Building Reproducible and Replicable Projects

R Workflows

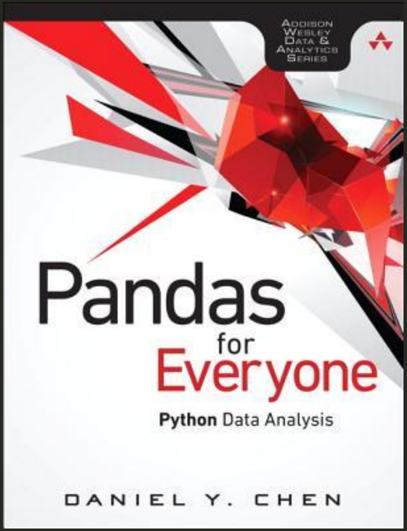
Daniel Chen @chendaniely

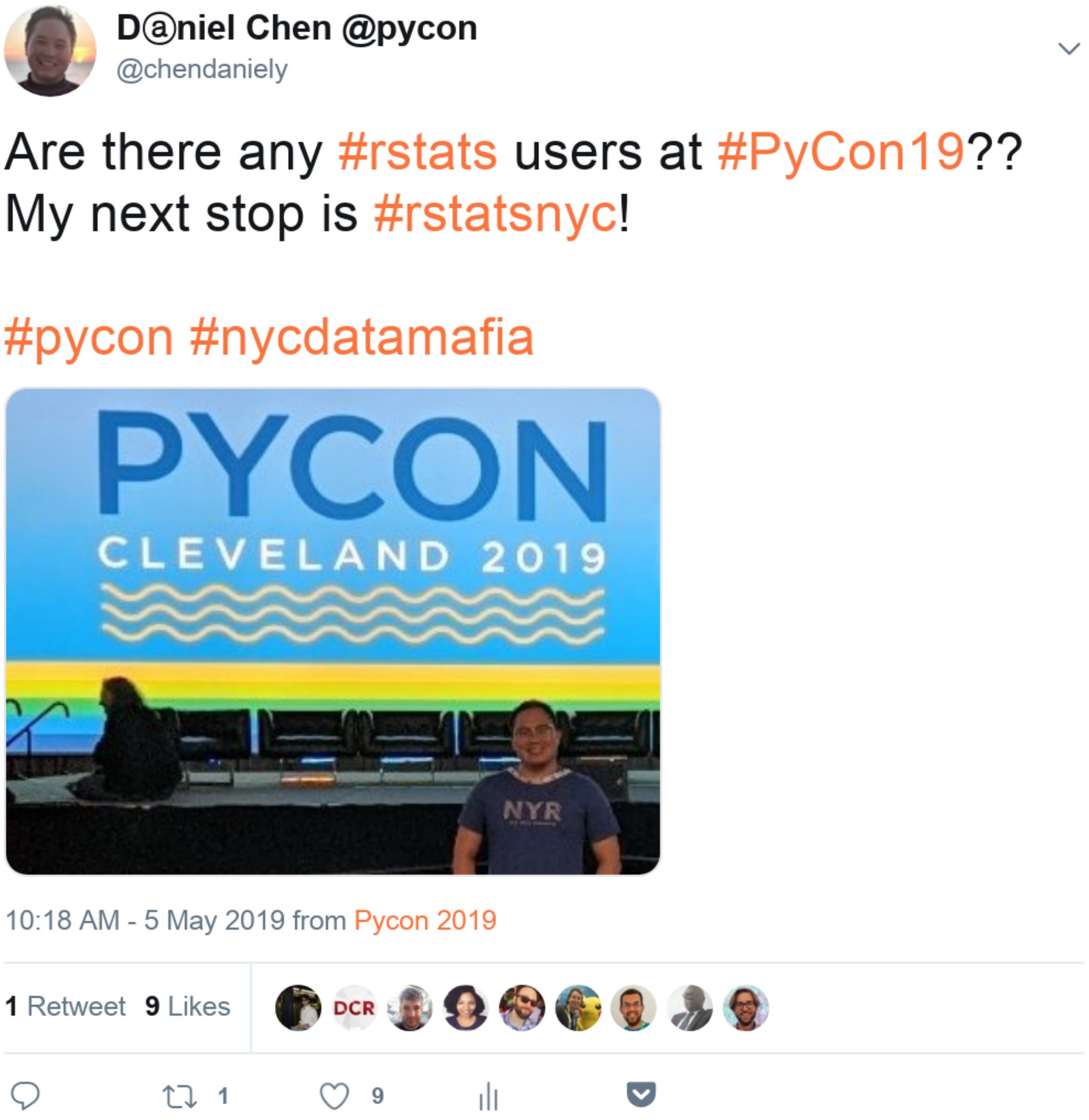
NYR Conference 2019
https://github.com/chendaniely/rstatsnyc_2019-workflow

hi!

