TEACHING  USING THE GitHub ECOSYSTEM

Colin Rundel

UseR! 2015 - Aalborg

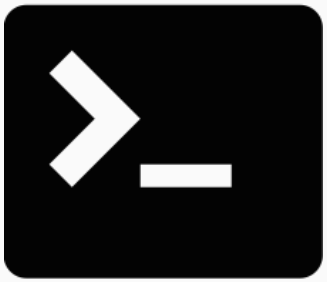
Duke University
Department of Statistical Science

STA 523 - STATISTICAL PROGRAMMING

Course details:

- First offered in Fall 2014
- Core course in Masters of Statistical Science Program
- Approximately 30 Students
  - 2/3 MSS & MSEM, 1/3 other MS & PhD
  - Divided into teams of 3-4
  - Disparate backgrounds
- Biweekly team programming assignments
- Team final project, individual final exam

TECHNICAL LEARNING OBJECTIVES



METHODOLOGICAL LEARNING OBJECTIVES

Collaboration

Reproducible Research

- R Markdown / knitr
- GNU Make

Data in the Real World

- Messy data
- Non-flat data

INFRASTRUCTURE

Dedicated departmental server

- RStudio Server Pro
- Individual departmental accounts
- System wide install of core packages

Github Organization

- 1 private repo / team (Github Education)
- Shared public repos (e.g. examples)

Continuous Integration

- TravisCI, Wercker, Drone, etc.

