13: Summary + Extras

`bit.ly/2018rr`

The most important tool is the mindset,
when starting, that the end product
will be reproducible.

– Keith Baggerly

Steps toward reproducible research

- ▶ Slow down
- ▶ Organize; document
- ▶ Everything with code
- ▶ Scripts → RMarkdown
- ▶ Code → functions → packages
- ▶ Version control with Git
- ▶ Automation with Make
- ▶ Choose a license
- ▶ Share your work with others

Make with R Markdown

To use Make with R Markdown, you'll use a command like:

```
R -e "rmarkdown::render('my_report.Rmd')"
```

You'll need to tell your operating system where it can find pandoc. RStudio includes pandoc, but you need to add the relevant directory to your PATH.

Mac:

```
/Applications/RStudio.app/Contents/MacOS/pandoc
```

Windows:

```
"c:\Program Files\RStudio\bin\pandoc"
```

~/.bash_profile

```
export PATH=$PATH:/Applications/RStudio.app/Contents/MacOS/pandoc

noclobber=1    # prevent overwriting of files
IGNOREEOF=1    # disable Ctrl-D as a way to exit
HISTCONTROL=ignoredups

alias cl='clear;cd'
alias rm='rm -i'
alias mv='mv -i'
alias cp='cp -i'
alias ls='ls -GF'
alias 'l.'='ls -d .[a-zA-Z]*'
alias ll='ls -lh'
alias md='mkdir'
alias rd='rmdir'
alias rmb='rm .*~ *~ *.bak *.bk!'

alias Rb='R CMD build --force --resave-data'
alias Ri='R CMD INSTALL --library=/Users/kbroman/Rlibs'
alias Rc='R CMD check --library=/Users/kbroman/Rlibs'
alias Rcc='R CMD check --as-cran --library=/Users/kbroman/Rlibs'
```
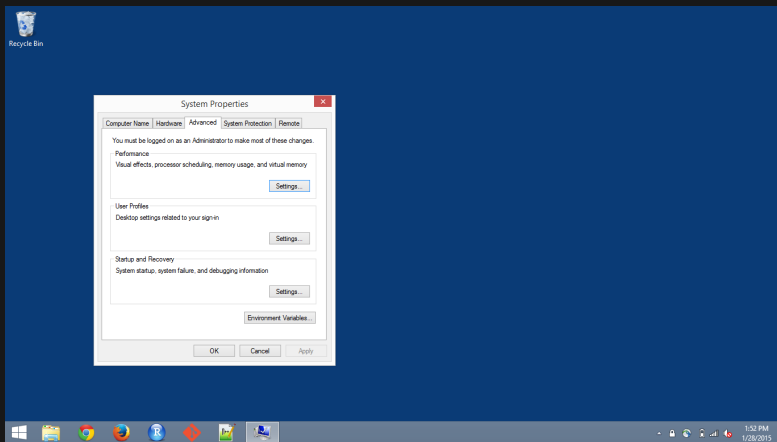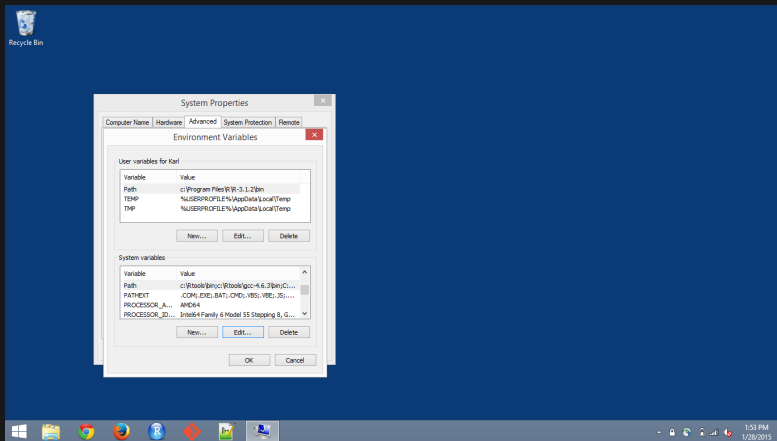
`PATH` in Windows

PATH in Windows

PATH in Windows

Challenges

- ► Daily maintenance
    - – READMEs up to date?
    - – Documentation matches code?
- ► Cleaning up the junk
    - – Move defunct stuff into an `Old/` subdirectory?
- ► Start over from the beginning, nicely?

Sharing your work

- ► Why share?
  - – Funding agency or journal requirement
  - – Increased visibility
  - – So that others can build on your work
- ► When?
  - – Continuously and instantaneously
  - – When you submit a paper
  - – When your paper appears
- ► Risks?

`bit.ly/rr_sharing_slides`

I'm not worried about being scooped,
I'm worried about being ignored.

– Magnus Nordborg

- ► Share more
- ► Share sooner
- ► Share in a way that it's easy for others to learn from and build upon

What to share?

- ► For sure
  - Primary dataset
  - Metadata
  - Data cleaning scripts
  - Analysis scripts
- ► It could help
  - Very-raw data
  - Processed/clean data
  - Intermediate results
- ► No
  - Confidential data (e.g. HIPAA data)
  - Passwords, private keys

Where to share?