I'm Daniel

- PhD Student: Virginia Tech
- Instructor
- Data Scientist
- Community Member
- RStudio Intern!
- Author:





D@niel Chen @pycon
@chendaniely

Are there any #rstats users at #PyCon19??
My next stop is #rstatsnyc!

#pycon #nycdatamafia

10:18 AM - 5 May 2019 from Pycon 2019

1 Retweet 9 Likes

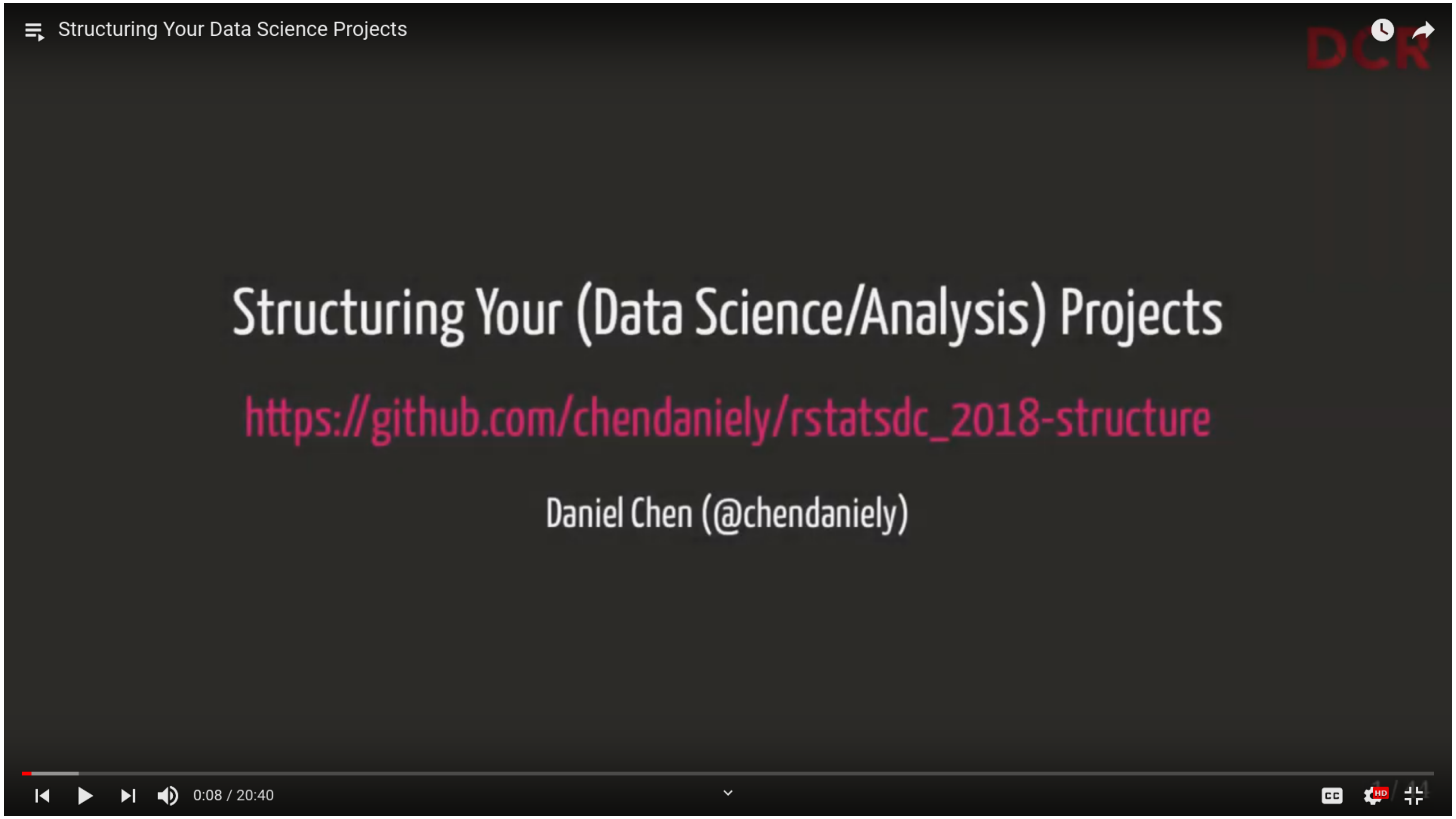
1 9

Caves!