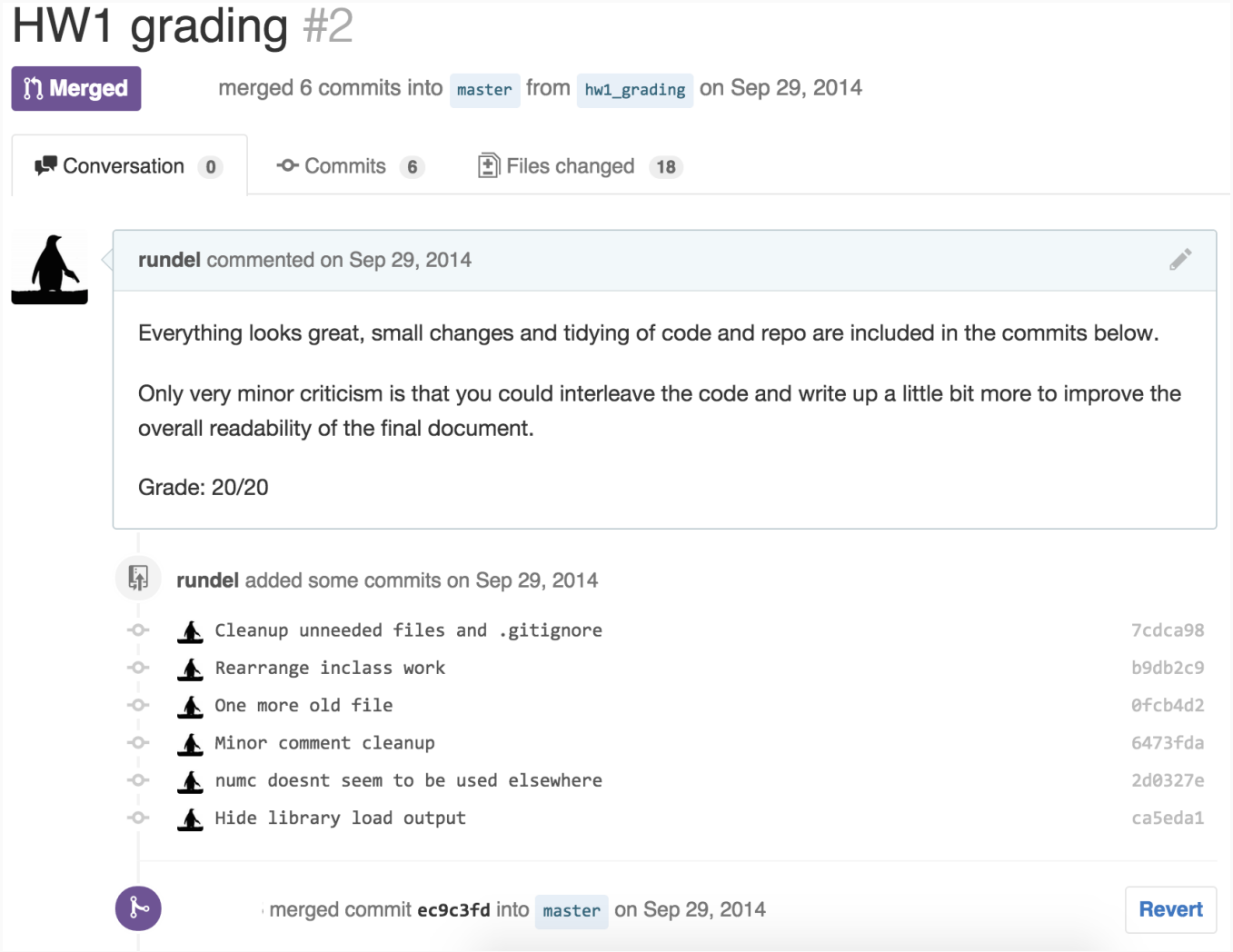
ASSIGNMENTS

All assignments are turned in via github (pull the repo at the deadline)

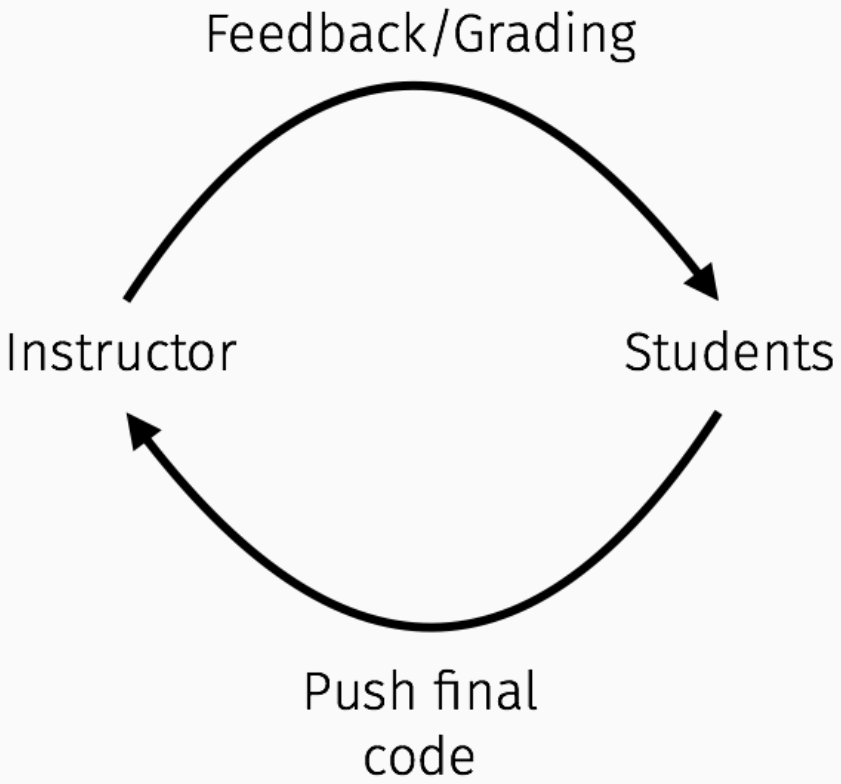
What do we get from this?

- Forces students to use version control
- Simplifies course administration
  - Code / documentation / scaffolding all in the same place
  - Easy to grab files (pull)
  - Easy to distribute files (push)
- Searchability
- Accountability

