STAT 545A
Class meeting #11
Wednesday, October 16, 2013

Dr. Jennifer (Jenny) Bryan

Department of Statistics and Michael Smith Laboratories

mostly we walked through material in a web block:
"R coding style and organizing analytical projects"

http://www.stat.ubc.ca/~jenny/STAT545A/block19_codeFormattingOrganization.html

and sometimes looked at these slides when there was info present here but not there or info presented more nicely for projection here

“Source is real.”

Philosophy practiced by the pros

"**The source code is real**. The objects are realizations of the source code. Source for EVERY user modified object is placed in a particular directory or directories, for later editing and retrieval."

-- from the ESS manual

coding style & standards

Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

- Donald E. Knuth

1. Rule of Modularity: Write simple parts connected by clean interfaces.

2. Rule of Clarity: Clarity is better than cleverness.

3. Rule of Composition: Design programs to be connected to other programs.

4. Rule of Separation: Separate policy from mechanism; separate interfaces from engines.

5. Rule of Simplicity: Design for simplicity; add complexity only where you must.

6. Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.

7. Rule of Transparency: Design for visibility to make inspection and debugging easier.

8. Rule of Robustness: Robustness is the child of transparency and simplicity.

9. Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.

10. Rule of Least Surprise: In interface design, always do the least surprising thing.

11. Rule of Silence: When a program has nothing surprising to say, it should say nothing.

12. Rule of Repair: When you must fail, fail noisily and as soon as possible.

13. Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.

14. Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.

15. Rule of Optimization: Prototype before polishing. Get it working before you optimize it.

16. Rule of Diversity: Distrust all claims for "one true way".

17. Rule of Extensibility: Design for the future, because it will be here sooner than you think.

These rules are from from a conference report "Good Programming Practices in Healthcare: Creating Robust Programs".

Load special libraries at top and remind yourself why needed

```
library(RColorBrewer)                    # will use for color-coding
                                         # continent

## 'home' directory for this analysis
whereAmI <- "/Users/jenny/teaching/2011/STAT545A/examples/gapminder/"

## data import from local file
gDat <- read.delim(jPaste(whereAmI,"data/gapminderDataFiveYear.txt"))

## reach out and touch the data
str(gDat)
## 'data.frame':        1704 obs. of  6 variables:
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ pop      : num  8425333 9240934 10267083 11537966 13079460 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 33 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ gdpPercap: num  779 821 853 836 740 ...
summary(gDat)
head(gDat)
peek(gDat)
```

Store useful info in comments; useful for quick "look up" later and for sanity checking when re-running analyses; DO NOT rely on this for anything truly important because it's not automatically updated!

Spaces around binary operators and after commas

```
plotGapminderOneYear <-
  function(jYear, gapDat, contDat,
           jXlim = c(200, 50000),
           jYlim = c(21, 84),
           jXlab = "Income per person (GDP/capita, inflation-adjusted $)",
           jYlab = "Life expectancy at birth (years)",
           jLightGray = 'grey80',
           jDarkGray = 'grey20',
           gdpTicks = c(200, 400, 1000, 2000, 4000, 10000, 20000, 40000),
           lifeExpTicks = seq(from = 20, to = 85, by = 5),
           yearCex = 15
           ) {

    ## map pop into circle radius
    jPopRadFun <- function(jPop) {
      sqrt(jPop/pi)
    }

    plot(lifeExp ~ gdpPercap, gapDat, subset = year == jYear,
         log = 'x', xlim = jXlim, ylim = jYlim,
         xaxt = "n", yaxt = "n", type = "n",
         xlab = jXlab, ylab = jYlab)
    abline(v = gdpTicks, col = jLightGray)
    abline(h = lifeExpTicks, col = jLightGray)
    text(x = sqrt(prod(jXlim)), y = mean(jYlim),
         jYear, adj = c(0.5, 0.5), cex = yearCex, col = jLightGray)
    axis(side = 1, at = gdpTicks, labels = gdpTicks)
    axis(side = 2, at = lifeExpTicks, labels = lifeExpTicks, las = 1)
    with(subset(gapDat, year == jYear),
         symbols(x = gdpPercap, y = lifeExp,
                 circles = jPopRadFun(pop), add = TRUE,
                 inches = 0.7,
                 fg = jDarkGray, bg = color))
    with(contDat,
         legend(x = 'bottomright', bty = 'n',
                legend = continent, fill = color))
  }
```

Line wrapping

Indenting, e.g. inside functions or if/then

Comments, properly indented

```
## BEGIN: detailed exploration of data

## do we have NAs?
sapply(gDat, function(x) sum(is.na(x)))
##   country      year       pop continent   lifeExp gdpPercap
##         0         0         0         0         0         0
## no NAs ... good!

## year
summary(gDat$year)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1950    1967    1982    1980    1996    2007

## confirming we have 1950, 1951, ..., 2007
all(sort(unique(gDat$year)) == 1950:2007) # TRUE
length(1950:2007)                         # 58 poss vals for year

table(gDat$year)

barchart(table(gDat$year))
## most countries have data every five years, e.g. 1952, 1957, 1962,
## and so on
dev.print(pdf,
          jPaste(whereAmI, "figs/barchartYear.pdf"),
          width = 5, height = 8)

dotplot(table(gDat$year),
        origin = 0,
        type = c("p", "h"))
dev.print(pdf,
          jPaste(whereAmI, "figs/dotplotYear.pdf"),
          width = 5, height = 8)


## country
str(gDat$country)                         # 187 countries
```

Naming conventions (iAlwaysDoItLikeThis)

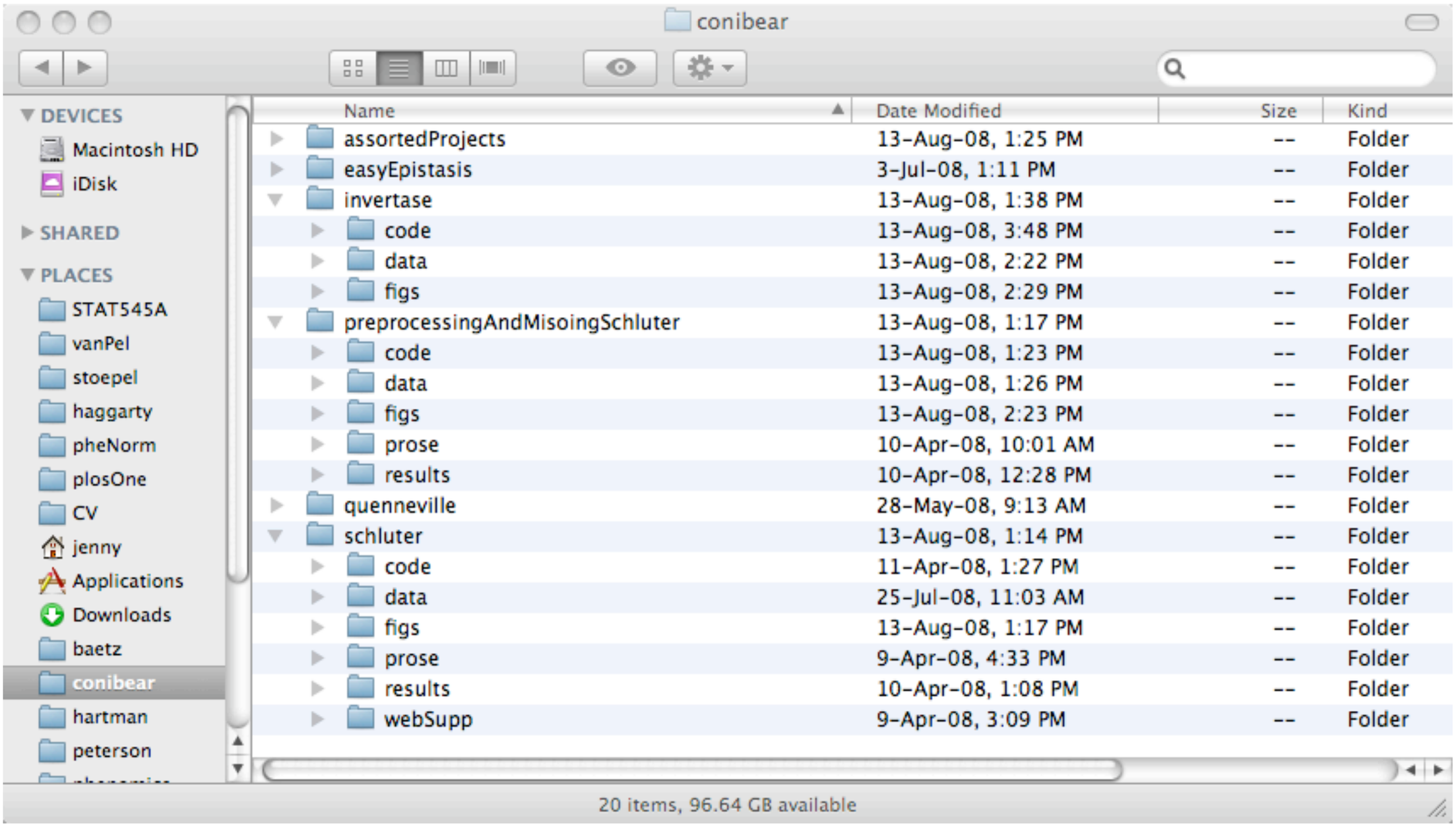
```
library(RColorBrewer)                    # will use for color-coding
                                         # continent

## 'home' directory for this analysis
whereAmI <- "/Users/jenny/teaching/2011/STAT545A/examples/gapminder/"

## data import from local file
gDat <- read.delim(jPaste(whereAmI,"data/gapminderDataFiveYear.txt"))

## reach out and touch the data
str(gDat)
## 'data.frame':        1704 obs. of  6 variables:
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ pop      : num  8425333 9240934 10267083 11537966 13079460 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 33 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ gdpPercap: num  779 821 853 836 740 ...
summary(gDat)
head(gDat)
peek(gDat)
```

I use "camelCase" to make identifier names.
The Google style guide forbids it! Illustrates that, in many details, there is no One True Way.
Pick a convention you like and then BE CONSISTENT.*

* at least most of the time ☺

How I organize my work

Contents of `/Users/jenny/research/conibear`



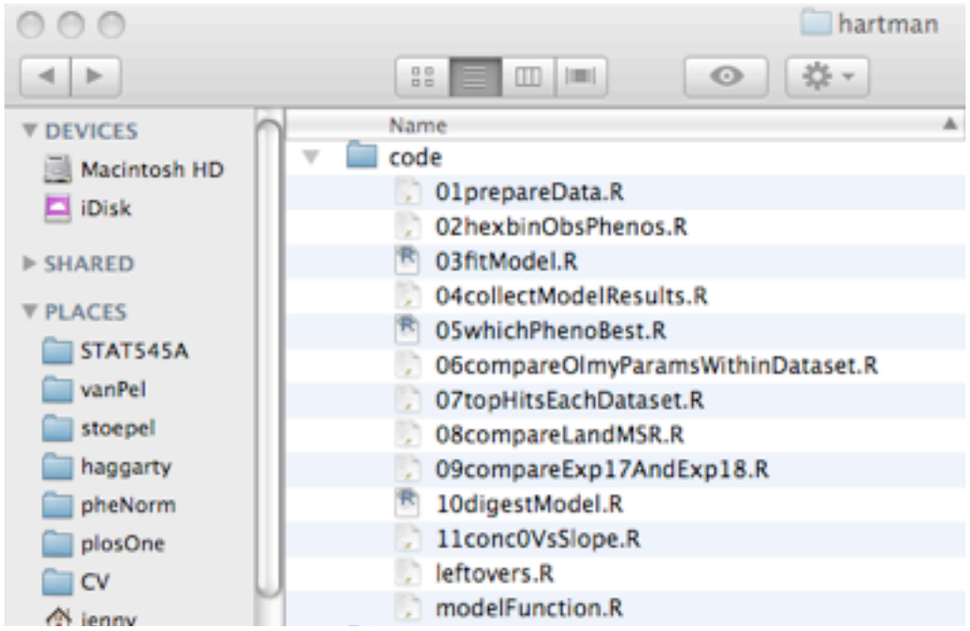
How I organize my work

- Directory name = last name of collaborator, if relevant, or one word that evokes the statistical project
- Subdirectories
  - **code** (R and Perl code, anything executed at the Unix command line during data cleaning /processing, etc.)
  - **data** (raw data from the outside world, "prepared" data after I've whipped it into shape)
  - **figs** (figures, usually in PDF form, with painfully informative names)

How I organize my work

- Subdirectories cont'd
  - **prose** (key emails, internal documentation and explanations, interim reports of analyses, talks, manuscripts, final publications)
  - **results** (mission critical intermediate and final results, generally in plain text delimited form, occasionally R objects, very rarely R workspaces)
  - **webSupp** (support for any web resource related to the project, such as sharing material with collaborators or web supplements for publications; lots of soft links to files found in other subdirectories)

