Introduction to Data Science with R

EDLD 651

Joe Nese

University of Oregon

Fall 2021

Introduction to the Course

Week 1

Agenda

- Introductions
- Syllabus
- Introduce R

Learning Objectives

- Understand the course requirements
- Get you excited about R!

Some COVID-19 Notes

- Masks required at all times
- Try to sit in the same seat each week
- See syllabus for related policies
- Communicate early and often with me
- Allow everyone grace
- Questions?

Courageous Conversations with Me

- Creating norms of openness, understanding, and development
- An opportunity for us to grow
- Assume positive intent
- Critical for my work (and yours)

Credit: Dr. Rhonda Nese

About me

- BA: UC Santa Barbara
- PhD, School Psychology: University of Maryland
- Behavioral Research & Teaching (BRT) at UO since 2009
- Research Associate Professor

Research

- Applied statistical methods to measure and monitor student growth
- Inform the applied research methodologies used by researchers
- Developing and improving systems that support data-based decision making using advanced technologies to influence teachers' instructional practices and increase student achievement
    - CORE and CORE II

Teaching

- EDLD 651 - this one!
- EDLD 654 - Applied Machine Learning for Educational Data Scientists
- EDLD 609 - Data Science Capstone

About you

Please introduce yourself

- Name and program/year of study
- Why you want to learn R?
- Do you have any R experience?
- Tell me whatever you'd like me and the class to know (e.g., pronouns, circumstances)
- How many beans are in the jar?
    - No changing your answer! *Academic integrity!*

The Great Bean Experiment!

Why is this important?

- reproducibility
- transparency
- open data and code

Create a journal article!

with `{rmarkdown}` or `{papaja}`

Like we just did!

Create slides!

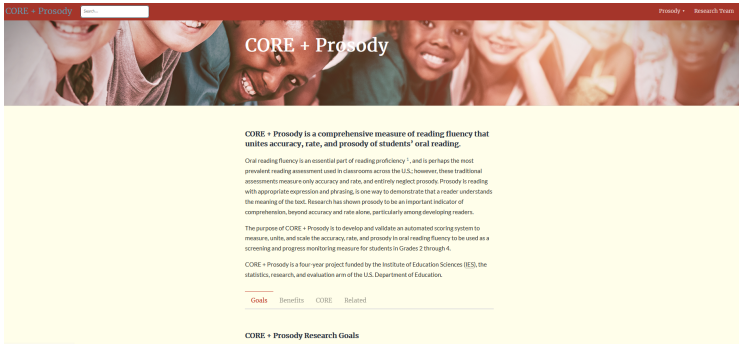
with {xaringan}

Like these!

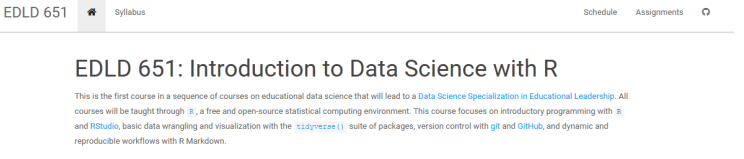
Create a website!