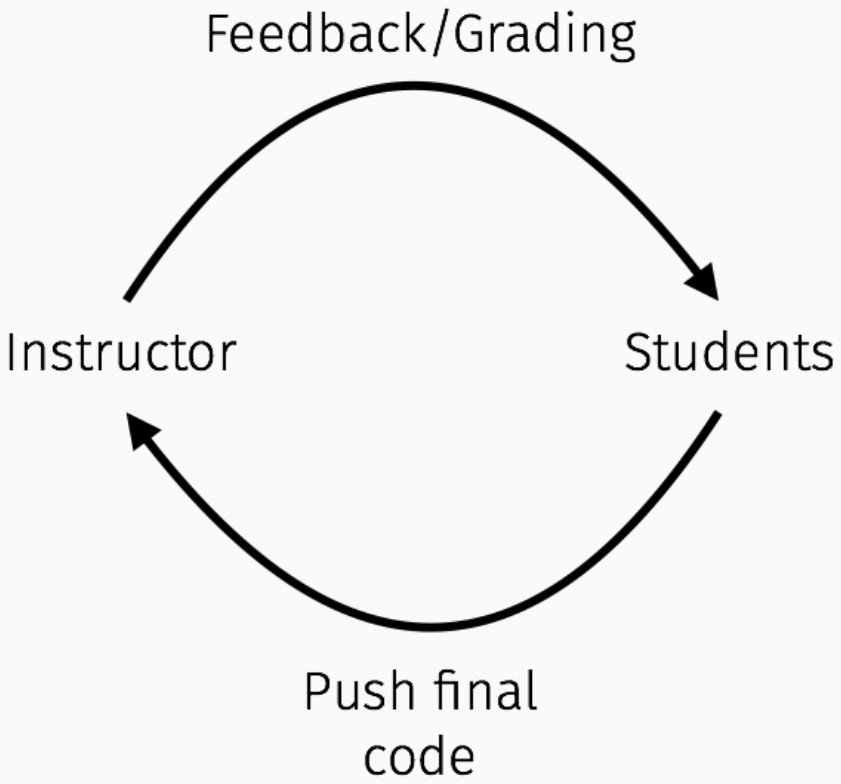
Grading and feedback is given via pull requests



COURSE PROCESS CARTOON

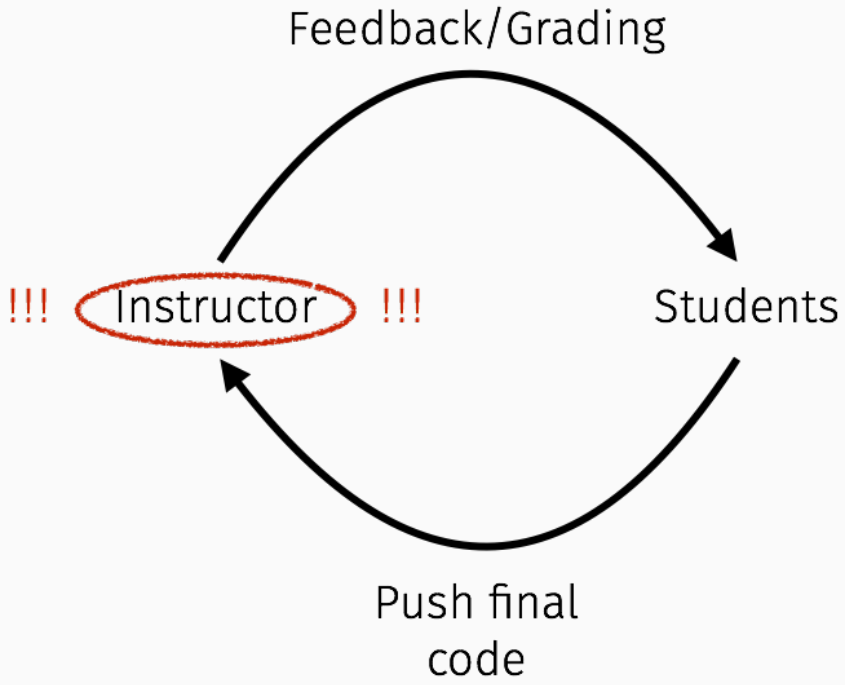


COURSE PROCESS CARTOON



Github does improve both parts of this cycle

COURSE PROCESS CARTOON



Github does improve both parts of this cycle but doesn't address the fact that *the instructor / TAs are the rate limiting step* (we don't scale well).

A PAINFULLY COMMON CONVERSATION

Student: We've submitted our HW3!