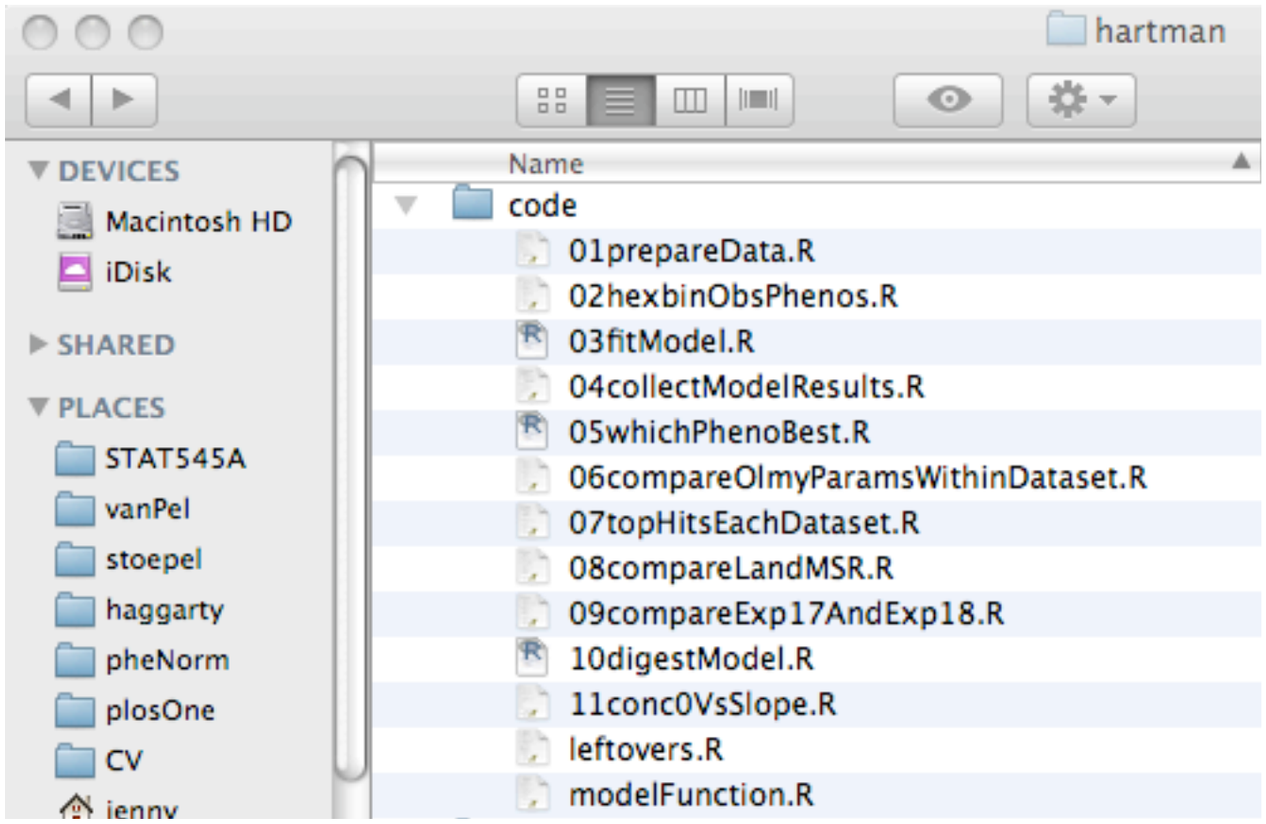
the **code** subdirectory



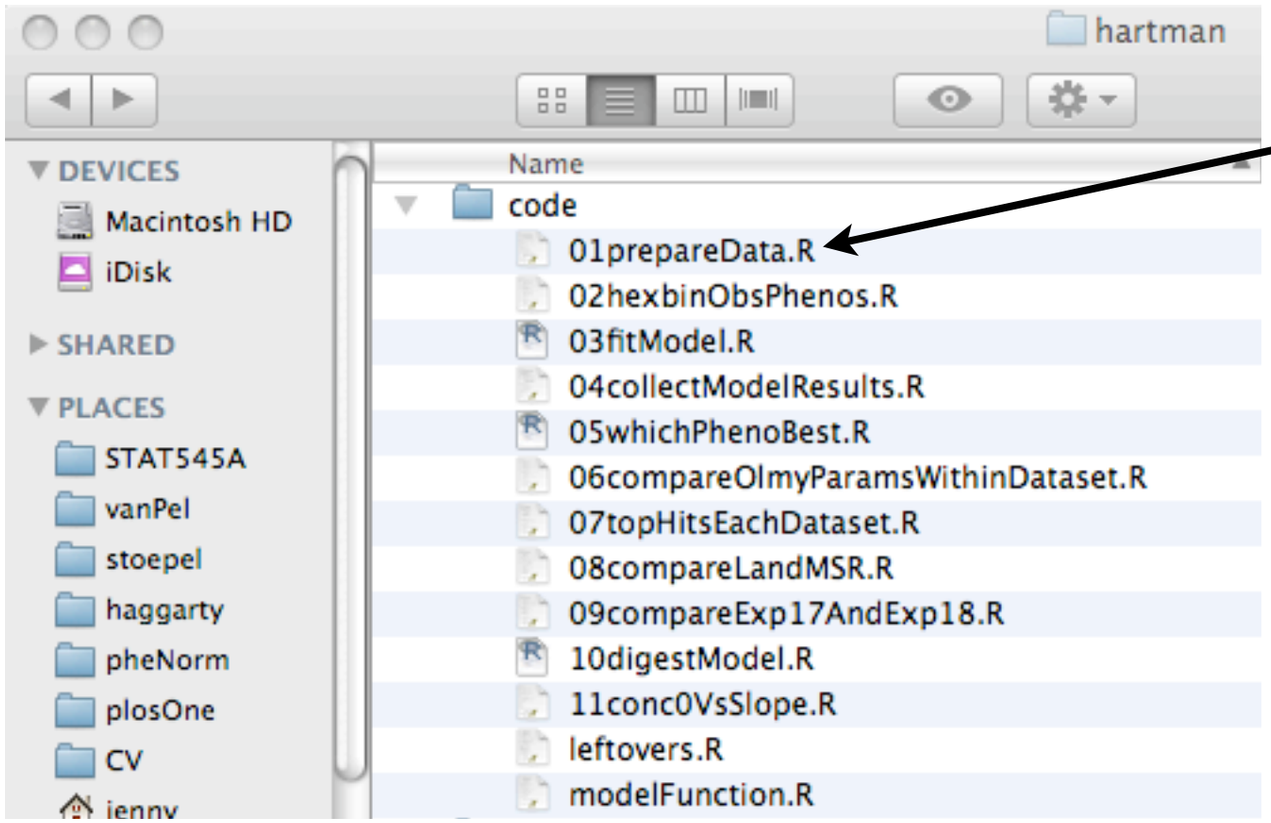

```
/Users/jenny/teaching/2011/STAT545A/examples/gapminder/code:
total used in directory 288 available 342408564
drwxr-xr-x  23 jenny  staff    782 Sep 30 13:42 .
drwxr-xr-x   7 jenny  staff    238 Mar 31  2011 ..
-rw-r--r--   1 jenny  staff   1239 Sep 30 10:33 .Rhistory
-rw-r--r--   1 jenny  staff   4878 Sep 13 13:10 bryan-a01-01-dataPrep.R
-rw-r--r--   1 jenny  staff   6250 Oct  8  2010 bryan-a01-02-dataMerge.R
-rw-r--r--   1 jenny  staff   4676 Sep 20 13:43 bryan-a01-03-dataExplore.R
-rw-r--r--   1 jenny  staff   3099 Oct  8  2010 bryan-a01-04-fillContinentData.R
-rw-r--r--   1 jenny  staff   4490 Sep 20  2010 bryan-a01-05-everyFiveYears.R
-rw-r--r--   1 jenny  staff   7493 Sep 26 19:40 bryan-a01-10-baseGraphicsStepByStep.R
-rw-r--r--   1 jenny  staff   1453 Sep 21  2010 bryan-a01-11-baseGraphicsPlotGapminderOneYear.R
-rw-r--r--   1 jenny  staff   3787 Sep 26 19:43 bryan-a01-12-baseGraphicsSoln.R
-rw-r--r--   1 jenny  staff  13679 Sep 30 13:42 bryan-a01-15-latticeStepByStep.R
-rw-r--r--   1 jenny  staff   2554 Sep 27 21:19 bryan-a01-16-latticePlotGapminderOneYear.R
-rw-r--r--   1 jenny  staff   3878 Sep 27 21:17 bryan-a01-17-latticeSoln.R
-rw-r--r--   1 jenny  staff   7174 Sep 27 21:33 bryan-a01-18-latticeDemos.R
-rw-r--r--   1 jenny  staff    312 Sep 28  2010 bryan-a01-19-latticeStruggles.R
-rw-r--r--   1 jenny  staff   4792 Sep 26 19:44 bryan-a01-30-makeGapminderColorScheme.R
-rw-r--r--   1 jenny  staff   5722 Sep 30 13:17 bryan-a01-40-dataAggregation.R
-rw-r--r--   1 jenny  staff   6137 Sep 26 19:47 bryan-a01-50-basicColorDemo.R
-rw-r--r--   1 jenny  staff   1424 Sep 19  2010 bryan-a01-99-scratchpad.R
```

- Use .R as the suffix for plain text files holding R code
- Break your code into down into sensible pieces
- Use highly informative names, possibly with numbering to harmonize logical and alphanumeric order