Ginnie Springs, FL

- https://www.submergedscuba.com/
- http://www.scuba.org.vt.edu/

Continuing the DCR 2018 Story



Structuring Your (Data Science/Analysis) Projects

https://github.com/chendaniely/rstatsdc_2018-structure

Daniel Chen (@chendaniely)

https://youtu.be/UQHz38s3DyA

Previously...

Structuring Your Data Science Projects

We are happy when our code just runs

R has given us the tools to make your projects more structured and organized

Many people converge on very similar project templates

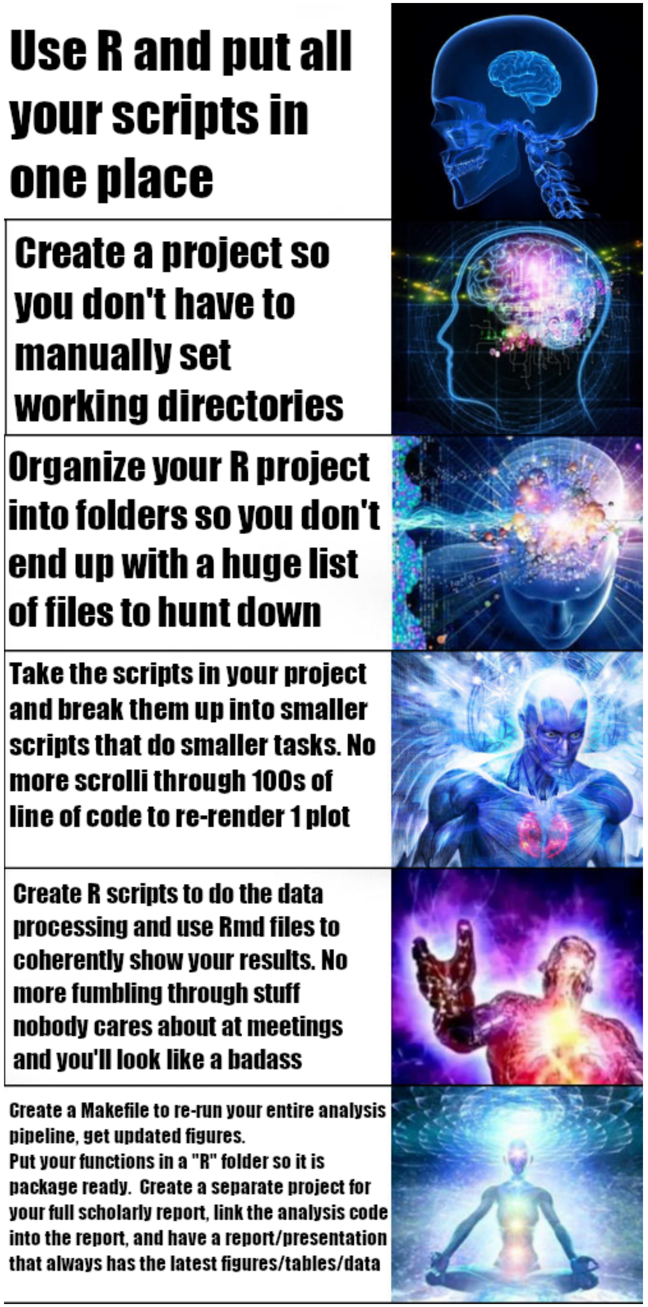
It doesn't matter where you are in your learning path

tl;dr

> I just want stuff to run the first time around

In sum...