+1 Day

Me: Your Rmd file doesn't knit, you used `setwd` *with an absolute path.*

+1 Day

Student: Ok we fixed that, does it work now?

+1 Day

Me: Nope, you used `lme4` *without checking if it was installed.*

+1 Day

⋮

A SOLUTION?

In the limited universe of this class,

$$\text{Me} = \text{CRAN}$$

$$\text{Student} = \text{Package Developer}$$

so what we want is some variant of `R CMD check`.

CHECK FEATURES

What should our `R CMD check` look like?

CHECK FEATURES

What should our `R CMD check` look like?

Ideally it should check that . . .

- the code runs, Rmds knit
- the coding style is consistent
- the repo is tidy
- the code runs in a reasonable time frame
- the implementations are correct

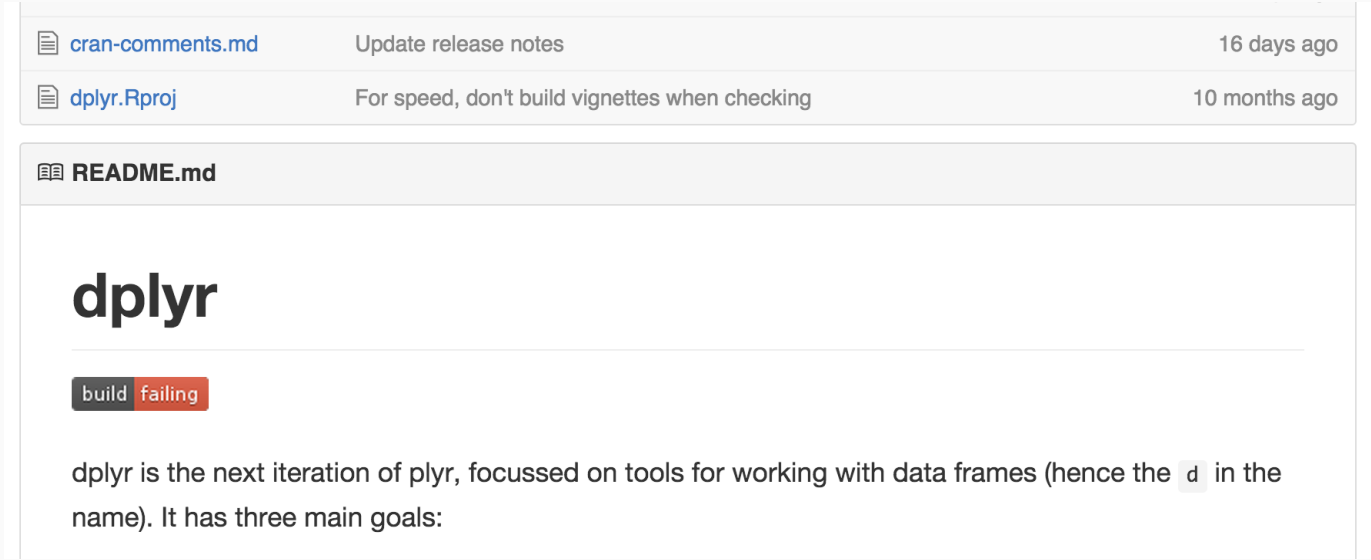
AUTOMATION

We would also like this to be done automatically.