the **code** subdirectory

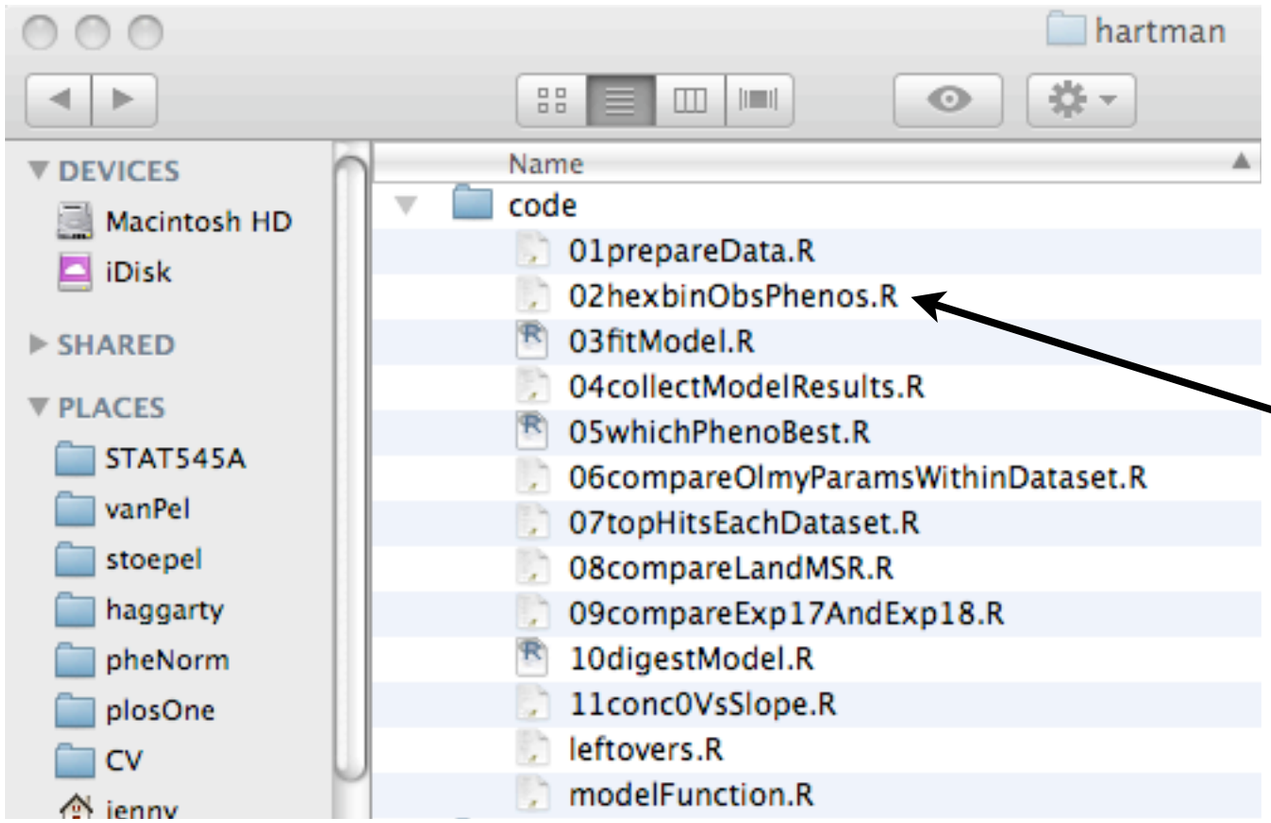


the **code** subdirectory



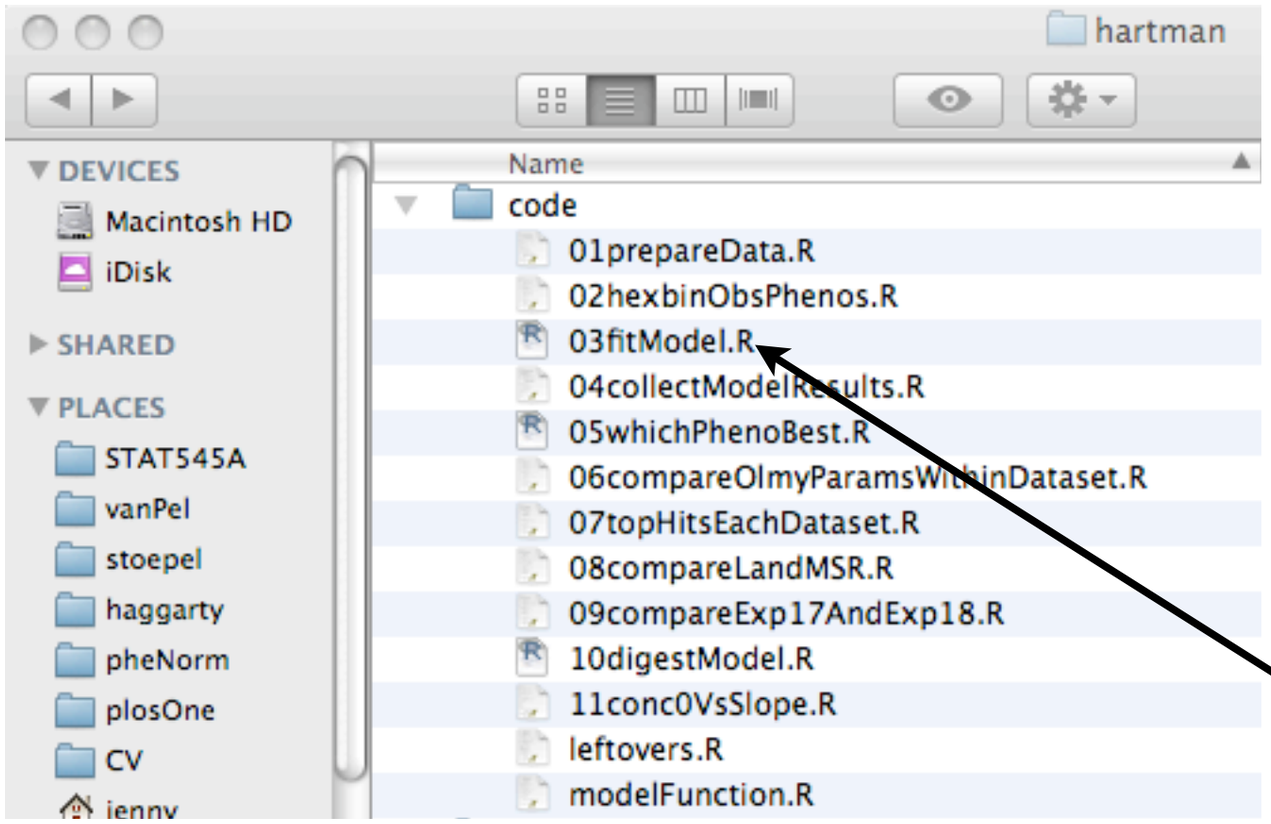
Data cleaning and prep

the **code** subdirectory



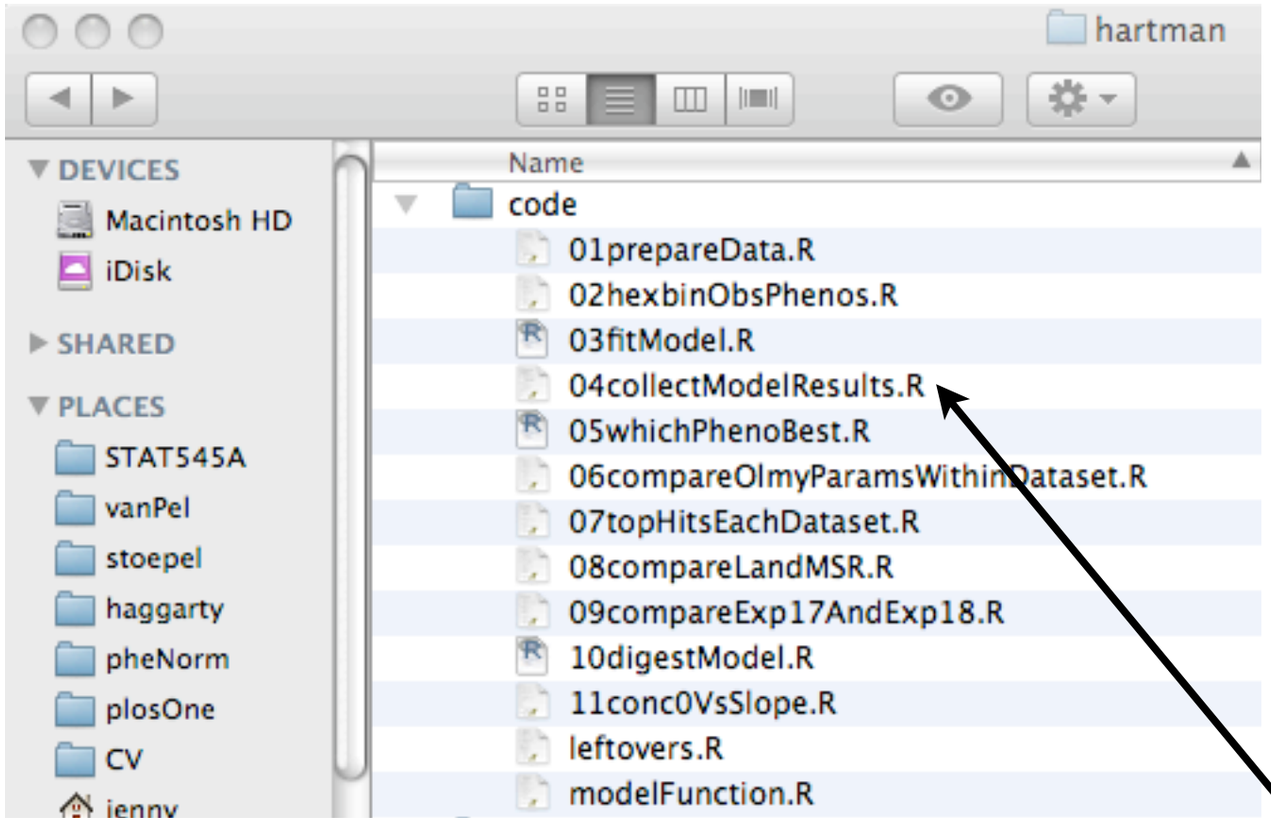
"Getting to know you" figures

the **code** subdirectory



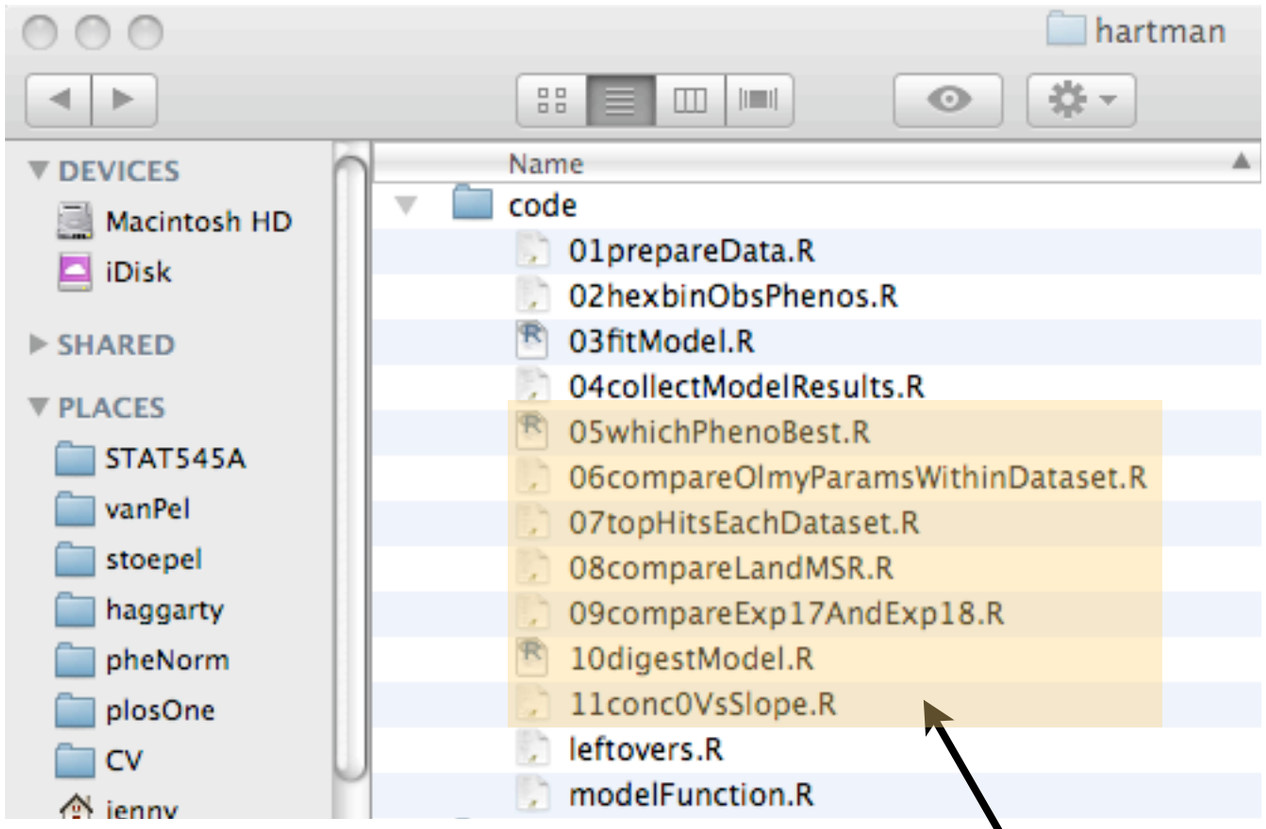
First real analysis

the **code** subdirectory



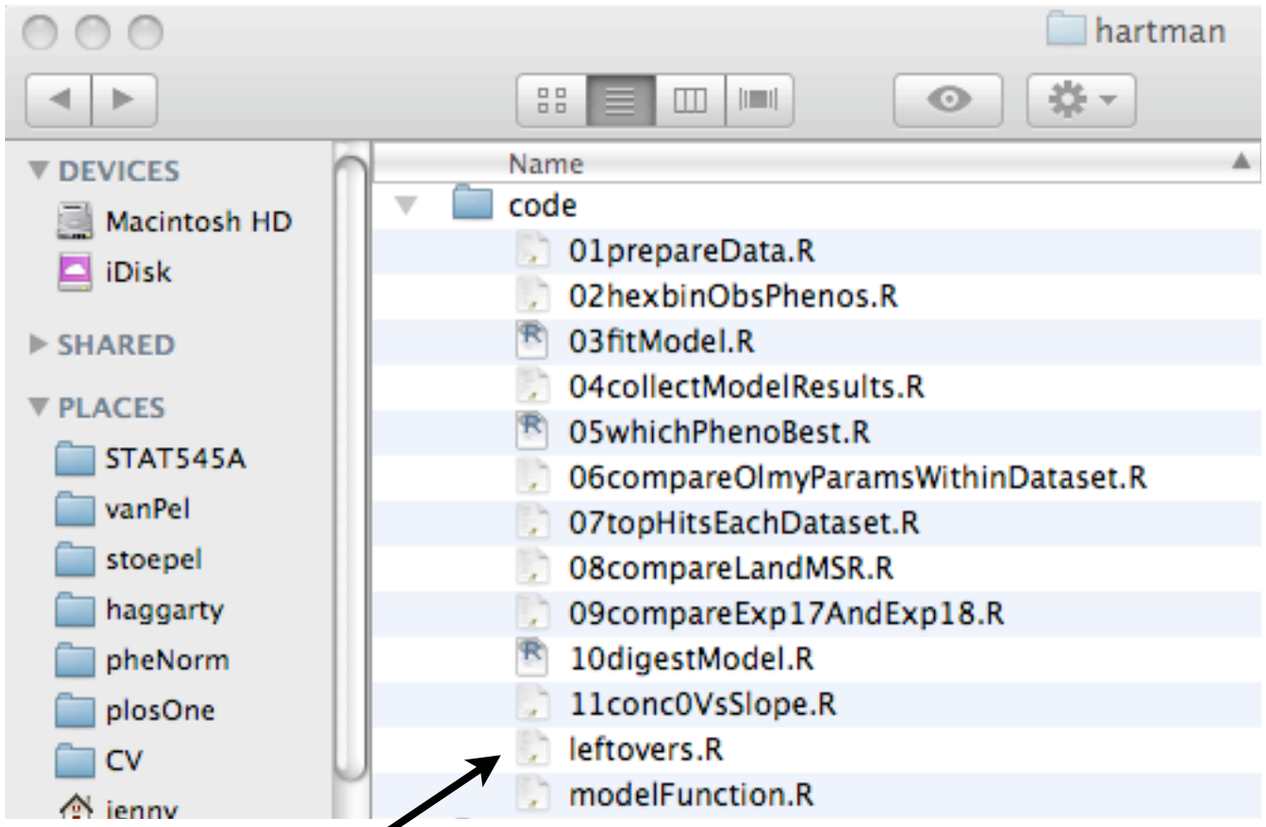
Getting analytical results into a useful form