1. Use R
2. Make a project
3. Organize the project into folders and use `here::here()` to get project relative paths
4. Break up scripts into smaller pieces
5. RMarkdown for things you want to show
6. Put functions in `R` so your analysis is package ready and write `Makefiles`, shell scripts, or other build scripts and link your projects to scholarship so your figures and tables are always up to date


Use R and put all your scripts in one place
Create a project so you don't have to manually set working directories
Organize your R project into folders so you don't end up with a huge list of files to hunt down
Take the scripts in your project and break them up into smaller scripts that do smaller tasks. No more scrolli through 100s of line of code to re-render 1 plot
Create R scripts to do the data processing and use Rmd files to coherently show your results. No more fumbling through stuff nobody cares about at meetings and you'll look like a badass
Create a Makefile to re-run your entire analysis pipeline, get updated figures. Put your functions in a "R" folder so it is package ready. Create a separate project for your full scholarly report, link the analysis code into the report, and have a report/presentation that always has the latest figures/tables/data

Where we originally started

1. A (billboard) dataset
2. A script that had everything in it

https://github.com/chendaniely/rstatsdc_2018-structure/blob/master/01-just_starting_out/analysis.R

Reproducibility vs Replicability

- **reproducibility** - the extent to which consistent results are obtained when an experiment is repeated
- **replicability** - the ability of a scientific experiment or trial to be repeated to obtain a consistent result

```
☺ find . -type f -not -path './.Rproj*'
./analysis/billboard_eda/03-eda.Rmd
./analysis/billboard_eda/02-01-clean.R
./analysis/billboard_eda/02-02-tidy.R
./analysis/billboard_eda/02-03-normalize.R
./analysis/billboard_eda/01-load.R
./analysis/billboard_eda/04-model.Rmd
./Makefile
./output/billboard_reports/03-eda.html
./output/billboard_reports/04-model.html
./output/billboard_model_plots/coefs_predict_rank_week.png
./output/billboard_model_plots/coefs_predict_rank_week_no_intercept.
./output/billboard_model_plots/coefs_predict_rank_week_artist_sorted.
./output/billboard_model_plots/coefs_predict_rank_week_artist.png
./output/billboard_rank_plots/avg_rnk_by_week.png
./output/billboard_rank_plots/avg_rnk_by_month.png
./output/billboard_rank_plots/avg_rank_by_week_across_months.png
./data/original/billboard.csv
./data/processed/billboard/billboard_clean.csv
./data/processed/billboard/billboard.csv
./data/processed/billboard/.gitkeep
./data/processed/billboard/songs.csv
./data/processed/billboard/billboard_tidy.csv
./data/processed/billboard/rank.csv
./ex-01-make.Rproj
```

Running scripts from command line

- Running form the root project directory

```
$ Rscript ./analysis/billboard_eda/01-load.R
```

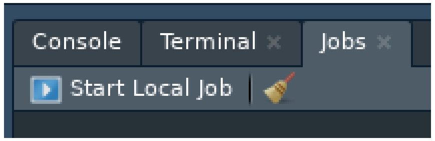
- Running the script with `here` and `rprojroot` libraries

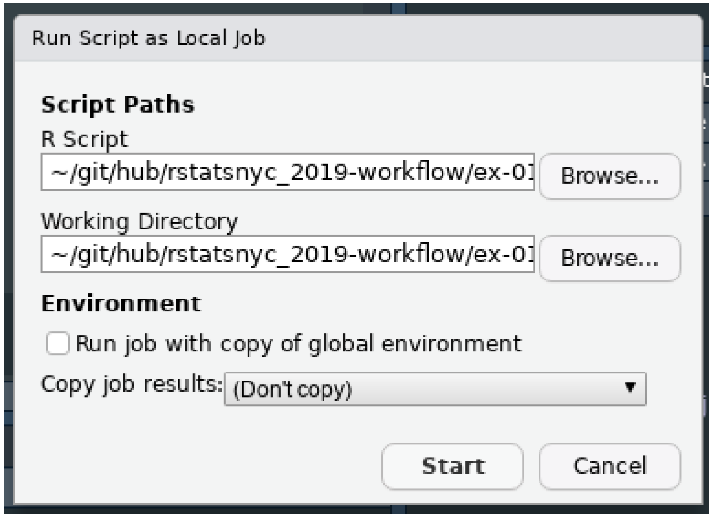
```
$ cd ./analysis/billboard_eda
$ Rscript 01-load.R
```

- Background the script

```
$ nohup Rscript 01-load.R &
```

Jobs tab in RStudio!