with {blogdown}, {distill}, {bookdown}, {rmarkdown}

For a project

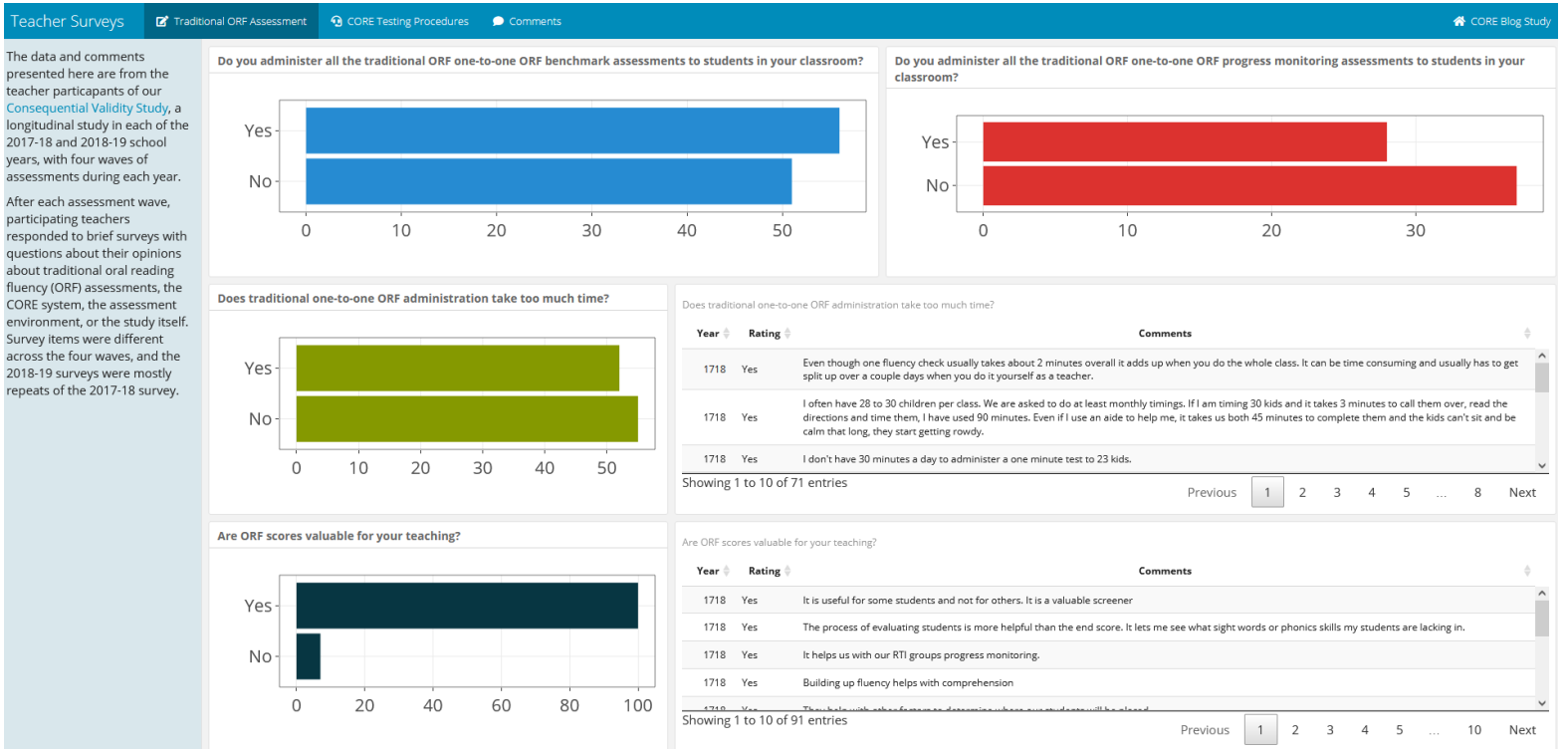


For this class



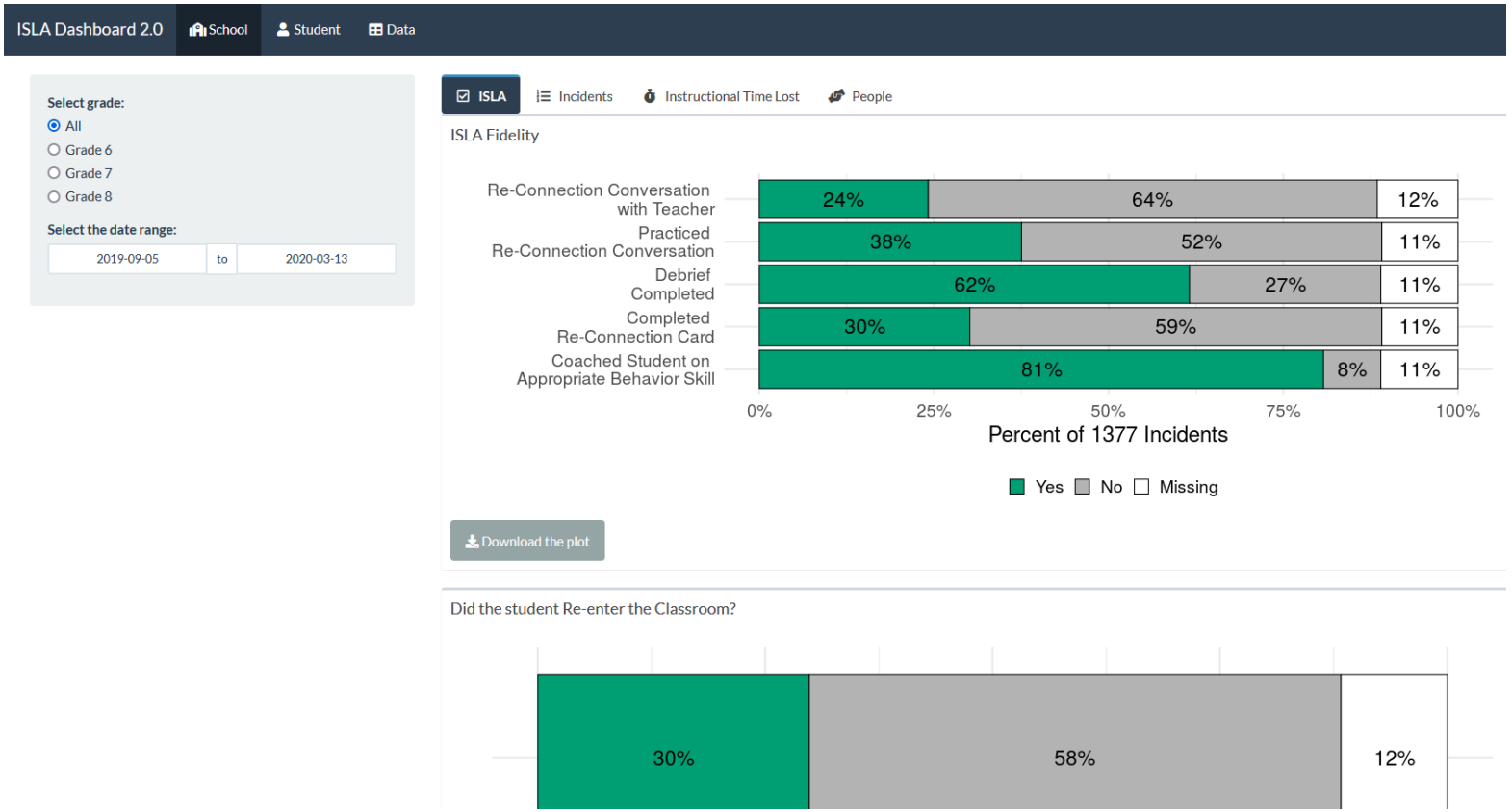
Create a dashboard!

with {flexdashboard}!



Create an app!

with {shiny}



Create a poster!

with {posterdown}!


I'M
#SCHITTSCREEK
Pop

Why is R awesome?

Data visualizations

- `{ggplot2}` -- your default by the end of this class, really powerful
- `{plotly}` -- interactive data visualizations
- `{shiny}` -- interactive data communications

Web

- `{blogdown}`, `{distill}`, `{bookdown}` -- build your own website
- `{rvest}` -- scrape web data

Modeling

- `{lme4}` -- multilevel modeling
- `{lavaan}` -- SEM
- `{tidymodels}` -- machine learning

Workflow!

- RStudio projects
- `{here}`

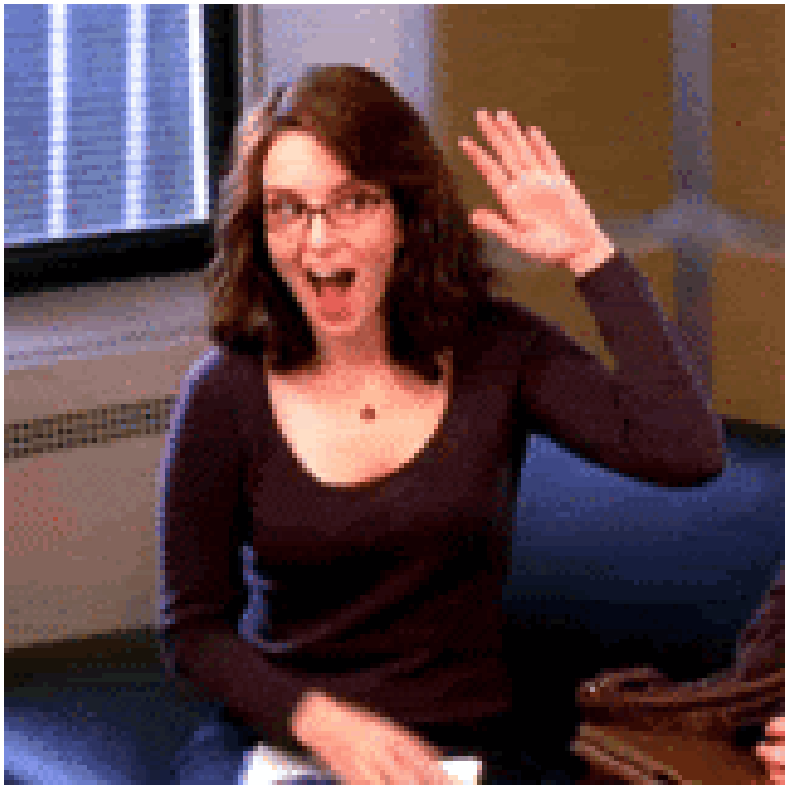
Acknowledgements

This course, and much of the materials prepared and content presented, was originally developed by Daniel Anderson

- Alison Hill, Chester Ismay, and Andrew Bray helped Daniel design the content for this course and the specialization as a whole