the **code** subdirectory



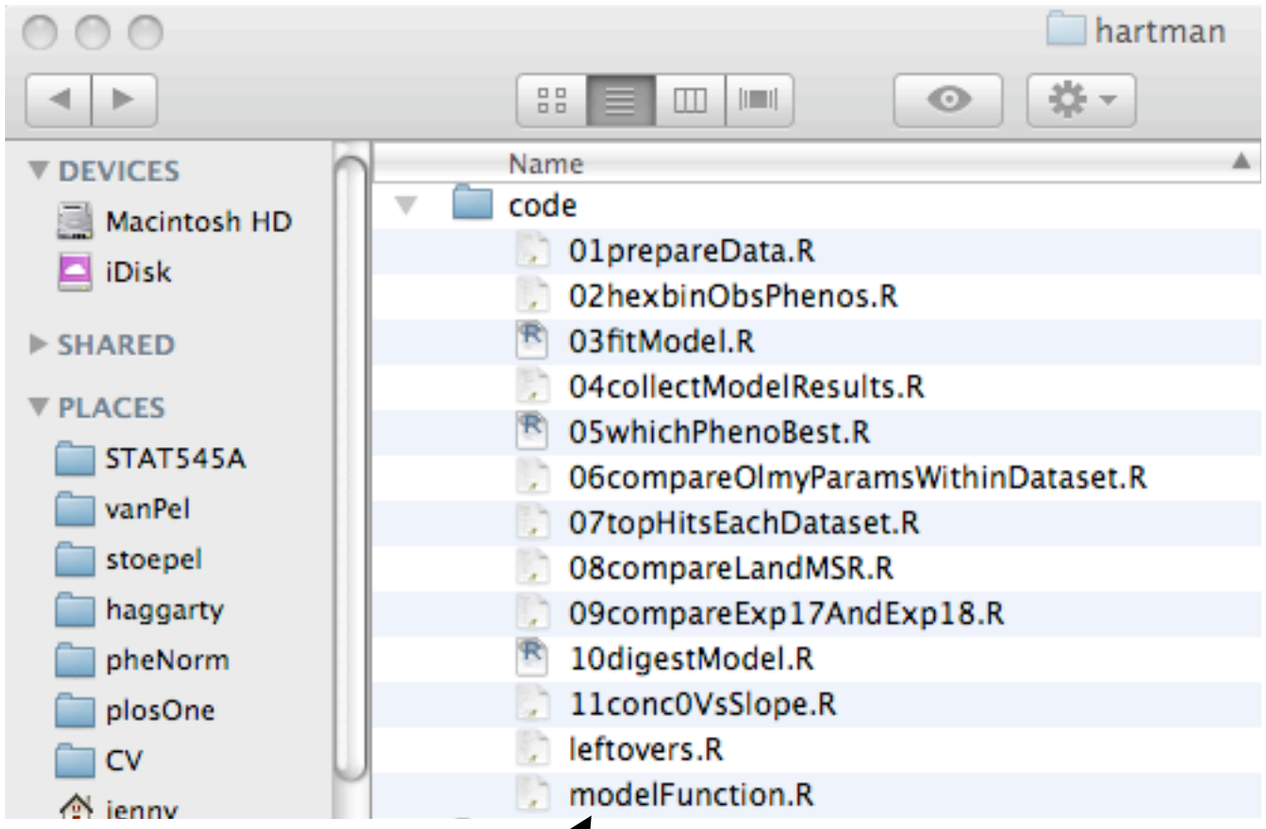
Analyzing the analytical results, including making more figures

the **code** subdirectory



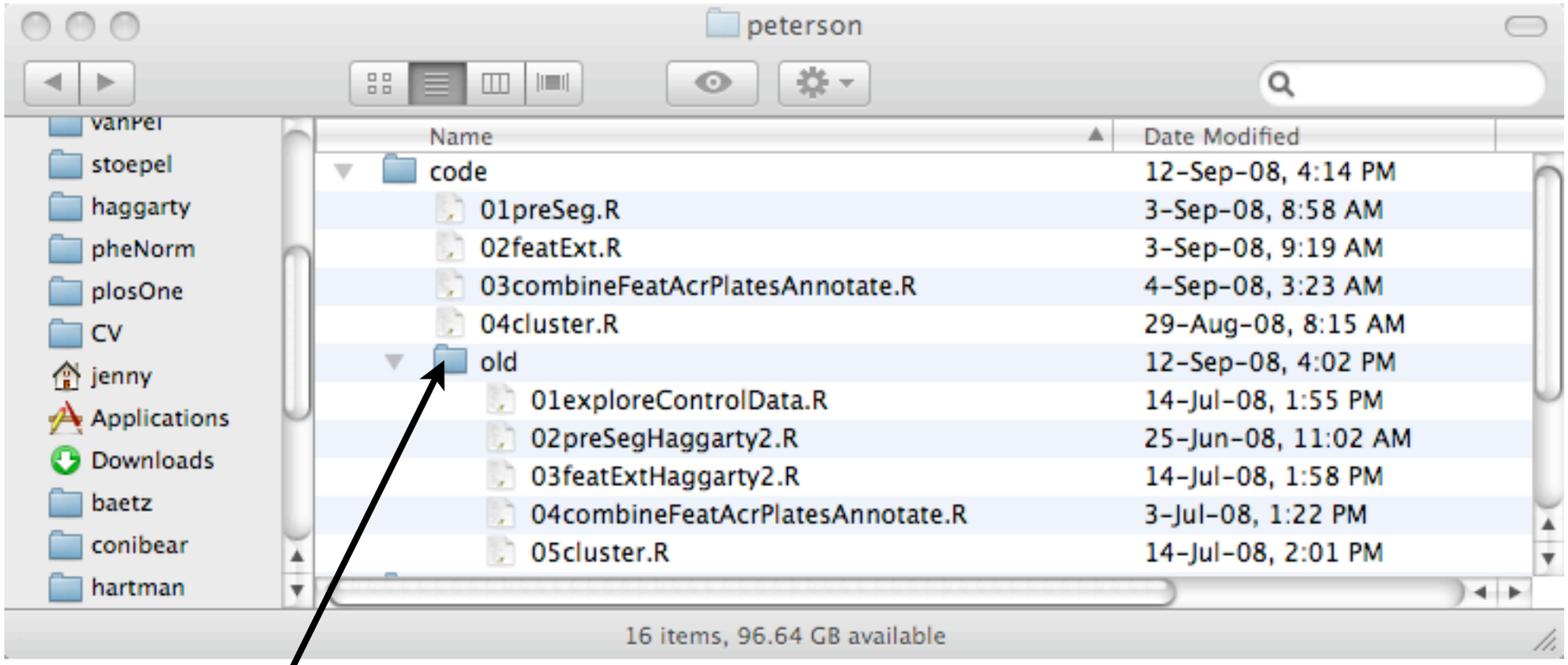
Code snippets you aren't currently using

the **code** subdirectory



Key functions

the **old** sub-subdirectory



Where files go to die ... but can still be resurrected!

Applies to code, data, figs, results, etc.

File obsolete material away religiously or you will confuse yourself.

Rough data cleaning and prep

```
es/gapminder/code:
08564
13:42 .
 2011 ..
10:33 .Rhistory
13:10 bryan-a01-01-dataPrep.R
 2010 bryan-a01-02-dataMerge.R
13:43 bryan-a01-03-dataExplore.R
 2010 bryan-a01-04-fillContinentData.R
 2010 bryan-a01-05-everyFiveYears.R
19:40 bryan-a01-10-baseGraphicsStepByStep.R
 2010 bryan-a01-11-baseGraphicsPlotGapminderOneYear.R
19:43 bryan-a01-12-baseGraphicsSoln.R
13:42 bryan-a01-15-latticeStepByStep.R
21:19 bryan-a01-16-latticePlotGapminderOneYear.R
21:17 bryan-a01-17-latticeSoln.R
21:33 bryan-a01-18-latticeDemos.R
 2010 bryan-a01-19-latticeStruggles.R
19:44 bryan-a01-30-makeGapminderColorScheme.R
13:17 bryan-a01-40-dataAggregation.R
19:47 bryan-a01-50-basicColorDemo.R
 2010 bryan-a01-99-scratchpad.R
```

"Getting to know you" figures & tables; diagnostics

Final data cleaning

Actual workhorse files; making figures

Key functions; creating general resources

Code snippets you aren't currently using

Good stackoverflow for further reading:

ESS workflow for R project/package development

Workflow for statistical analysis and report writing

How does software development compare with statistical programming/analysis?

Suggestions for statistical computing workflow [closed]

Bottom line: most projects break down *at least* into:

import

clean

analyze

Break your R code down accordingly.

Have some system and BE CONSISTENT.

stackoverflow Questions Tags Users Badges Unanswered

Suggestions for statistical computing workflow [closed]

3

Note: I chose to ask this here instead of at stats.stackexchange.com because it is about software workflow tools and not about any particular methods. I felt that people more intimately familiar with the actual software packages would be able to help more, because I'm specifically trying to avoid the common answer I get from academics, which is to just always use R or Matlab and then make grad students figure out how to make stuff work for large data.

My motivation for how I teach this course is *exactly this*:

to save a bunch of diverse grad students from figuring how to "make this stuff work" ...

(or worse, to save them from NOT ever figuring it out or figuring it out reeeeaaaalllly slowly).

Low-tech documentation of an analysis

DMSO/RA Analysis			
Task	Input	Code	Output
Read in CEL files and pre-process the data.	CEL files	(used default settings for RMA in Bioconductor package affy version 1.6.7)	dmsoRARaw.txt
Read in pre-processed data and address probeset naming issues.	dmsoRARaw.txt	dmsoRANameFix.r	dmsoRAData.txt dmsoRAProbeNames.txt
Fit a one-way ANOVA model to each probeset.	dmsoRAData.txt	dmsoRAFit.r	dmsoRAmlmOutput.robj
Visualization of the estimated DMSO and RA effects and the ESC signature change.	(figures for paper used Bioconductor package 'hexbin') (this code uses Bioconductor package 'geneplotter') dmsoRAmlmOutput.robj grFunScatter.r dmsoRAFilterFun.r	dmsoRAScatterPlot.r	dmsoRAsmsc.pdf dmsoRAsmscZoom.pdf dmsoRAsmscZoomFilt2COL.pdf
Conduct the bootstrap -- EXPOSITORY TOY EXAMPLE FOR INTERACTIVE USE	dmsoRAmlmOutput.robj	dmsoRABootstrapToy.r	dmsoRArandomResidsToy.txt dmsoRAbootResultsToy.txt
Conduct the bootstrap -- THE FULL ANALYSIS	dmsoRAmlmOutput.robj	(meant to be executed in BATCH mode) dmsoRABootstrap.r (shell script that calls above repeatedly) dmsoRABootstrap.sh	dmsoRAbootManager.txt (files not posted, due to size and the fact that results vary due to randomization) dmsoRArandomResids.txt dmsoRAbootResults.txt
Apply (three) definition(s) of the ESC change signature to obtain, for each probeset, a confidence value (CV).	dmsoRAbootResults.txt dmsoRAfilterFun.r	dmsoRACV.r	dmsoRAcvBasis.txt dmsoRAavgCV.txt

Low-tech documentation of an analysis

jenny › Groups › STAT545A

Notebook

View | Edit | Revisions

bryan-a01

jenny
2:35pm Mon Sep 20

Highlight changes

Here is the code JB used to prepare the Gapminder data, to implement her solutions, and to demonstrate various things with the Gapminder data. The inputs and outputs are also linked from here, most notably all figures.