Run Script as Local Job

Script Paths

R Script: ~/git/hub/rstatsnyc_2019-workflow/ex-01

Working Directory: ~/git/hub/rstatsnyc_2019-workflow/ex-01

Environment

☐ Run job with copy of global environment

Copy job results: (Don't copy)

Build scripts (Make)

A task consists of:

```
target: dependency1 dependency2
    action1
    action2
```

Build scripts (Make)

```
BILLBOARD=./analysis/billboard_eda/

all : commands

## commands        : show all commands.
commands :
    @grep -E '^##' Makefile | sed -e 's/## //g'

## billboard_eda   : re-generate billboard eda analsyis
billboard_eda :
    Rscript ${BILLBOARD}/01*
    Rscript ${BILLBOARD}/02-01*
    Rscript ${BILLBOARD}/02-02*
    Rscript ${BILLBOARD}/02-03*
    Rscript -e "rmarkdown::render(here::here('./analysis/billboard_ed
    Rscript -e "rmarkdown::render(here::here('./analysis/billboard_ed

## clean           : clean up junk files.
clean :
    find data/processed/ -type f -name '*.csv' | xargs rm
    find analysis/ type f -name '*.html' | xargs rm
```

File inputs

file	input
01-load.R	'./data/original/billboard.csv'
02-01-clean.R	'./data/processed/billboard/billboard.csv'
02-02-tidy.R	'./data/processed/billboard/billboard_clean.csv'
02-03-normalize.R	'./data/processed/billboard/billboard_tidy.csv'
03-eda.Rmd	'./data/processed/billboard/rank.csv'
04-model.Rmd	'./data/processed/billboard/billboard_tidy.csv'

File outputs

file	output
01-load.R	'./data/processed/billboard/billboard.csv'
02-01-clean.R	'./data/processed/billboard/billboard_clean.csv'
02-02-tidy.R	'./data/processed/billboard/billboard_tidy.csv'
02-03-normalize.R	'./data/processed/billboard//songs.csv'; ./data/processed/billboard/rank.cs
03-eda.Rmd	'./output/billboard_rank_plots/avg_rnk_by_week.png'; ./output/billboard_rank_plots/avg_rnk_by_month.png; ./output/billboard_rank_plots/avg_rank_by_week_across_months.png
04-model.Rmd	'./output/billboard_model_plots/coefs_predict_rank_week.png'; ./output/billboard_model_plots/coefs_predict_rank_week_no_intercept.pr ./output/billboard_model_plots/coefs_predict_rank_week_artist.png

Draw your build scripts (Make)

Build scripts (Make) -- "better" Variables

```
BILLBOARD=./analysis/billboard_eda
DO=./data/original
DP_BILLBOARD=./data/processed/billboard
O_BILLBOARD_PLOTS=./output/billboard_rank_plots
O_BILLBOARD_REPORTS=./output/billboard_reports
```

Build scripts (Make) -- "better" tidy

```
## tidy              : make tidy dataset for analysis
.PHONY: tidy
tidy: $(DP_BILLBOARD)/billboard_tidy.csv \
      $(DP_BILLBOARD)/songs.csv \
      $(DP_BILLBOARD)/rank.csv

$(DP_BILLBOARD)/billboard.csv: $(DO)/billboard.csv $(BILLBOARD)/01-lo
    Rscript $(BILLBOARD)/01-load.R
$(DP_BILLBOARD)/billboard_clean.csv: $(DP_BILLBOARD)/billboard.csv \
                                     $(BILLBOARD)/02-01*.R
    Rscript $(BILLBOARD)/02-01*.R
$(DP_BILLBOARD)/billboard_tidy.csv: $(DP_BILLBOARD)/billboard_clean.c
                                    $(BILLBOARD)/02-02*.R
    Rscript $(BILLBOARD)/02-02*.R
$(DP_BILLBOARD)/songs.csv: $(DP_BILLBOARD)/billboard_tidy.csv \
                           $(BILLBOARD)/02-03*.R
    Rscript $(BILLBOARD)/02-03*.R
$(DP_BILLBOARD)/rank.csv: $(DP_BILLBOARD)/billboard_tidy.csv \
                          $(BILLBOARD)/02-03*.R
    Rscript $(BILLBOARD)/02-03*.R
```