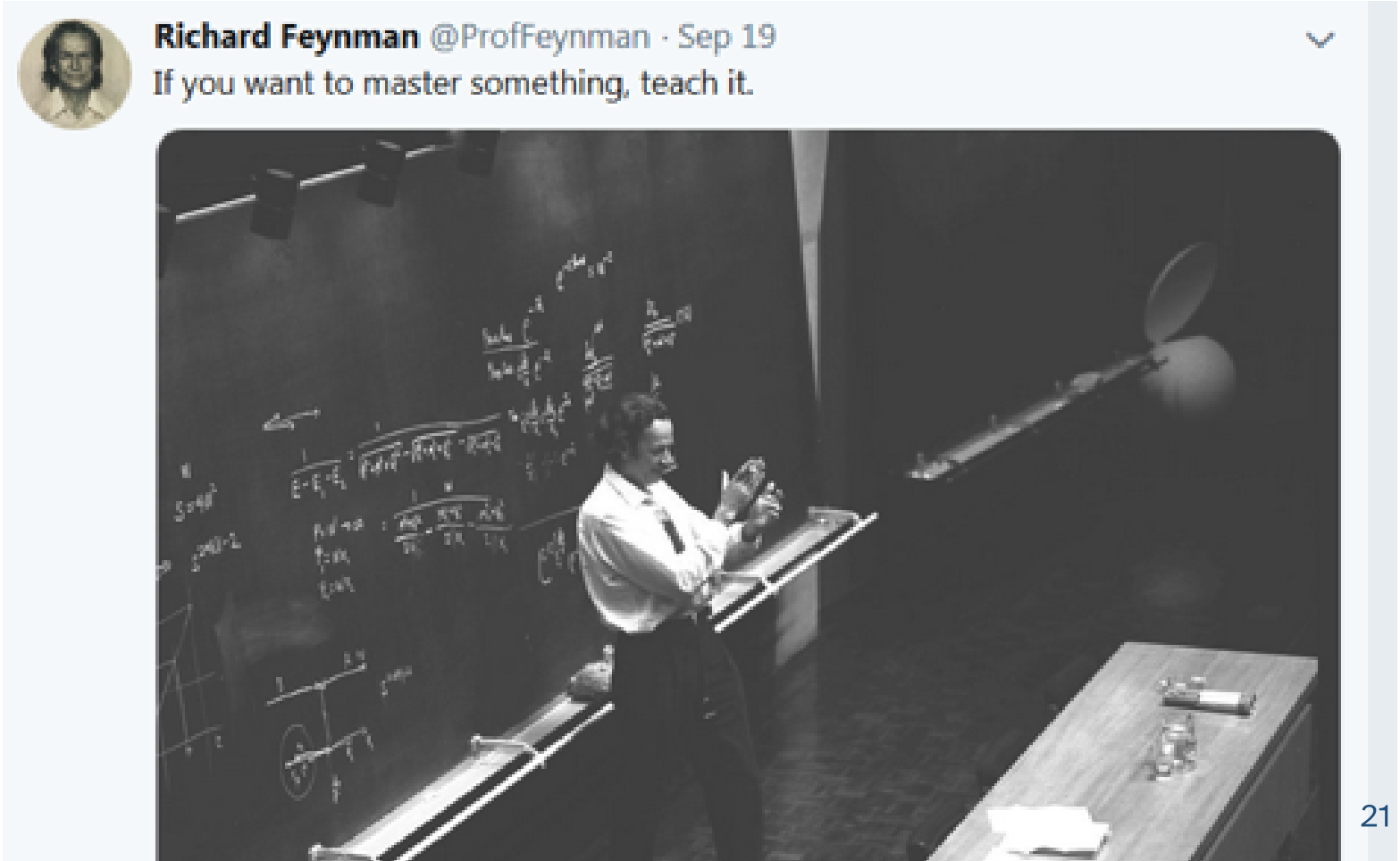
What this class is about

Celebrating successes!



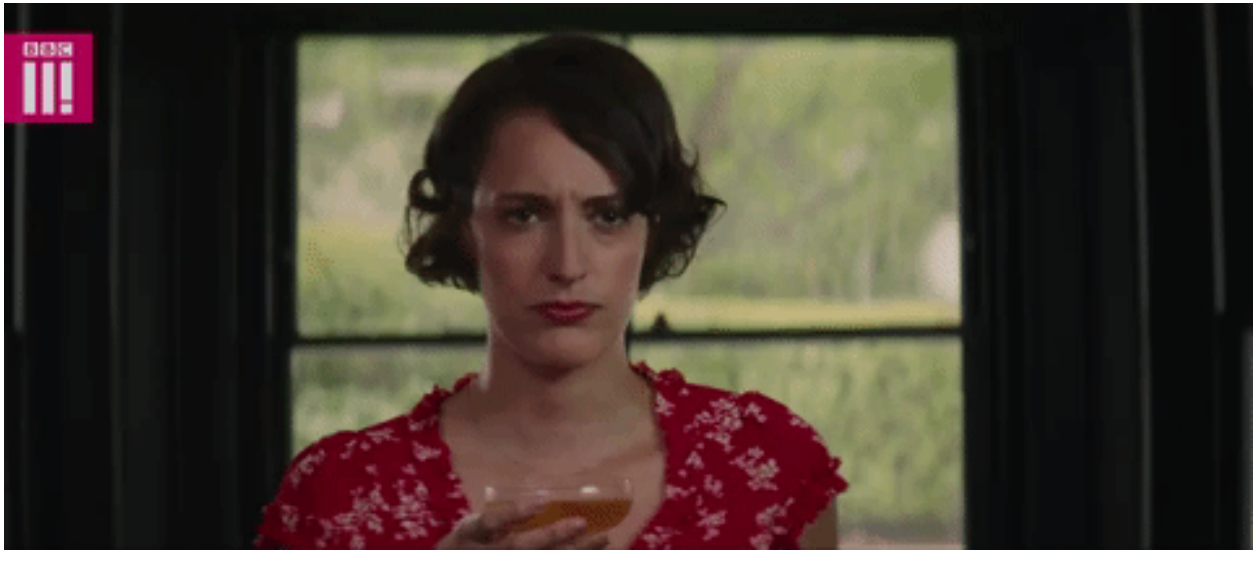
Dr. Richard Feynman

American theoretical physicist, Nobel Laureate



What this class is about

Celebrating failures!



What does Richard Feynman have to say about failing?

What this class is about

Celebrating trying!

Very smart person Richard Feynman said:

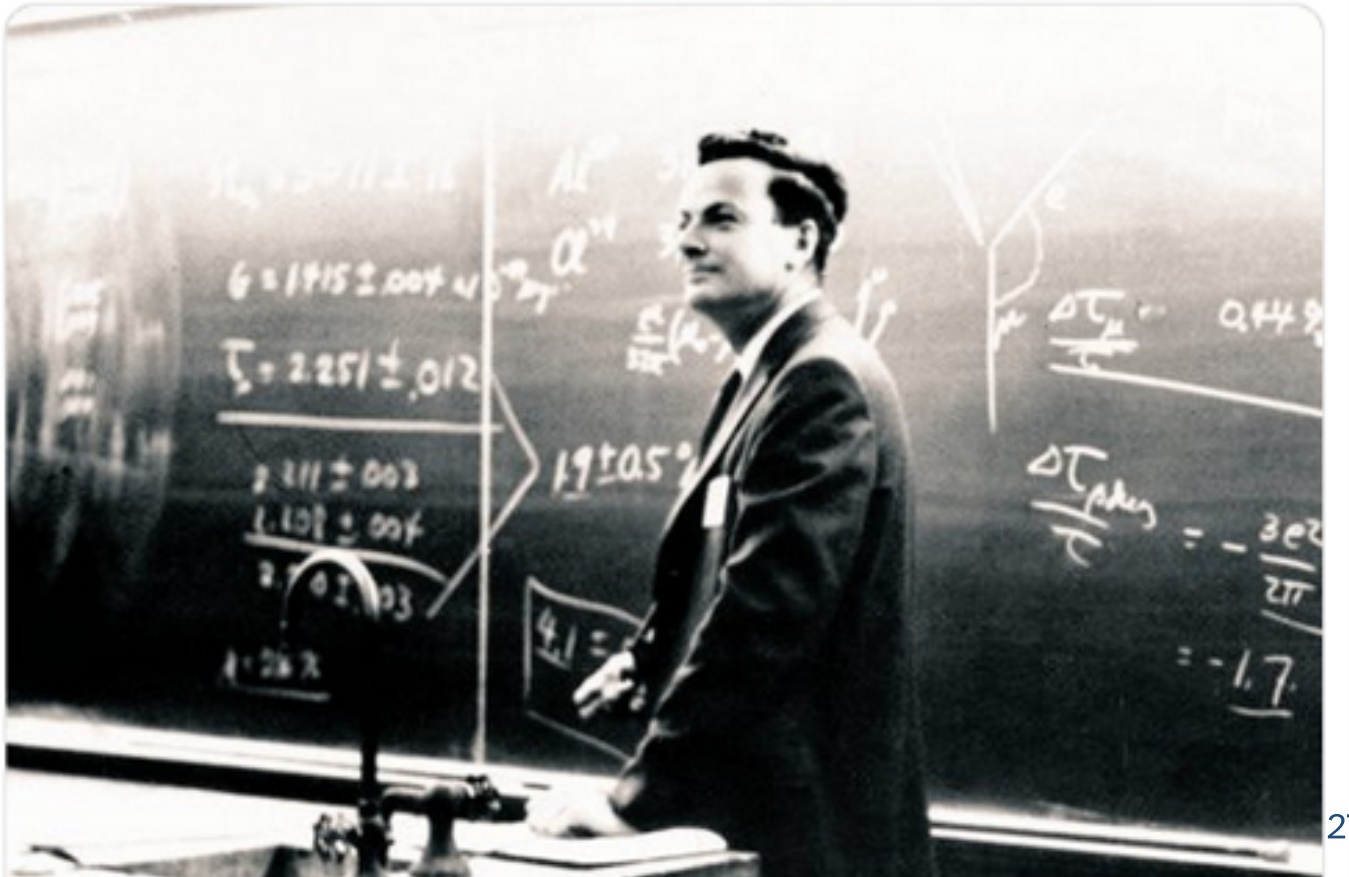
What this class is about

Learning to problem solve!

Richard?

Richard Feynman @ProfFeynman · Sep 5

It's OK to say, "I don't know." The pleasure is in finding the thing out.



This class...

...is

- data visualization
- data structuring and manipulations
- reproducible workflows
- a LOT (content and assignments at a fast pace)

...is not

- all encompassing
- first of the series
- a statistics course
    - but we'll use some stats in examples

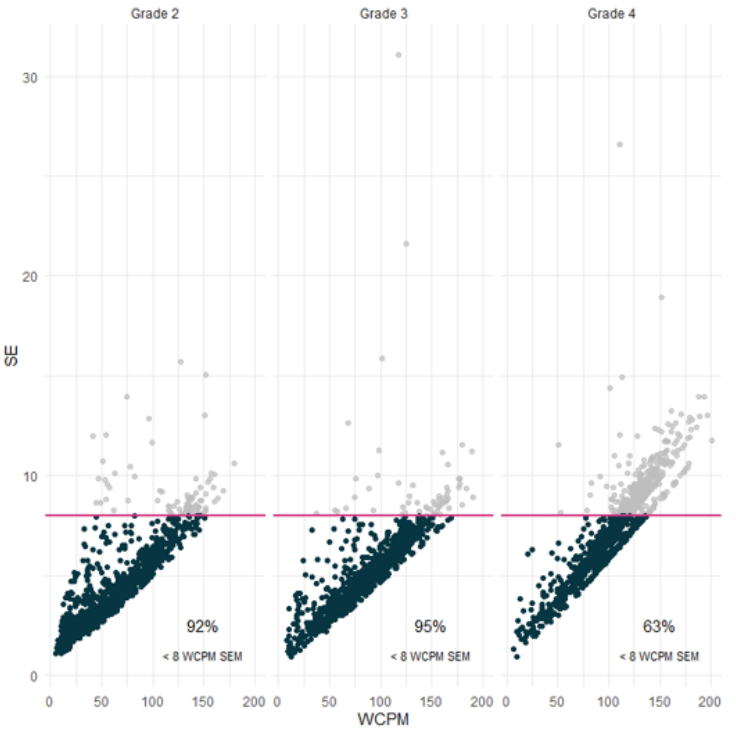
Shares

Sometimes I may ask people to share with the class something they have learned.

- A success, a new `{package}` or `function()`
- *Completely* voluntary
  - **BUT**, you might get a hex sticker

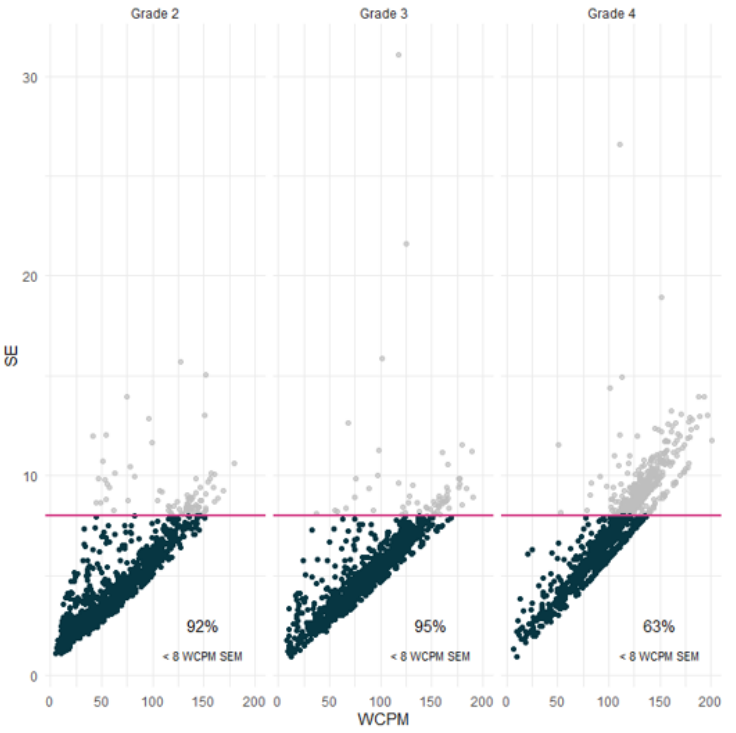
A sharing example

> "I made this cool figure!
>
> I used the `{gghighlight}` package for the first time!
>
> And I annoted my facets separately!"



Another sharing example

> "I went to re-run my ‘cool figure’ a couple months later and my code did not work!
>
> I spent [*mumble mumble*] minutes getting it to work again!"



Syllabus

Course Learning Outcomes

- Understand the R package ecosystem
  - how to find, install, load, and learn about them
- Read "flat" (i.e., rectangular) datasets into R
- Perform basic data manipulations / transformations in R with the `{tidyverse}`
  - leverage appropriate functions for introductory data science tasks
  - prepare data using scripts and reproducible workflows
- Use version control with R via git and GitHub
- Use R Markdown to create reproducible dynamic reports
- Understand and create different types of data visualizations

Course Site

https://uo-datasci-specialization.github.io/c1-intro-fall-2021/index.html

- schedule
- slides (before each class)
- assignments (after each class)
- syllabus
- data

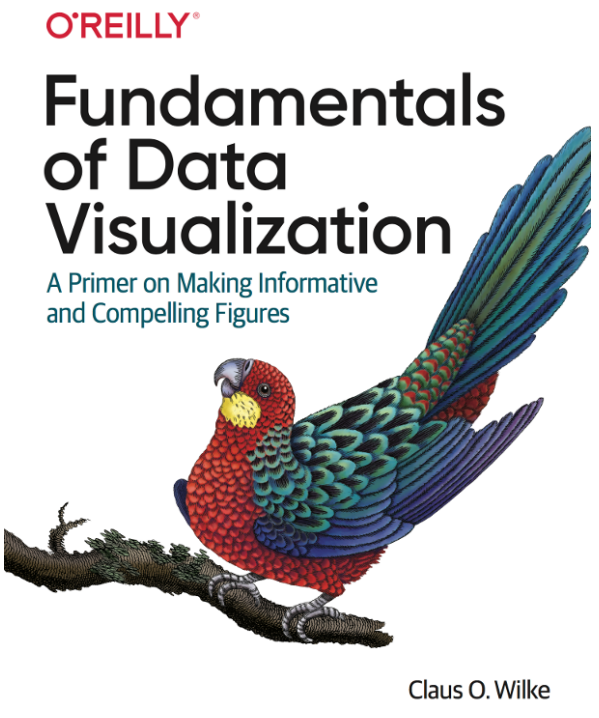
Canvas

- submit assignments
- announcements
- data?