Click here if you just want to browse JB's Gapminder project directory. Keep reading for a guided tour.

Data preparation. This is how JB took data from Gapminder and prepared it for processing with R.

- Input
  - *(Indirectly, three Excel spreadsheets downloaded from the Gapminder website in 2009. Local copies I work from now are gapdata003.xls (population), life-expectancy-reference-spreadsheet-20090204-xls-format.xls (life expectancy), and gapdata001-1.xlsx (GDP per capita). See comments in code for how I used those files to generate plain text, delimited inputs.)*
  - data/totalPop.txt
  - data/lifeExpect.txt
  - data/gdpPercap.txt
- Code
  - code/bryan-a01-01-dataPrep.R
- Output
  - data/totalPopClean.txt
  - data/lifeExpectClean.txt
  - data/gdpClean.txt

around here we transitioned to topics that are also relevant to homework 6

http://www.stat.ubc.ca/~jenny/STAT545A/hw06_puttingAllTogether.html

and we visited some of the resources linked to at the bottom of that assignment

project organization / literate programming /
reproducible research

collaboration / open science

The Trifecta of Vexing Issues in Scientific Statistical Computing

version control / back up / archive

project organization / literate programming /
reproducible research

Sweave
knitr

What the cool kids are
doing ....

collaboration / open science

github
Rforge
sourceforge

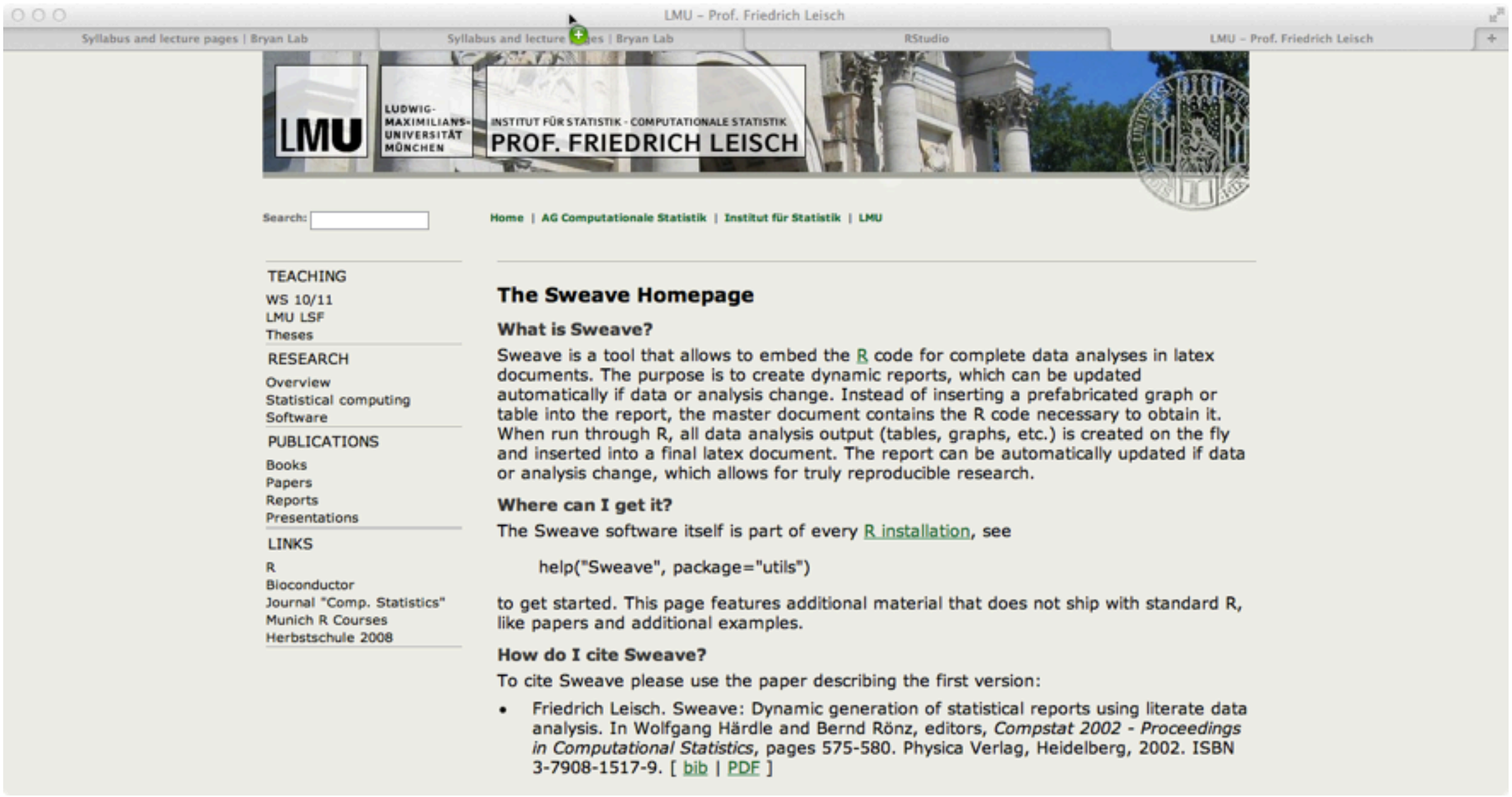
The Trifecta of Vexing Issues in
Scientific Statistical Computing

git
subversion
mercurial

version control / back up / archive

project organization / literate programming / reproducible research

Sweave

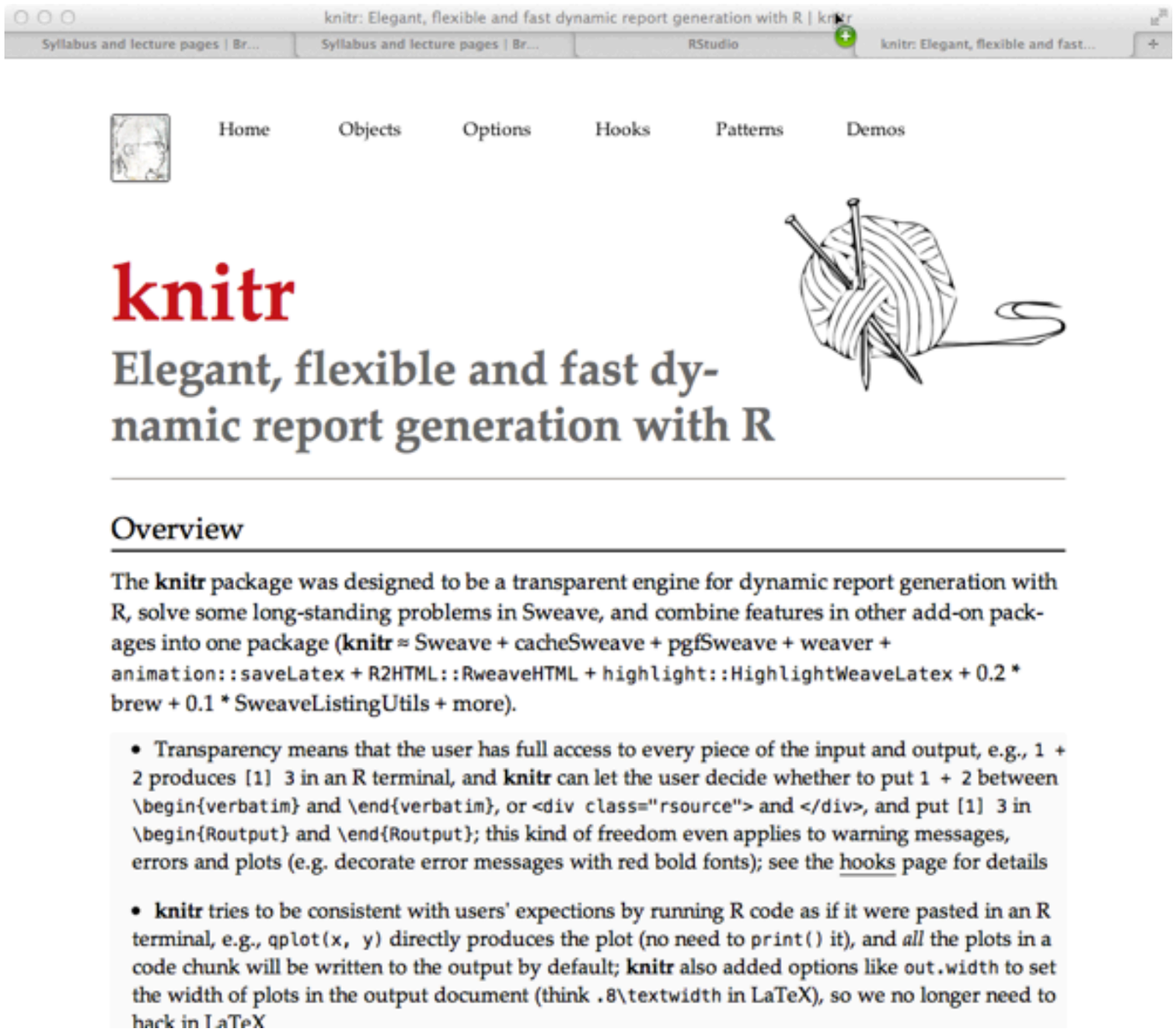


"Sweave is a tool that allows to embed the R code for complete data analyses in latex documents. The purpose is to create dynamic reports, which can be updated automatically if data or analysis change. Instead of inserting a prefabricated graph or table into the report, the master document contains the R code necessary to obtain it. When run through R, all data analysis output (tables, graphs, etc.) is created on the fly and inserted into a final latex document. The report can be automatically updated if data or analysis change, which allows for truly reproducible research."

from http://www.stat.uni-muenchen.de/~leisch/Sweave/

project organization / literate programming / reproducible research

knitr



The **knitr** package was designed to be a transparent engine for dynamic report generation with R, solve some long-standing problems in Sweave, and combine features in other add-on packages into one package (**knitr** ≈ Sweave + cacheSweave + pgfSweave + weaver + `animation::saveLatex` + `R2HTML::RweaveHTML` + `highlight::HighlightWeaveLatex` + 0.2 * brew + 0.1 * SweaveListingUtils + more).

Getting Started with R Markdown, knitr, and Rstudio 0.96

Syllabus and lecture pages | Br... | Syllabus and lecture pages | Br... | RStudio | Getting Started with R Markdo...

Jeromy Anglim's Blog: Psychology and Statistics

| HOME | SITE MAP | R | RESEARCH | ABOUT | SUBSCRIBE |

THURSDAY, MAY 17, 2012

Getting Started with R Markdown, knitr, and Rstudio 0.96

This post examines the features of R Markdown using knitr in Rstudio 0.96. This combination of tools provides an exciting improvement in usability for reproducible analysis. Specifically, this post (1) discusses getting started with R Markdown and `knitr` in Rstudio 0.96; (2) provides a basic example of producing console output and plots using R Markdown; (3) highlights several code chunk options such as caching and controlling how input and output is displayed; (4) demonstrates use of standard Markdown notation as well as the extended features of formulas and tables; and (5) discusses the implications of R Markdown. This post was produced with R Markdown. The source code is available here as a gist. The post may be most useful if the source code and displayed post are viewed side by side. In some instances, I include a copy of the R Markdown in the displayed HTML, but most of the t
are reading the source and post side by side.