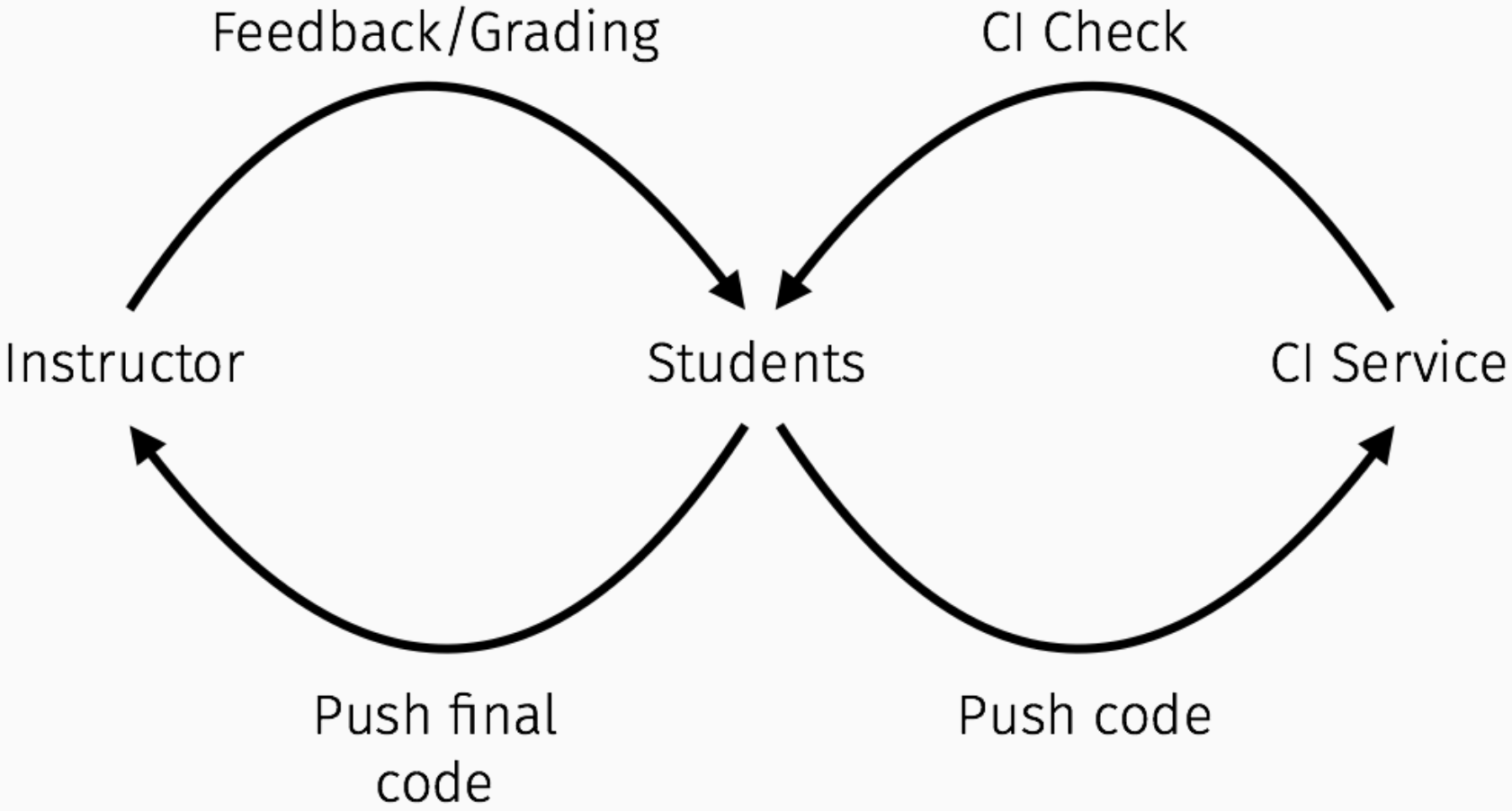
AUTOMATION

We would also like this to be done automatically.