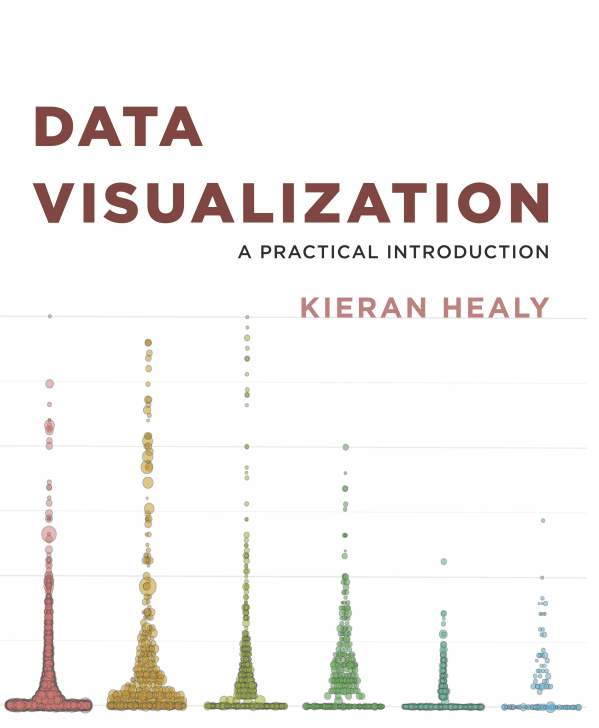
Required Textbooks (free)

Books not required (but possibly helpful)

https://clauswilke.com/dataviz/



https://socviz.co/



RStudio Primers, R-Bootcamp, & Codecademy

- Supplemental learning opportunities that I hope are helpful
- Part of your “Supplemental Learning” grade
- First time we are using Codecademy
- I will be interested in hearing your feedback
  - Not using Datacamp

Resources

- learnR4free
  - all sorts of resources (books, videos, websites, papers) to learn R for free
- R-Ladies
  - Global organization to promote gender diversity in the R community
- R-Ladies Portland
  - Local chapter
- R Users Groups
  - Eugene
  - Portland

Resources (con't)

- Twitter
  - Very active community. Browse the #rstats hashtag, and/or see syllabus for some recommendations for follows
- RStudio Community
  - Similar to stackoverflow but friendlier (great place to post questions)
  - Opinionated
  - RStudio-philosophy dominant (as is this class)
- R4DS
  - Online group of supportive people all trying to learn R, using the same book we will be using for this class
  - Tidy Tuesday: Data viz with open data each Tuesday, follow along on Twitter #tidytuesday

Weekly Schedule

- Readings - do before class
- Homework - due before next class
    - You **can** work in groups on these

Assignements

Homework (200 points)

- Homeworks – 10 at 10 points each (100 points)
- Supplemental Learning - 10 at 10 points each (100 points)
  - RStudio Primers $\times$ 4
  - R-Bootcamp $\times$ 2
  - Codeacademy $\times$ 4
  - Screenshot of specific part of "Supplemental Learning"

Final Project (200 points)

- Outline (15 points)
- Draft Data Prep Script (25 points)
- Peer Review of Draft Data Prep Script (25 points)
- Final Project Presentation (25 points)
- Final Paper (110 points)

400 points total

Grading

Grading Components

Lower %	Lower point range	Grade	Upper point range	Upper %
97	388	A+		
93	372	A	384	96
90	360	A-	368	92
87	348	B+	356	89
83	332	B	344	86
80	320	B-	328	82
77	308	C+	316	79
73	292	C	304	76
70	280	C-	288	72
		F	276	69

Homework

- Scored on a “best honest effort” basis
  - generally zero or full credit
- If you find yourself stuck and unable to proceed, **please contact the instructors for help rather than submitting incomplete work**
  - Contacting the instructor is part of the “best honest effort” and can result in full credit for an assignment even if the work is not fully complete
- **If the assignment is not complete, and the student has not contacted the instructor for help, it is likely to result is a partial credit score or a zero** * Labs submitted late will be docked by 30% (3 points)
  - Labs are generally due the class after they are assigned, before class starts

Final Project

- Group project, 3-4 people
- Rmarkdown document

Final project must:

- Be fully reproducible
  - This implies the data are open
- Be a collaborative project hosted on GitHub
- Move data from its raw "messy" format to a tidy data format
- Include at least two exploratory plots
- Include at least summary statistics of the data in tables, although fitted models are also encouraged

Final Project - Dates

- **Week 3 (10/20)**: Self-selected groups finalized
- **Week 5 (11/3)**: Final Project Outline due
- **Week 8 (11/24)**: Data prep script due
- **Week 9 (12/1)**: Peer review due
- **Week 10 (12/8)**: Final project presentations
- **Week 11 (12/15)**: Final Paper due

Final Project - Paper Scoring Rubric

Criteria	Points Possible
Writing	
Abstract	5
Introduction	5
Methods	5
Results	5
Discussion	5
References	5
Code	
Document is fully reproducible	25
Demonstrate use of inline code	5
At least two data visualizations	10 (5 pts each)
Demonstrate tidying messy data using:	
`pivot_longer()`	5
`mutate()`	5
`select()` and `filter()`	5
`pivot_wider()`	5

Final Project - Outline

Primary purpose: Get feedback and give me a preview of your project

- Description of the data to be used
- Discussion of preparatory work that needs to be done
- How the requirements of the final project will be met
- Anything you want feedback on

Final Project - Data Prep Script

- Expected to be a work in progress
- Provided to your peers so they can learn from you as much as you can learn from their feedback

Peer Review

- Understand the purpose of the exercise
- Conducted as a professional product
- Should be **very** encouraging
- Zero tolerance policy for inappropriate comments

Final Project - Presentation

Order randomly assigned. Should cover the following:

- Share your journey (everyone, at least for a minute or two)
- Discuss challenges you had along the way
- Celebrate your successes
- Discuss challenges you are still facing
- Discuss substantive findings
- Show off your cool figures!
- Discuss next R hurdle you want to address

Final Project – Presentation Scoring Rubric

Final Presentation Rubric

Criteria	Points possible
Challenges faced along the way	5
Victories and things to celebrate	5
Challenges you are still facing	5
Substantive findings/interpretations	5
Next R hurdle to tackle	5
Total	**25**

Final Projct - Paper

- Research Paper
  - Abstract, Intro, Methods, Results, Discussion, References
  - Should be brief: 3,500 words max
- No code displayed - should look like any other manuscript being submitted for publication
- Include at least 1 table
- Include at least 2 plots
- Should be fully open, reproducible, and housed on GitHub
  - I should be able to clone your repository, open the R Studio Project, and reproduce the full manuscript (by knitting the R Markdown doc)

git and GitHub

- Will be required for final project
- What is it?
    - Version control system
    - Collaboration tool
    - Can be powerful for transparency and reproducibility
    - More to come
- Think of a GitHub profile as a public resume and treat it as such
- Username advice from Jenny Bryan

Git might be frustrating, but we'll get through it together!

Welcome to !