- ▶ Domain-specific repository
  - – Genbank, dbGaP, etc.
  - – See `re3data.org`
- ▶ Figshare, Dryad, Zenodo
- ▶ Institutional repository
- ▶ GitHub, BitBucket

Resources

- ▶ R Markdown
  - – `rmarkdown.rstudio.com`
- ▶ R Packages
  - – Releasing to CRAN: `r-pkgs.had.co.nz/release.html`
  - – Leek group: `github.com/jtleek/rpackages`
  - – When to trust an R package: `bit.ly/trust_r_pkg`
- ▶ Make
  - – `kbroman.org/minimal_make`
  - – remake R package, `github.com/richfitz/remake`
- ▶ Git
  - – Git branches: `nicercode.github.io/git/branches.html`
  - – Hadley on Git/GitHub: `r-pkgs.had.co.nz/git.html`
  - – Git subtrees: `bit.ly/git_subtree`
- ▶ Also see `bit.ly/sisbid3_resources`

Some of the things we didn't cover

R CMD BATCH

Why is it cool?
– Scripting lets you run everything in the background
– Increases the odds you've got *everything* reproducible

Why didn't we cover it?
– A bit geek-heavy

Where would we point you?
– Stackoverflow discussion: `bit.ly/so_rscript`
– Codecademy Learn the Command Line lesson:
  `bit.ly/learn_command_line`

Coding conventions

Why are they cool?
– They help you keep things consistent between team members
– They make code easier to read, and more likely to be used

Why didn't we cover them?
– Not enough time

Where would we point you?
– Hadley's recommendations `adv-r.had.co.nz/Style.html`
– Google's recommendations
`google.github.io/styleguide/Rguide.xml`

Code review

Why is it cool?
– Helps to find bugs and clean up confusing bits
– Potentially a test of the reproducibility of your work

Why didn't we cover it?
– Not enough time

Where would we point you?
– Software Carpentry blog post, `bit.ly/swc_codereview`
– Titus Brown's blog post,
  `http://bit.ly/titus_codereview`

Software testing

Why is it cool?
– Explicit tests help you to avoid bugs, and to find bugs sooner

Why didn't we cover it?
– Not enough time

Where would we point you?
– testthat package, `github.com/hadley/testthat`
– Testing R Code book

Continuous integration (eg Travis)

Why is it cool?

– Automatically build and run tests when you push to GitHub

Why didn't we cover it?

– Not enough time

Where would we point you?

– Julia Silge blog post,
  `juliasilge.com/blog/beginners-guide-to-travis`
– Hadley's R packages book,
  `r-pkgs.had.co.nz/check.html#travis`

Capturing dependencies

Why is it cool?

– Ensure that your carefully constructed reproducible project doesn't fail due to a change in one of the packages you use

Why didn't we cover it?

– Not enough time

Where would we point you?

– packrat package, `github.com/rstudio/packrat`
– checkpoint package, `github.com/RevolutionAnalytics/checkpoint`

Containers (e.g. docker)

Why are they cool?
– Capture your entire environment, so your project is *for sure* fully reproducible.

Why didn't we cover them?
– A bit technical

Where would we point you?
– Rocker: Docker for R
– R Docker tutorial

R Markdown templates

Why are they cool?
– More complete control over the appearance of your document

Why didn't we cover them?
– A bit technical

Where would we point you?
– R Markdown documentation

knitr Bootstrap

Why is it cool?
  – Allows for generation of slicker reports

Why didn't we cover it?
  – A bit technical

Where would we point you?
  – github.com/jimhester/knitrBootstrap

GitHub pages

Why are they cool?
– Webpages built entirely in Markdown, providing nicer interfaces to your content

Why didn't we cover them?
– Tangential to *reproducible research*?

Where would we point you?
– `pages.github.com`
– `kbroman.org/simple_site`
– `bookdown.org/yihuit/blogdown`

Bookdown

Why is it cool?
– Write a book (or book-like object) entirely in R Markdown

Why didn't we cover it?
– Not enough time

Where would we point you?
– `bookdown.org/yihui/bookdown`

Shiny!

Why is it cool?
– Interactive pictures have pizzazz.

Why didn't we cover it?
– Tangential to *reproducible research*?

Where would we point you?
– `shiny.rstudio.com`
– `shiny.rstudio.com/tutorial`

Feedback we'd like from you (1)

What motivated us to teach this course?
What would we see as a positive outcome?

- ▶ Given this motivation, are we doing things right?
- ▶ What motivated you to take this course?
- ▶ Were there specific sessions you found really useful/really useless?
- ▶ Points you'd like us to expand on?
- ▶ Were there points you were hoping we'd cover that we didn't?

Feedback we'd like from you (2)

- ▶ Do you have examples/anecdotes you think we might be able to use that you'd be willing to share?
- ▶ Were there ways we could've used time more effectively?
- ▶ Were there ways we could've used our TAs more effectively?
- ▶ Can you see things you learned in this course changing how you do things day to day?
  - – Why or why not?
  - – Can we ask you again in 6 months?
  - – Can we ask you again in a year?
- ▶ Could you write this down now? (anonymous is fine)