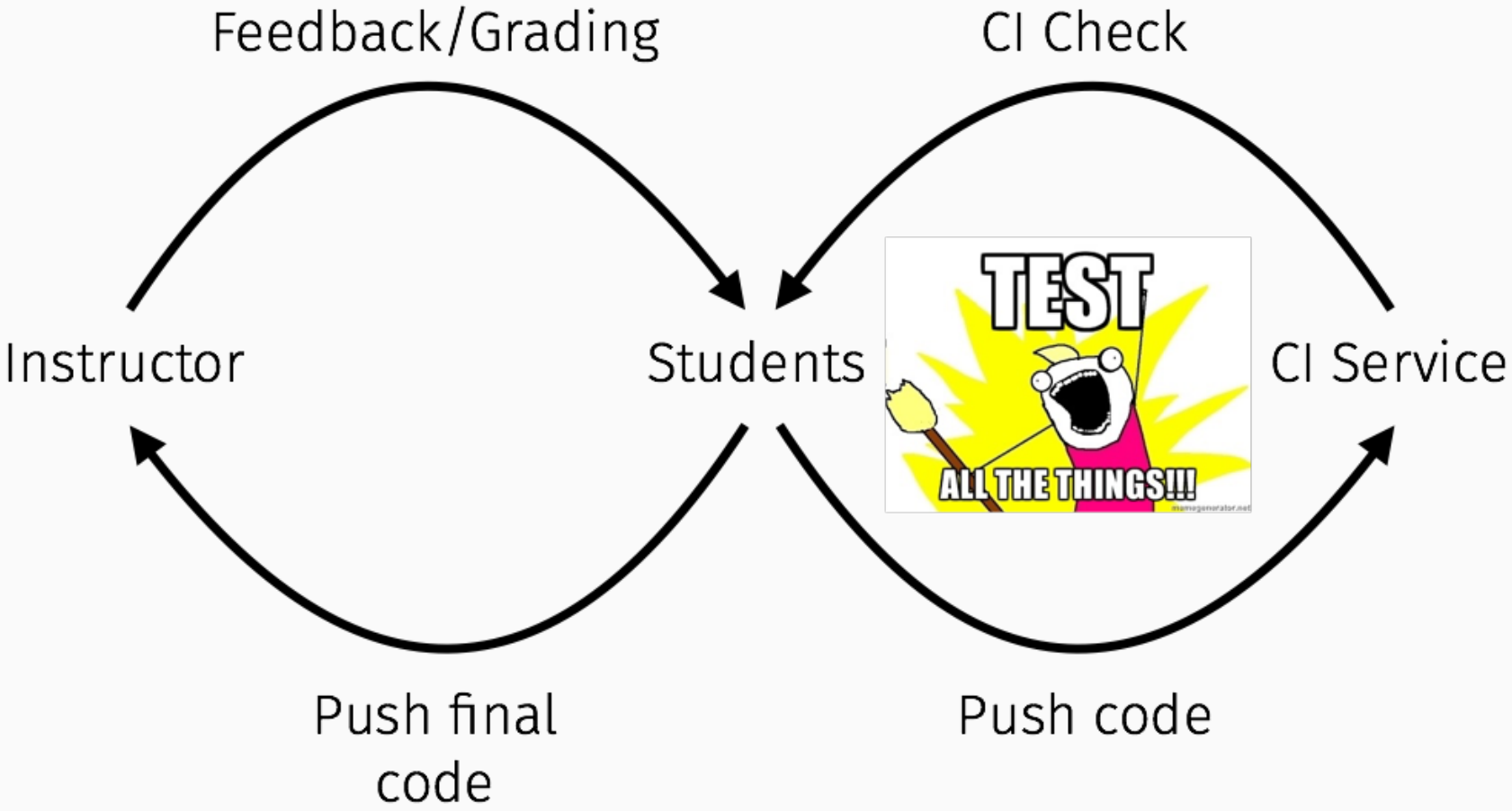
Again we can take a lesson from the package developers.



COURSE PROCESS CARTOON - IMPROVED



COURSE PROCESS CARTOON - IMPROVED



IMPLEMENTATION

This is currently more aspirational than reality, but the following is planned for the coming Fall 2015 semester.

Key details (subject to change):

- Adopting wercker for CI (uses Docker, steps)
- Enforced coding style via `lintr`
- Enforced directory structure
- Allowed file/filetype whitelist
- `testthat` for testing implementation assignments
- automated scoring of prediction contests

CONCLUSIONS

Using github gives you a lot (for free) . . .

- Version control
- Accessible web UI
- Education support
- Collaboration tools
- Search tools
- CI tools

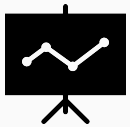
Needs of an R programming class are very similar to the needs of the R development community

- No need to reinvent the wheel - use the existing solutions
- Teach the tools students will continue to use

QUESTIONS, COMMENTS?

rundel@gmail.com

github.com/rundel/



github.com/rundel/Presentations/

bit.ly/Sta523_2014 | bit.ly/Sta523_2015