Getting started

To work with R Markdown, if necessary:

- Install R
- Install the lastest version of RStudio (at time of pos
- Install the latest version of the `knitr` package:

```
install.packages("knitr")
```

To run the basic working example that produced this blo

- Open R Studio, and go to File - New - R Markdown
- If necessary install `ggplot2` and `lattice` packages:

```
install.packages("ggplot2"); install.packages(
```

- Paste in the contents of the gist (which contains th

I'm an academic bridging I/O psychology and statistics. My blog contains 100+ posts focused on data analysis in the social sciences. If you're new, check out the Site Map. If you love R, check out the 40+ posts on R. If you want to follow the blog, see the RSS

Search This Blog

Search

FeedBurner FeedCount

901 readers

Subscribe via RSS

Posts

Comments

Categories

ability academia APAStyle Article

Example of using R Markdown — Gist

https gist.github.com/2716336

Syllabus and lecture pages | Br... | Syllabus and lecture pages | Br... | RStudio

github:gist

jennybc | Dashboard

New Gist | My Gists | Starred Gists | All Gist

gist: 2716336 download fork star

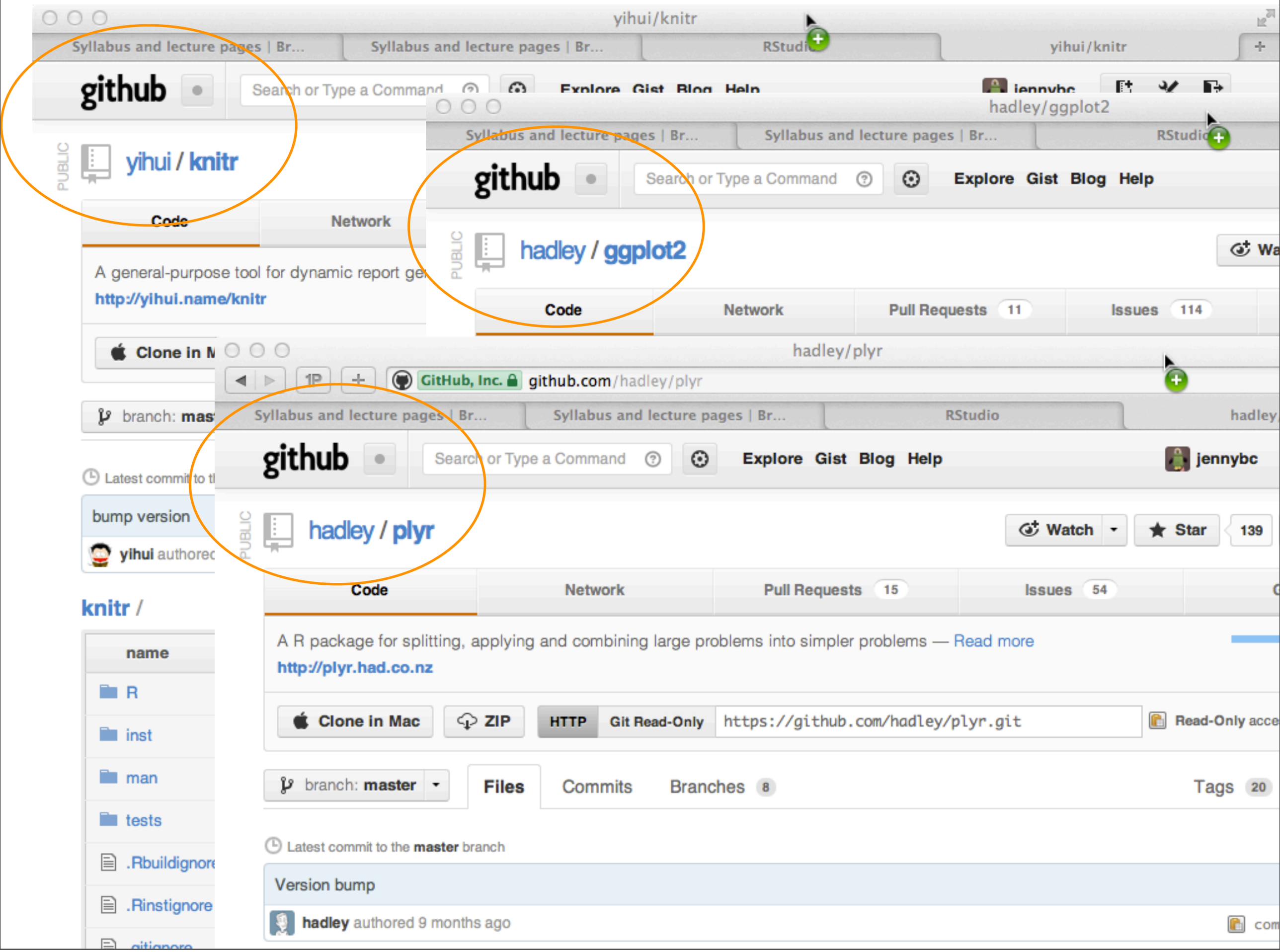
Description: Example of using R Markdown

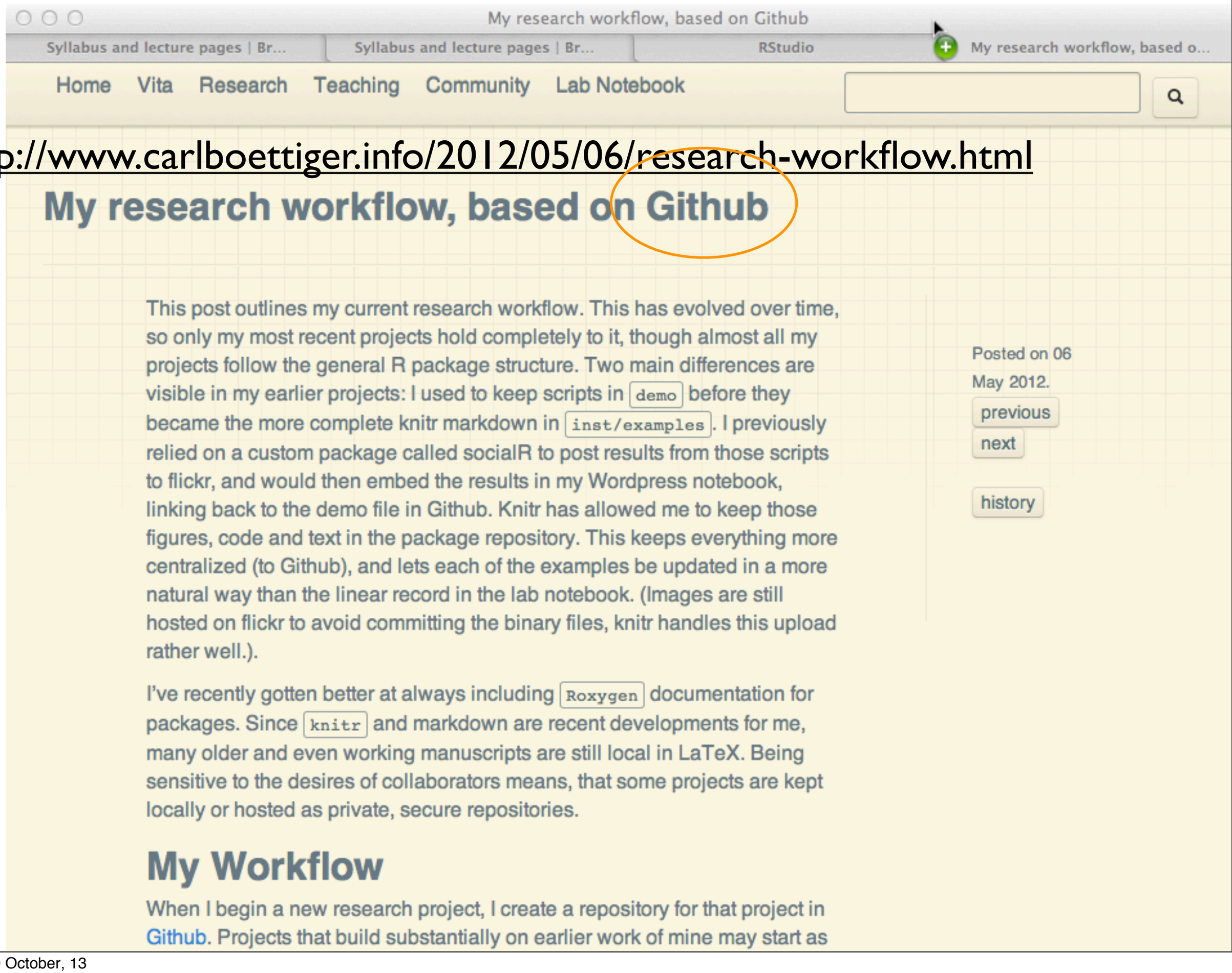
Public Clone URL: git://gist.github.com/2716336.git

Embed All Files: show embed

example-r-markdown.rmd # embed raw

```
1 This post examines the features of [R Markdown](http://www.rstudio.org/docs/author:
2 using [knitr](http://yihui.name/knitr/) in Rstudio 0.96.
3 This combination of tools provides an exciting improvement in usability for
```


yihui/knitr
Syllabus and lecture pages | Br...
Syllabus and lecture pages | Br...
RStudio
yihui/knitr
github
Search or Type a Command
Explore Gist Blog Help
jennybc
PUBLIC
yihui / knitr
Code
Network
A general-purpose tool for dynamic report ge
http://yihui.name/knitr
Clone in M
branch: mas
Latest commit to t
bump version
yihui authored
knitr /
name
R
inst
man
tests
.Rbuildignore
.Rinstignore
hadley/ggplot2
Syllabus and lecture pages | Br...
Syllabus and lecture pages | Br...
RStudio
github
Search or Type a Command
Explore Gist Blog Help
PUBLIC
hadley / ggplot2
Code
Network
Pull Requests 11
Issues 114
hadley/plyr
GitHub, Inc.
github.com/hadley/plyr
Syllabus and lecture pages | Br...
Syllabus and lecture pages | Br...
RStudio
hadley
github
Search or Type a Command
Explore Gist Blog Help
jennybc
PUBLIC
hadley / plyr
Watch
Star
139
Code
Network
Pull Requests 15
Issues 54
A R package for splitting, applying and combining large problems into simpler problems — Read more
http://plyr.had.co.nz
Clone in Mac
ZIP
HTTP
Git Read-Only
https://github.com/hadley/plyr.git
Read-Only acce
branch: master
Files
Commits
Branches 8
Tags 20
Latest commit to the master branch
Version bump
hadley authored 9 months ago


My research workflow, based on Github
Syllabus and lecture pages | Br...
Syllabus and lecture pages | Br...
RStudio
My research workflow, based o...
Home
Vita
Research
Teaching
Community
Lab Notebook
http://www.carlboettiger.info/2012/05/06/research-workflow.html
My research workflow, based on Github
This post outlines my current research workflow. This has evolved over time, so only my most recent projects hold completely to it, though almost all my projects follow the general R package structure. Two main differences are visible in my earlier projects: I used to keep scripts in demo before they became the more complete knitr markdown in inst/examples. I previously relied on a custom package called socialR to post results from those scripts to flickr, and would then embed the results in my Wordpress notebook, linking back to the demo file in Github. Knitr has allowed me to keep those figures, code and text in the package repository. This keeps everything more centralized (to Github), and lets each of the examples be updated in a more natural way than the linear record in the lab notebook. (Images are still hosted on flickr to avoid committing the binary files, knitr handles this upload rather well.).
I've recently gotten better at always including Roxygen documentation for packages. Since knitr and markdown are recent developments for me, many older and even working manuscripts are still local in LaTeX. Being sensitive to the desires of collaborators means, that some projects are kept locally or hosted as private, secure repositories.
Posted on 06 May 2012.
previous
next
history
My Workflow
When I begin a new research project, I create a repository for that project in Github. Projects that build substantially on earlier work of mine may start as

http://support.rstudio.org/help/kb/advanced/using-version-control-with-rstudio

RStudio

rstudio.org/docs/version_control/overview

Reader

Syllabus and lecture pages | Bryan Lab | Syllabus and lecture pages | Bryan Lab | RStudio

Home Screenshots Download Docs Support Development Blog

Using Version Control with RStudio

Overview

Version control is an indispensable tool for coordinating the work of teams and also has many benefits for individual work. The following StackOverflow discussions describe some of these benefits:

- Why should I use version control?
- R and version control for the solo data analyst

RStudio includes integrated support for two open source version control systems:

- Git
- Subversion

To use version control with RStudio, you should first ensure that you have installed Git and/or Subversion (details below). Next, you should become familiar with using RStudio Projects (which are required for version control features to be enabled).

Installation

Prior to using RStudio's version control features you will need to ensure that you have Git and/or Subversion installed on your system. The following describes how to do this for various platforms.

Git

Prior to using Git with RStudio you should install it using the appropriate method for your platform:

- **Windows:** http://code.google.com/p/msysgit/
- **OSX:** http://code.google.com/p/git-osx-installer/
- **Debian/Ubuntu:** `sudo apt-get install git-core`
- **Fedora/RedHat:** `sudo yum install git-core`

An excellent resource for learning more about Git and how to use it is the Pro Git online book. Another good resource for learning about git is the Introduction to Git provided by GitHub.

project organization / literate programming /
reproducible research

Sweave
knitr

What the cool kids are
doing ....

collaboration / open science

github
Rforge
sourceforge

The Trifecta of Vexing Issues in
Scientific Statistical Computing

git
subversion
mercurial

version control / back up / archive

Bottom line: do something deliberate that has a good hassle: result ratio for you.

Be open to upgrading your approach as time goes on.

Keep your eyes and ears open re: developments in this area!

