected	CONM
Extra Variables and Parameters Selected	C INDL STD