Build scripts (Make) -- "better" eda and model

```
## eda                  : create the eda report
.PHONY: eda
eda: $(DP_BILLBOARD)/rank.csv \
     $(O_BILLBOARD_REPORTS)/03-eda.html

$(O_BILLBOARD_REPORTS)/03-eda.html: $(BILLBOARD)/03-eda.Rmd
    Rscript -e "rmarkdown::render(here::here('./analysis/billboard_ed

## model                : create the model report
.PHONY: model
model: $(DP_BILLBOARD)/billboard_tidy.csv \
       $(O_BILLBOARD_REPORTS)/04-model.html
```

Build scripts (Make) -- "better" eda and model

```
## eda                 : create the eda report
.PHONY: eda
eda: $(DP_BILLBOARD)/rank.csv \
     $(O_BILLBOARD_REPORTS)/03-eda.html

$(O_BILLBOARD_REPORTS)/03-eda.html: $(BILLBOARD)/03-eda.Rmd
    Rscript -e "rmarkdown::render(here::here('./analysis/billboard_ec

## model               : create the model report
.PHONY: model
model: $(DP_BILLBOARD)/billboard_tidy.csv \
       $(O_BILLBOARD_REPORTS)/04-model.html
```

Build scripts (Make) -- "better" reports

```
## reports          : create the eda and model reports
.PHONY: reports
reports: $(O_BILLBOARD_REPORTS)/03-eda.html \
         $(O_BILLBOARD_REPORTS)/04-model.html

## tidy_and_reports: run the entire tidy and report pipeline
.PHONY: tidy_and_reports
tidy_and_reports: tidy reports
```

```
make -f Makefile2 clean
make -f Makefile2 tidy_and_reports
# make -f Makefile2 tidy
# make -f Makefile2 reports
```

Introducing ... drake!

```r
# Install the latest stable release from CRAN.
install.packages("drake")
```

- Just like `make` it will only run scripts for things that are out-of-date.
- If something does not need to be re-run, it won't re-run it.
- The manual: https://ropenscilabs.github.io/drake-manual
- The docs: https://ropensci.github.io/drake/

But... It's not exactly like make

- drake plan: data frame with columns named target and command
- target: R object
- command: expression to produce it

A drake plan

```r
plan <- drake_plan(
  raw_data = readxl::read_excel(file_in("raw_data.xlsx")),
  data = raw_data %>%
    mutate(Species = forcats::fct_inorder(Species)),
  hist = create_plot(data),
  fit = lm(Sepal.Width ~ Petal.Width + Species, data),
  report = rmarkdown::render(
    knitr_in("report.Rmd"),
    output_file = file_out("report.html"),
    quiet = TRUE
  )
)
plan
#> # A tibble: 5 x 2
#>   target   command
#>   <chr>    <expr>
#> 1 raw_data readxl::read_excel(file_in("raw_data.xlsx"))
#> 2 data     raw_data %>% mutate(Species = forcats::fct_inorder(Spe
#> 3 hist     create_plot(data)
#> 4 fit      lm(Sepal.Width ~ Petal.Width + Species, data)
#> 5 report   rmarkdown::render(knitr_in("report.Rmd"), output_file
```

What about all my scripts?

drake plans rely more on in-memory functions

```r
bad_plan <- drake_plan(
  my_data = source(file_in("get_data.R")),
  my_analysis = source(file_in("analyze_data.R")),
  my_summaries = source(file_in("summarize_data.R"))
)
```

I personally can't use this

- I'd have to re-write all my scripts
- Patch really bad code
  - wrapping each of my scripts around a funciton call
  - still need to point out my inputs and outputs

Instead of just complaining...

- Project I'm working on this summer (RStudio) -- `grader`
- Automatically grade student's code given solution code and report why/where the code is wrong
- Given a script, what are the file inputs and outputs
    - More than just string extration

The plan

- Use Abstract Syntax Trees (ASTs)

```
library(lobstr)

ast(a + b + c)
#> █─`+`
#> ├─█─`+`
#> │ ├─a
#> │ └─b
#> └─c
```

- Use implement my find input/output script dectector
- Make my workflow work in `drake`?

Thanks!

@chendaniely

Slides: https://github.com/chendaniely/rstatsnyc_2019-workflow