Your R life, in general

- I keep a file that records everything about my current and, sometimes, older R installations
- Top of `/Users/jenny/resources/R/code/2012-04-setup.R`:

```
### installed binary of ...
### R version 2.15.0 (2012-03-30)
### Copyright (C) 2012 The R Foundation for Statistical Computing
### ISBN 3-900051-07-0
### Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

## no longer needed ... set in .Rprofile
## options(CRAN = "http://cran.stat.sfu.ca/")

install.packages(pkgs = "RColorBrewer")
install.packages(pkgs = "car")
### installed dependencies MASS nnet survival

install.packages(pkgs = "R2HTML")

install.packages(pkgs = "latticeExtra")
### I note this installed the dependency 'lattice'
### huh? I guess my lattice was out of date?

install.packages(pkgs = "hexbin")
```

I always install packages from here, with a line of R code. Easy to see what packages I use, get back up and running after a re-install, notes-to-self about glitches, etc.

Your R life, in general

- I update R on an 'as needed' basis (probably should do more often ...); my setup file makes it easy to get back in business quickly because all add-ons are documented there
- I generally only re-install packages upon the first occasion of needing them, so I don't accumulate a huge number of packages I used once, like, back in 2006
- In my setup file, I have a comment line reading "nothing below here re-installed yet" and, as I re-install packages, I move the associated lines of code and updated comments up above this line

Your R life, in general

- You can set up certain things you want for every R session at startup in ~/.Rprofile
  - A stackoverflow thread entitled “Expert R users, what's in your .Rprofile?”

My current .Rprofile

```
cat("\n  Get some real work done, Jenny!\n\n")

## add lattice to the default packages, set a CRAN mirror
oldPkgs <- getOption("defaultPackages")
oldRepos <- getOption("repos")
oldRepos["CRAN"] <- "http://cran.stat.sfu.ca/"
## "http://cran.cnr.Berkeley.edu"
options(defaultPackages = c(oldPkgs, "lattice", "roxygen2"),
        repos = oldRepos)

## source all JB-written helper / handy functions
foo <- list.files("~/resources/R/code/jHandy", full.names = TRUE)
foo <- foo[-grep(".R~", foo)]         # omit backup files
for(i in foo) {
  cat("\n sourcing ", i, "\n")
  source(i)
}

## reduce my problems with str'ed objects line wrapping in an
## unattractive way
options(str = list(strict.width = "cut",
          digits.d = 3, vec.len = 4),
        devtools.path = "~/resources/R/librarySandbox")

if (interactive()) {
  suppressMessages(require(devtools))
}

## there was a period when I also included
## 'device = "quartz"' here,
## but that currently isn't necessary

lattice::lattice.options(default.theme = jTheme)
```

this is out of date but I'll leave in case it's helpful to see?

Directory listing of jHandy

```
/Users/jenny/resources/R/code/jHandy:
total used in directory 80 available 272603144
drwxr-xr-x  12 jenny  staff   408 May 16 15:50 .
drwxr-xr-x  21 jenny  staff   714 Oct  2 15:33 ..
-rw-r--r--   1 jenny  staff   157 Mar 22  2010 jExtract.R
-rw-r--r--   1 jenny  staff  1045 Jun  9  2011 jFactor.R
-rw-r--r--   1 jenny  staff    45 Sep 15  2009 jPaste.R
-rw-r--r--   1 jenny  staff   447 Jan 28  2011 jSubset.R
-rw-r--r--   1 jenny  staff  2157 May 16 15:50 jTheme.R
-rw-r--r--   1 jenny  staff  2089 May 16 15:47 jTheme.R~
-rw-r--r--   1 jenny  staff   209 Oct 28  2011 jWriteTable.R
-rw-r--r--   1 jenny  staff   209 Oct 28  2011 jWriteTable.R~
-rw-r--r--   1 jenny  staff   209 Mar  5  2009 peek.R
-rw-r--r--   1 jenny  staff   214 Mar  5  2009 refactor.R
```

Full disclosure: one should probably convert personal functions that are used throughout your code into a proper R package .....

package management

a “library” is where R stores its packages

for years, I never messed with or questioned the defaults ... a fine strategy for new users

at some point you may want to get fancier

Link to the R Installation and Administration Manual,
section 6 Add-on packages
http://cran.r-project.org/doc/manuals/R-admin.html#Add_002don-packages

Helpful documentation written for Johns Hopkins Biostat system re: "Creating a personal R package library"
http://www.biostat.jhsph.edu/bit/R-personal-library.html

How to manage multiple package locations (folders) in R?
http://stackoverflow.com/questions/7993061/how-to-manage-multiple-package-locations-folders-in-r

the default library on my system:
/Library/Frameworks/R.framework/Versions/2.15/Resources/library

I keep two other libraries within my own user filespace

[1] for packages I download from CRAN
/Users/jenny/resources/R/libraryCRAN

[2] for packages I am developing
/Users/jenny/resources/R/libraryDev

To notify R about this I created a .Renviron file in my home directory that contains this:
R_LIBS=~/resources/R/libraryCRAN:~/resources/R/libraryDev

library situation in a fresh default R installation (on Mac OS)

```
> R.home(component = "home")
[1] "/Library/Frameworks/R.framework/Resources"

> .Library
[1] "/Library/Frameworks/R.framework/Resources/library"

> .libPaths()
[1] "/Library/Frameworks/R.framework/Versions/2.15/Resources/library"
```

library situation for JB today

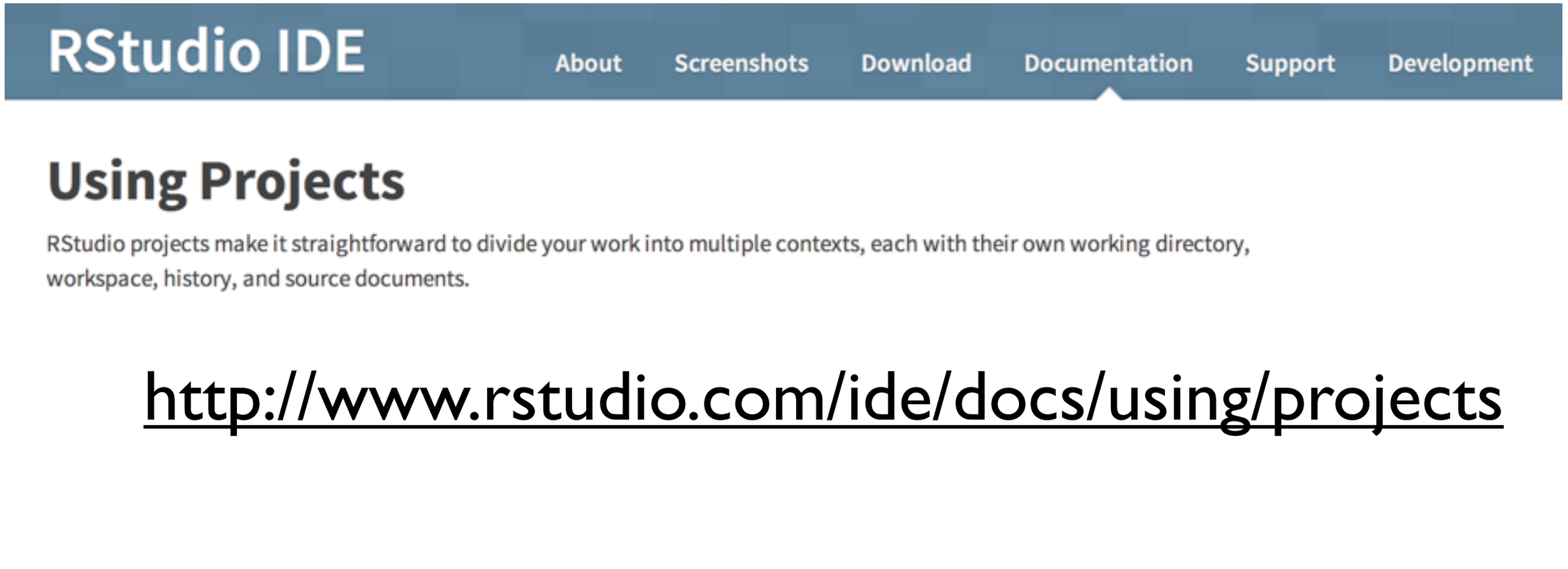
```
> R.home(component = "home")
[1] "/Library/Frameworks/R.framework/Resources"

> .Library
[1] "/Library/Frameworks/R.framework/Resources/library"

> .libPaths()
[1] "/Users/jenny/resources/R/libraryCRAN"
[2] "/Users/jenny/resources/R/libraryDev"
[3] "/Library/Frameworks/R.framework/Versions/2.15/Resources/library"
```

deep thoughts and pretty pictures to shape your data analytical mindset ....

revisiting some promises and themes from first day of class

A place for everything and everything in its place

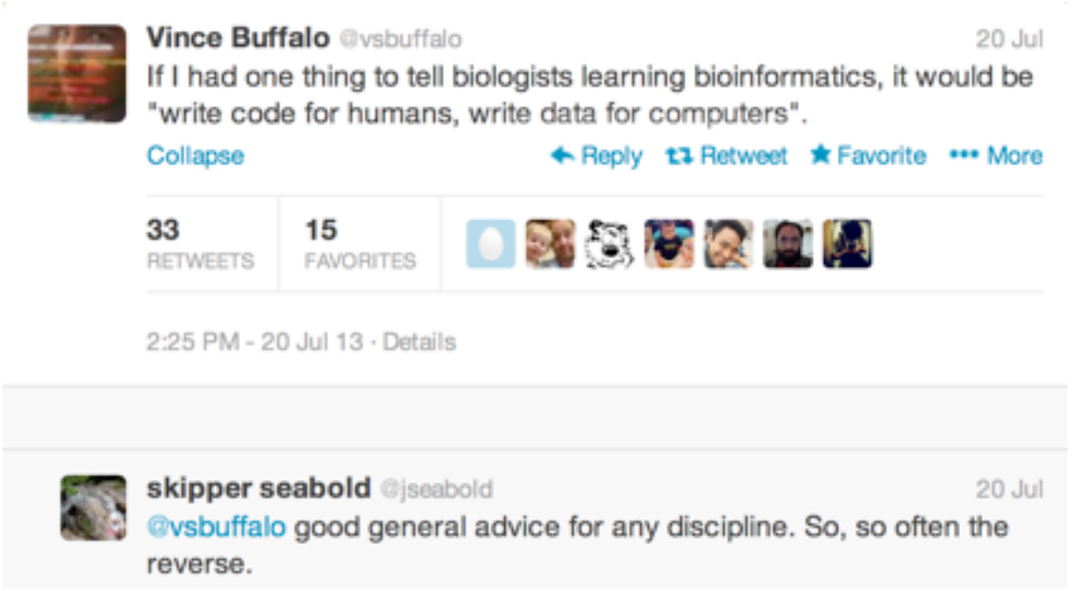
«



http://www.rstudio.com/ide/docs/using/projects

Use RStudio Projects!

write code for humans, write data for computers



> Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.
>
> **Donald Knuth**

Nine simple ways to make it easier to (re)use your data

Ethan P. White, Elita Baldridge, Zachary T. Brym, Kenneth J. Locey, Daniel J. McGlinn, and Sarah R. Supp

Our nine recommendations are:

this section is especially good reading

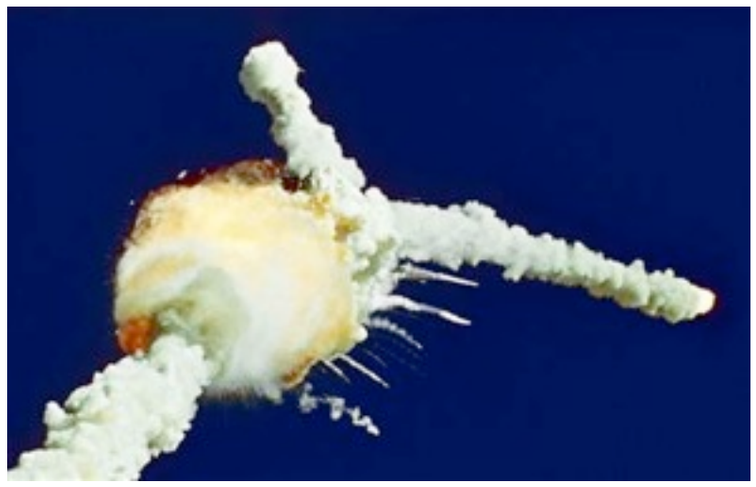
1. Share your data
2. Provide metadata
3. Provide an unprocessed form of the data
4. Use standard data formats (including file formats, table structures, and cell contents)
5. Use good null values
6. Make it easy to combine your data with other datasets
7. Perform basic quality control
8. Use an established repository
9. Use an established and liberal license

White et al. (2013) Nine simple ways to make it easier to (re)use your data. *PeerJ PrePrints* 1:e7v2 http://dx.doi.org/10.7287/peerj.preprints.7v2