What is R

- R is an environment and programming language, created primarily for statistical analyses and graphics
- No point-and-click interface
- Open source
  - source code is freely available, and can be redistributed and modified
- Incredibly powerful and flexible
  - Vast array of external packages available for specilized functions (analyses, data visualizations, automate the data “cleaning" process)

Sara Kahanamoku
@sara_kahanamoku

Follow

#rstats thought of the day: you must become comfortable with failing (code breaking, not knowing how to do something) in order to code well. Good code is born from failure. 🤘🏽

7:53 AM - 25 May 2018 from Boston, MA

Hannah A. Brazeau
@HABrazeau

Follow

Impost-R syndrome: when you're convinced everyone but you learns R effortlessly.

12:28 PM - 17 Feb 2017

Marc Joanisse
@drmarcj

Follow

Learning R is easy, I've done it at least twice a week for the last 5 years.

> The bad news is that whenever you learn a new skill you're going to suck. It's going to be frustrating. The good news is that is typical and happens to everyone and it is only temporary. You can't go from knowing nothing to becoming an expert without going through a period of great frustration and great suckiness.
>
> -- Hadley Wickham

Moving to code/programming

Advantages

- Flexibility
  - Essentially anything is possible
- Transparency
  - Documented history of your analysis
- Efficiency
  - Many tasks can be automated

Disadvantages

- Steep learning curve
  - Definitely requires a significant time investment
  - Similar to learning a new language
- You will lose patience with point-and-click interfaces
- Likely to become "one of the converted"

Code-based Interface

R

This is the R console

Code-based Interface

RStudio

This is the RStudio IDE

(Integrated Development Environment)

How to learn R?

- Time
- Use it!
- Dedication and determination help
- Be patient and forgiving with yourself, it will feel slow at first
- I still get frustrated

R as a big calculator

```
3 + 2
```

```
## [1] 5
```

```
3 / (3+2)
```

```
## [1] 0.6
```

Object Assignment <-

Objects are stored in active memory of your computer with names that you provide

<- is the assignment operator

It is used to assign names to objects (like an = operator)

```
a <- 3
b <- 2
a
```

```
## [1] 3
```

```
a + b
```

```
## [1] 5
```

```
a / (a + b)
```

```
## [1] 0.6
```

Re-assignment

```
a <- 10
a
```

```
## [1] 10
```

```
a <- "EDLD 651"
a
```

```
## [1] "EDLD 651"
```

Data Types

Data can be a variety of types

- character: `"Hello world!"` or `"EDLD 651"`
- numeric: `2` and/or `15.5`
    - decimal also called `double`
- integer: `2L` (the `L` tells `R` to store this as an integer)
- logical: `TRUE` or `FALSE`

These objects can be extremely useful in programming

Objects

Objects can also be:

- variables
- data sets
- models
- results
- functions
- figures
- more...

You can then use or manipulate these in different ways

- plots
- functions
- operators (arithmetic, logical, comparison)

R Functions

Anything that carries out an operation in R is a function, even +

Functions are generally followed by ()

- function()
  - sum()
  - lm()
  - sqrt()

package::function()

the package is also called the "namespace"

Getting help

`?` before a function name can be very helpful (and also confusing early on)

- Helpful for understanding formal *arguments* of a function
    - *arguments* are options within a `function()`

`?function_name()`

```
?mutate() #(mutate() is a function in the {dplyr} package)

?dplyr::mutate()
```

function, and the library it is in.

mean {base} R Documentation

Arithmetic Mean

Description

What it does.

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

The function's name, and in the parentheses the named arguments it expects, in the order it expects them. If an argument has a default value, it is shown. Arguments without default values (e.g. x) must be provided by you.

Arguments

More details on each named argument. This will tell you what class of thing each argument has to be—an object, a number, a data frame, a logical value, etc.

`x`	An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.
`trim`	the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
`na.rm`	a logical value indicating whether `NA` values should be stripped before the computation proceeds.
`...`	further arguments passed to or from other methods.

The ellipsis allows other arguments to be passed to and from the function.

Value

What the function returns—i.e., the result of whatever operation or calculation it performs. This can be a single number, as here, or a multi-part object such as a list, a data frame, a plot, or a model.

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.
If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`weighted.mean`, `mean.POSIXct`, `colMeans` for row and column means.

Other related functions

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

[Package *base* version 3.4.3 Index]

Visit the package's Index page to look for Demos and Vignettes detailing how it works.

https://socviz.co/

Getting help (con't)

- Resources on syllabus
- Your classmates!
- Me
  - Appointment by email
  - If you email me with a question, provide your R files & data
- Google

Back to Dr. Feynman

(Isaac Asimov, really)

Installs

Install Check

Installs

- R
- Rstudio
- Git
- GitKraken

Also

- Register for GitHub account
  - username advice from Jenny Bryan

Customize!

Customize your RStudio Display!

Tools ➡ Global Options...

General

- Workspace
  - "*Restore .RData into workspace at startup*" -- **Uncheck**
  - "*Save workspace to .RData on exit:*" -- **Never**
- History
  - "*“Always save history (even when not saving .RData)*" -- **Check**
- Appearance
  - Font, font size
  - Editor theme!!
- Pane Layout
- Code
  - Highlight R function calls (up to you)
  - Show indent guides (up to you)

Keyboard Shortcuts

There are a lot...

Tools ➡ Keyboard Shortcuts Help (`Alt + Shift + K`)

Tools ➡ Modify Keyboard Shortcuts

My favorites (for Macs, sub `Command` for `Ctrl`)

- `Ctrl + Enter` = run code
- `Ctrl + Shift + M` = insert pipe (`%>%`)
- `Ctrl + Alt + I` = insert code chunk in R Markdown
- `Ctrl + Shift + C` = comment a block

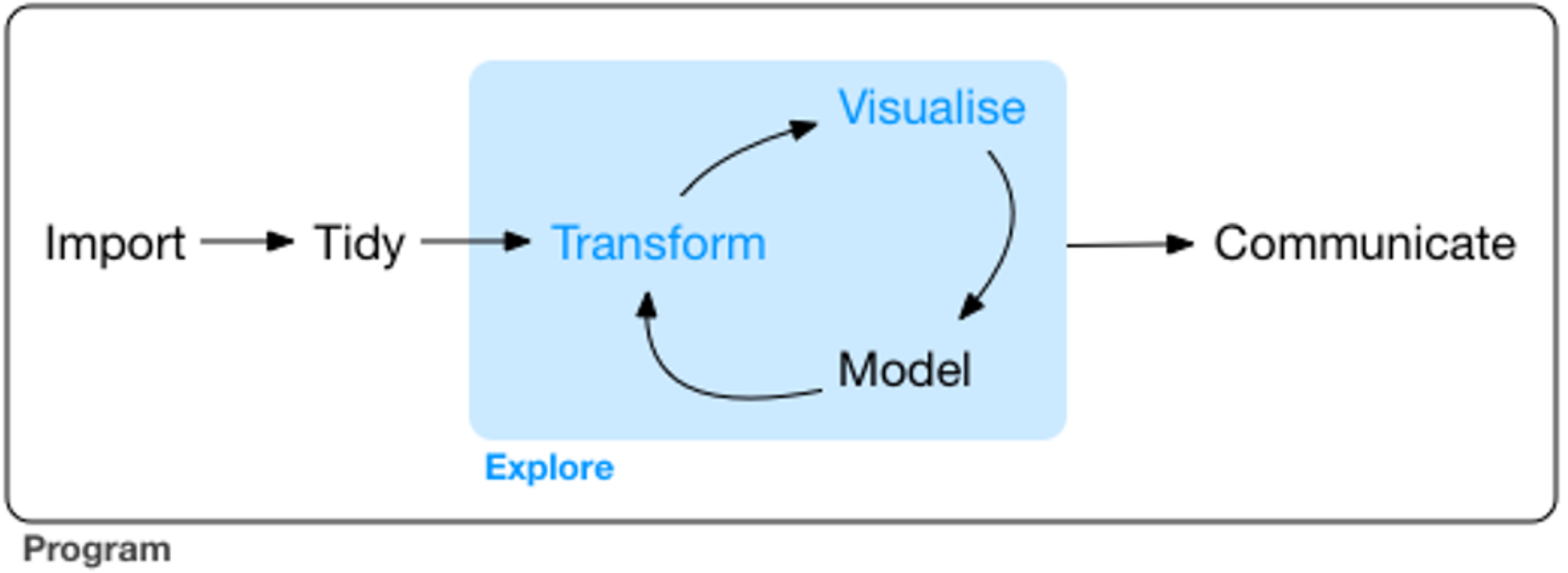
Other good ones

- `Alt + -` = insert assignment operator (<-)
- `Ctrl + Shift + Alt + Up/Down` = add cursor above/below current cursor

[demo]

Packages

The data science pipeline



How do we go about this?

"Out of the box" functionality

R Packages provide functions (and datasets) for you to use

Some packages are pre-loaded

- {base}
- {graphics}
- {stats}

Some packages are pre-installed

- {boot}
- {MASS}
- {Matrix}

Pre-loaded vs Installed

Pre-loaded packages work on launch

For example, `plot()` is part of the `{graphics}` package, which ships with `R`

```
plot(x = 1:10, y = 1:10)
```

{base} package

- All functions come from a package
- What do you get with the following?

```
?'+'
```

Arithmetic {base} R Documentation

Arithmetic Operators

Description

These unary and binary operators perform arithmetic on numeric or complex vectors (or objects which can be coerced to them).

Usage

```
+ x
- x
x + y
x - y
x * y
x / y
x ^ y
x %% y
x %/% y
```

Arguments

x, y	numeric or complex vectors or objects which can be coerced to such, or other objects for which methods have been written.

Details

Packages on CRAN

There is a TON of functionality that comes with R right from your initial download **BUT** the functionality can be extended by installing other packages

CRAN is the official repository

- a network of servers maintained by the R community around the world
- coordinated by the R Foundation
- a package needs to pass several tests to be published on CRAN

Most often, you will first install a package, then load it

- `install.packages("package_name")`
- `library(package_name)`

You will only need to install a package **once** (generally). After that, it is in your library and only needs to be loaded

Currently, the CRAN package repository features 18,000+ packages

- 16,300+ last year at this time
- constantly evolving to keep up with the varied demands of the data science community

One more time

`install.packages("package_name")`

- you will only need to do this the **FIRST** time you use the package
- don't keep this code line (you can run it in the console, or delete it from your script)
- notices the quotes

`library(package_name)`

- you will do this each time you use the package in your scripts
- notices no quotes

Other packages

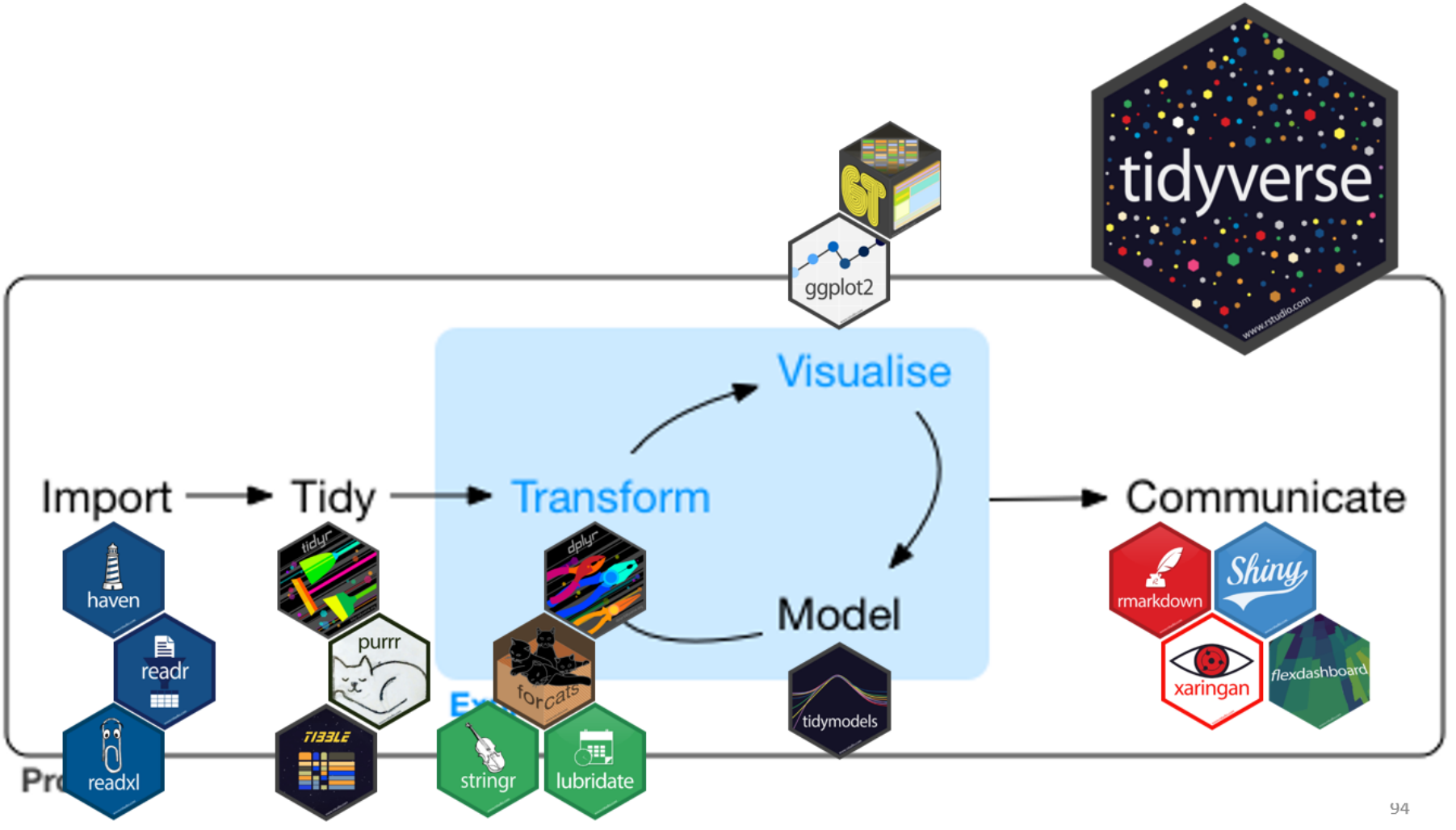
On GitHub

- Popular repository for open source projects
  - Integrated with git (version control)
  - Easy to share/collaborate
- Not necessarily R specific
- Generally, these packages “in development,” or are the beta versions of existing packages
- No review process associated with GitHub

Possibilities

With just a basic knowledge of R you have access to thousands of packages

- Expanding on a daily basis
- Provides access to cutting edge and specialized functionality for analysis, data visualization, and data preparation
- Some of the most modern thinking on data analysis topics are often represented in these packages


tidyverse
ggplot2
Visualise
Import
Tidy
Transform
Model
Communicate
haven
readr
readxl
tidyr
purrr
tibble
dplyr
forcats
stringr
lubridate
tidymodels
rmarkdown
Shiny
xaringan
flexdashboard

Let's dive in

(or end here for today)

How do we access variables?