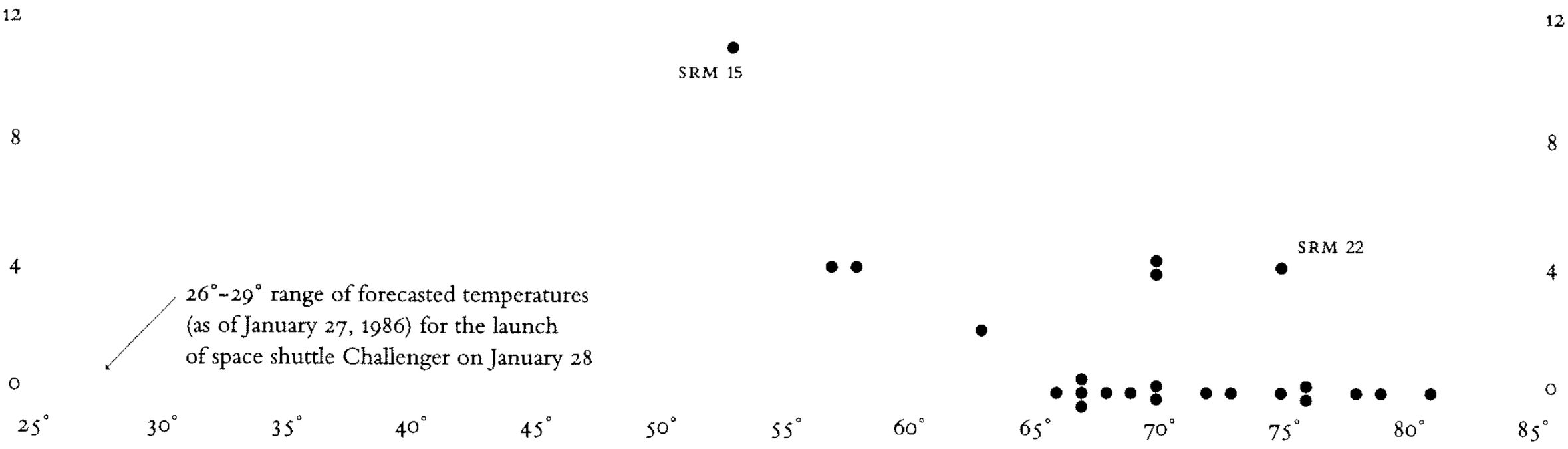
reshape your data

as in real life, it has a tendency to get short and fat, when you'd really prefer tall and skinny

“A picture is worth a thousand words”



O-ring damage
index, each launch



source is real

PERFECT MATCH: TRIFLE WITH MOSCATO

AT A GLANCE

SERVES 20 PEOPLE

1 HR PREPARATION
50 MIN COOKING (PLUS COOLING, SETTING)

You'll need

1.5 kg	blackberries or mulberries, plus extra to serve (see note)
300 gm	caster sugar
2	vanilla beans, split and seeds scraped
10	gelatine leaves (titanium strength), softened in cold water for 5 minutes
300 ml	pink moscato
1	lemon, juice only
330 ml	crème de mûre (see note)
1.25 kg	crème fraiche
150 ml	milk, or enough to thin
2	lemons, finely grated rind only
40 gm	(¼ cup) pure icing sugar, sifted
	Sponge
8	eggs, at room temperature
250 gm	raw caster sugar
250 gm	plain flour, sieved
50 gm	butter, melted and cooled

Method

1. For sponge, preheat oven to 175C. Whisk eggs and sugar in an electric mixer until tripled in volume (7 minutes). Fold through flour in batches, fold in butter, pour into a 28cm-square cake tin lined with baking paper. Bake until golden and centre springs back when pressed (20-25 minutes). Cool in tin, turn out, halve sponge horizontally, trim each half to fit a 6 litre-capacity glass bowl, then remove from bowl and set aside, reserving trimmings.

2. Meanwhile, combine 1kg berries, sugar, 1 vanilla bean and seeds and 1.1 litres water in a large saucepan, simmer over low heat until infused (50 minutes). Strain through a fine sieve (discard solids), transfer 1 litre hot liquid to a bowl (reserve remainder). Squeeze excess water from gelatine, add to bowl, stir to dissolve. Add moscato, lemon juice and 80ml crème de mûre. Strain half into trifle bowl, scatter over 250gm berries and refrigerate until set (2-2½ hours). Chill remaining berry jelly, removing from refrigerator if it starts to set.

3. Reduce 250ml remaining liquid (discard excess) over high heat to 50ml or until syrupy (10-15 minutes), refrigerate until required.

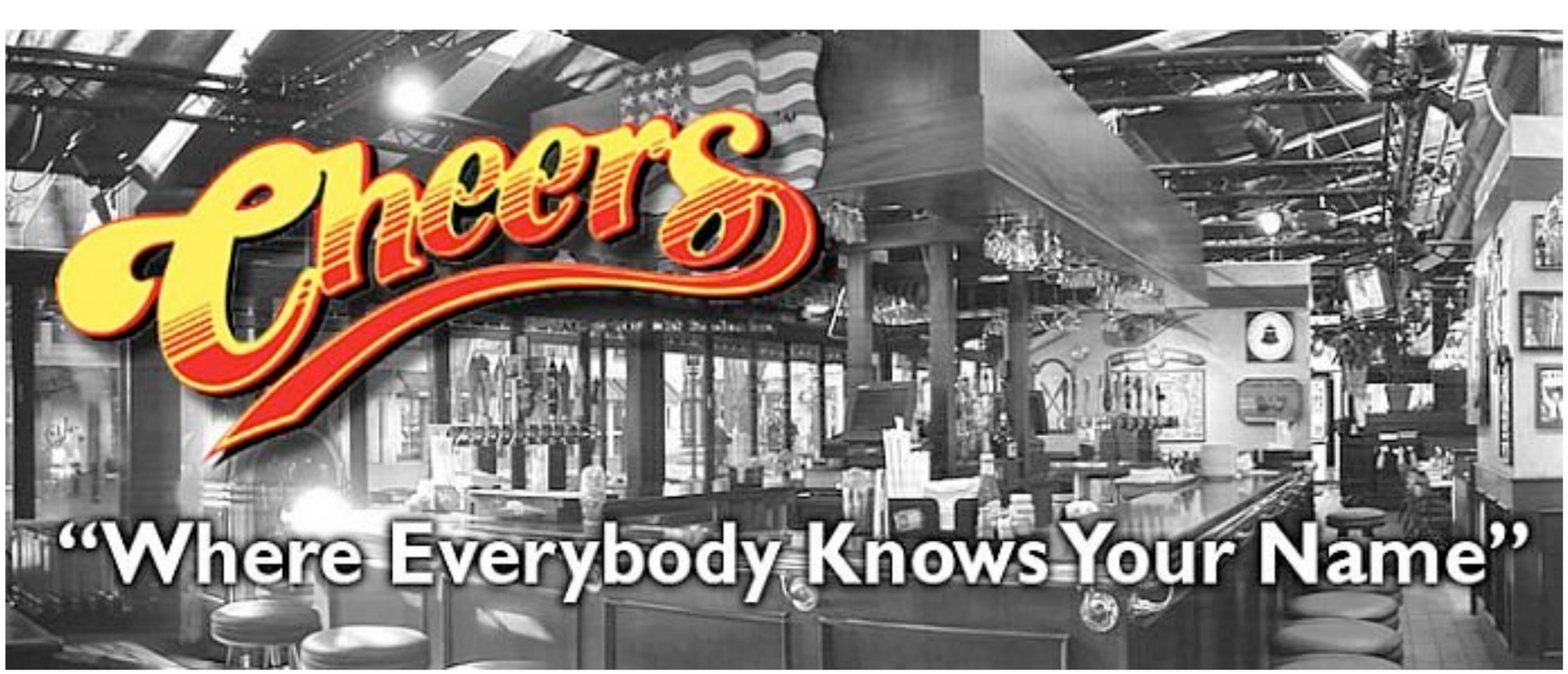
4. Meanwhile, combine crème fraiche, milk, rind, icing sugar and remaining vanilla seeds in a bowl, adding extra milk if necessary until spreadable. Spread one-third over set jelly, top with a sponge round, fill any gaps with trimmings, drizzle with 125ml crème de mûre. Scatter over remaining berries, pour over remaining jelly (mixture should be starting to set). Refrigerate until set (2-2½ hours). Top with half the remaining crème fraiche mixture, then remaining sponge. Drizzle with remaining crème de mûre, top with remaining crème fraiche mixture. Cover, refrigerate overnight. Serve scattered with extra berries and drizzled with blackberry syrup.

source is real

"**The source code is real**. The objects are realizations of the source code. Source for EVERY user modified object is placed in a particular directory or directories, for later editing and retrieval."

-- from the Emacs Speaks Statistics (ESS) manual

Names matter



ain’t nothing like the real thing

minimize the creation of excerpts and copies of your data ... it will just confuse you later

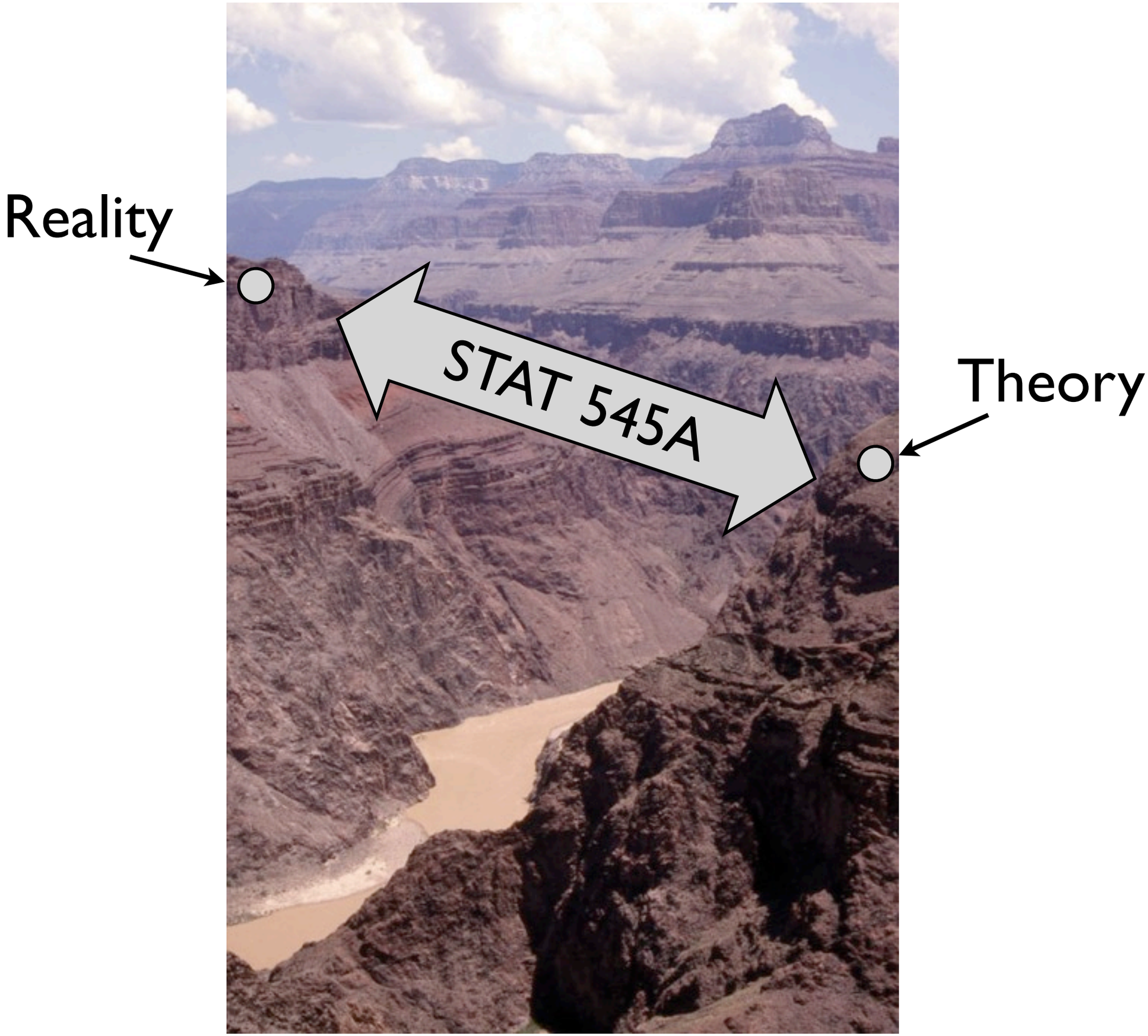
weak links in the chain: process, packaging and presentation

Two inter-related goals

- Foster your development of a personal philosophy for data analysis, esp. exploratory and descriptive analysis.
- Help you assemble a modern toolchain and workflows for data analysis.

My hope:

You'll leave this course with (at least the beginnings of) a confident, deliberate attitude about how to approach data analysis and the practical skills to put your attitude into action.


Reality
STAT 545A
Theory