- Generally, in this course, with `{tidyverse}` tools
- Sometimes with `$` or with `[]`
  - `data_name$variable_name`
  - `data_name["variable_name"]`

Some data

Let's load the {tidyverse}, and then we'll access some variables from the data sets within the packages

- Look at the gss_cat data frame

```
# install.packages("tidyverse")
## if you have never installed {tidyverse} you'll need to do that first
library(tidyverse)

gss_cat
```

```
## # A tibble: 21,483 x 9
##     year marital         age race  rincome        partyid   relig  denom  t
##    <int> <fct>         <int> <fct> <fct>          <fct>     <fct>  <fct>
##  1  2000 Never married    26 White $8000 to 9999  Ind,near~ Prote~ South~
##  2  2000 Divorced         48 White $8000 to 9999  Not str ~ Prote~ Bapti~
##  3  2000 Widowed          67 White Not applicable Independ~ Prote~ No de~
##  4  2000 Never married    39 White Not applicable Ind,near~ Ortho~ Not a~
##  5  2000 Divorced         25 White Not applicable Not str ~ None   Not a~
##  6  2000 Married          25 White $20000 - 24999 Strong d~ Prote~ South~
##  7  2000 Never married    36 White $25000 or more Not str ~ Chris~ Not a~
##  8  2000 Divorced         44 White $7000 to 7999  Ind,near~ Prote~ Luthe~
```

Selecting varaibles

Select the `marital` variable with `$`

```
gss_cat$marital
```

```
##    [1] Never married Divorced      Widowed       Never married Divorced
##    [6] Married       Never married Divorced      Married       Married
##   [11] Married       Married       Married       Married       Divorced
##   [16] Married       Widowed       Never married Married       Married
##   [21] Married       Married       Never married Widowed       Widowed
##   [26] Widowed       Widowed       Widowed       Divorced      Widowed
##   [31] Widowed       Married       Married       Never married Married
##   [36] Never married Never married Never married Never married Never marrie
##   [41] Married       Married       Divorced      Never married Never marrie
##   [46] Never married Married       Married       Married       Married
##   [51] Never married Married       Married       Married       Married
##   [56] Divorced      Divorced      Divorced      Never married Never marrie
##   [61] Married       Married       Never married Divorced      Never marrie
##   [66] Widowed       Divorced      Married       Never married Never marrie
##   [71] Widowed       Widowed       Widowed       Widowed       Widowed
##   [76] Never married Widowed       Never married Married       Never marrie
##   [81] Married       Married       Widowed       Married       Married
##   [86] Divorced      Never married Separated     Never married Widowed
```

Look at the `str`ucture of an object

```
str(gss_cat$marital)
```

```
##  Factor w/ 6 levels "No answer","Never married",..: 2 4 5 2 4 6 2 4 6 6 ...
```

```
str(gss_cat)
```

```
## tibble [21,483 x 9] (S3: tbl_df/tbl/data.frame)
##  $ year   : int [1:21483] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000
##  $ marital: Factor w/ 6 levels "No answer","Never married",..: 2 4 5 2 4 6
##  $ age    : int [1:21483] 26 48 67 39 25 25 36 44 44 47 ...
##  $ race   : Factor w/ 4 levels "Other","Black",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ rincome: Factor w/ 16 levels "No answer","Don't know",..: 8 8 16 16 16 5
##  $ partyid: Factor w/ 10 levels "No answer","Don't know",..: 6 5 7 6 9 10 5
##  $ relig  : Factor w/ 16 levels "No answer","Don't know",..: 15 15 15 6 12
##  $ denom  : Factor w/ 30 levels "No answer","Don't know",..: 25 23 3 30 30
##  $ tvhours: int [1:21483] 12 NA 2 4 1 NA 3 NA 0 3 ...
```

Because gss_cat is a tibble

```
gss_cat
```

```
## # A tibble: 21,483 x 9
##     year marital         age race  rincome        partyid   relig  denom  t
##    <int> <fct>         <int> <fct> <fct>          <fct>     <fct>  <fct>
##  1  2000 Never married    26 White $8000 to 9999  Ind,near~ Prote~ South~
##  2  2000 Divorced         48 White $8000 to 9999  Not str ~ Prote~ Bapti~
##  3  2000 Widowed          67 White Not applicable Independ~ Prote~ No de~
##  4  2000 Never married    39 White Not applicable Ind,near~ Ortho~ Not a~
##  5  2000 Divorced         25 White Not applicable Not str ~ None   Not a~
##  6  2000 Married          25 White $20000 - 24999 Strong d~ Prote~ South~
##  7  2000 Never married    36 White $25000 or more Not str ~ Chris~ Not a~
##  8  2000 Divorced         44 White $7000 to 7999  Ind,near~ Prote~ Luthe~
##  9  2000 Married          44 White $25000 or more Not str ~ Prote~ Other
## 10  2000 Married          47 White $25000 or more Strong r~ Prote~ South~
## # ... with 21,473 more rows
```

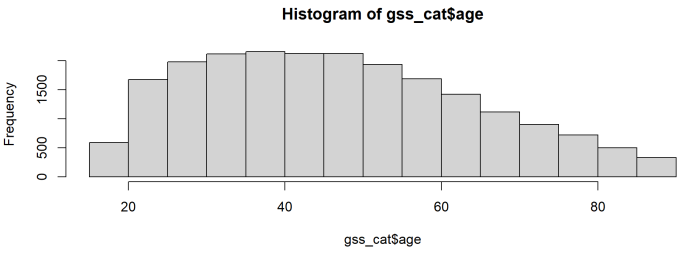
When to use $?

- Often when you need to use {base} R
- Execute a function casually

```
table(gss_cat$marital)
```

```
##
##      No answer Never married      Sepa
##             17          5416
##        Married
##          10117
```

```
hist(gss_cat$age)
```



Your turn

Run `table()` on the *religion* variable

Produce a `hist`ogram of the *tvhours* variable

The pipe operator (%>%)

- The `%>%` operator (`Ctrl + Shift + M`) inserts the input from the left as the first argument in the next function
- To start, you can read it as, "then"
- It is crucial for work in the `{tidyverse}`

```
gss_cat %>%
  count(marital)
```

```
## # A tibble: 6 x 2
##   marital           n
##   <fct>         <int>
## 1 No answer        17
## 2 Never married  5416
## 3 Separated       743
## 4 Divorced       3383
## 5 Widowed        1807
## 6 Married       10117
```

We will talk about this a lot more later

Why use %>%

Chaining arguments is **efficient** and **easy to read**

```
gss_cat %>%
  filter(relig == "Buddhism",
         age > 55) %>%
  select(age, partyid, rincome) %>%
  arrange(age) %>%
  slice(1:5)
```

```
## # A tibble: 5 x 3
##     age partyid          rincome
##   <int> <fct>            <fct>
## 1    56 Not str democrat $25000 or more
## 2    56 Ind,near dem     $25000 or more
## 3    56 Not str democrat $4000 to 4999
## 4    56 Independent      $25000 or more
## 5    57 Independent      Not applicable
```

Equivalent to:

```
slice(arrange(select(filter(gss_cat, relig == "Buddhism", age > 55), age
```

Next time

Before next class

- Reading
  - Week 1 reading (if you haven't already)
  - Project-oriented Workflow
  - R4DS Ch 8
  - R4DS 27
  - here::here() Jenny Bryan
  - `{rio}` vignette
- Assignments
  - A Quick Tour of the RStudio IDE
  - RStudio Primer: Programming basics
  - Codecademy: Introduction to R Syntax