

**STAT 545A**

**Class meeting #9**

**Wednesday, October 3, 2012**

**Dr. Jennifer (Jenny) Bryan**

**Department of Statistics and Michael Smith Laboratories**



# Review of last class

lattice graphics: to get good results, you only need to learn basic commands and arguments

To get great results, you need to be courageous and specify lots of seemingly arcane arguments, redefine lists of graphics parameters, and redefine panel functions. BUT it's not as hard as it seems! Walk before you run. But get moving.

Typical workflow: “so so” plot using built-in facilities. Gradually take control of what you need to change by first doing nothing (i.e. specifying the graphics parameters or the panel function but specify the default!), then baby step up to the full glorious figure you want.

# Review of last class

Another fruitful approach: find a plot in Sarkar's or Murrell's book (or in this class), go get the code that produced it and slowly “substitute” your data and desires into the original

the “type” argument is extremely useful!

`show.settings()` and `trellis.par.get()` are useful for grasping what default colors, symbols, etc. are, so you can take control and substitute your own colors, symbols, etc.

high-volume scatterplots benefit from some fancier treatment: hexagonal binning, 2-dimensional density estimation

For next Wednesday:

I'd like you to start your page for the final project. At the very least, give a brief description of what you plan to do. Link to a data source. Pose some questions you might try to answer or some issues you hope to explore.

Ideally you will be making even more progress, i.e. working on data acquisition, import, cleaning.

Now I will continue my efforts to turn you all  
into professional R scripters .....

“Habits of highly effective programmers”

**“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

# Jenny's slight expansion ...

Actually, there are a few other things that are real, besides source code.

Input data: perfectly preserved file of data as it came from its “producer” (maybe revoke write permission?)

Clean data: plain text, delimited, clean data file that you created from the mess you got above (maybe revoke write permission?)

Figures: lots of them, with meaningful names, stored to file(s) using a command (not the mouse)

Important statistical results: stored to file with a command in the *plainest* form possible, i.e. plain text if feasible or as R objects otherwise

# The other philosophy

R objects are real. Figures you see popping up on your screen are real. The R code you typed at the command line late last night is real.

R objects are created by typing at the command line and are changed using `fix()` or are recreated. R workspaces are saved and reloaded. Etc.

I **cannot responsibly recommend** this approach.

If you're in this class, you're the kind of person for whom this approach is not robust.

# Why “source code is real”?

- “Objects are real” comes more naturally to those accustomed to a GUI, to Excel, etc.
- “Source is real” places emphasis on the logic of your analysis, not on the specific numerical result obtained by applying it to a dataset.
- “Source is real” leaves us in a much better position to replicate the analysis -- perhaps with the same data, perhaps not -- or to use it as the starting point for a new analysis.
- “Source is real” approach has a built-in mechanism for documenting exactly what was done.
- Jenny’s modification to the philosophy acknowledges that we don’t always want to go back and rerun everything.

# “Source code is real”: how to implement

- Lowest-tech: Open a text editor and an R session. Write R code in editor, copy to clipboard, paste into R. And/or compose commands at the R prompt and copy the “keepers” into a .R file in the editor.
- Much more pleasant and sustainable: Use a smart editor or IDE that can send lines (or other logical chunks) of R code to a live R session.

To be clear: An R transcript is NOT the same as R code.

```
> x=read.delim(<...>)
> y=x$lifeExp
> z=x$gdpPercap
> plot(y,z)
```

```
x <- read.delim(<...>)
plot(lifeExp ~ gdpPercap, x)
```

# 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

# Why can't I just “wing it”?

- Good practices make your analytical work
  - Less prone to error
  - Quicker and more pleasant
  - Mastery of these skills gives you the psychic energy to do the job right, i.e. not get lazy
  - Easier for you to use, maintain, improve
  - Easier for others to read, use, modify
- No better forum or time to make a serious investment in your analytical “set-up”. Just Do It!

# R style, standards, habits

- Google's R style guide (and the discussion in the R community); Hadley Wickham's adaptation
- R Coding Conventions
- Gelman blogged about R style and about the divisive issue of underscores versus dots (read the comments)
- Keynote talk by Martin Machler from useR 2004
- Hadley Wickham's rubric for marking R code
- Chapter 2 in S Poetry
- Karl Broman's Coding practices
- Conference report “Good Programming Practices in Healthcare Creating Robust Programs” (mostly about SAS but the rules on pages 3 and 4 are really good)

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.

# Coding conventions

- Trust me -- certain practices will make your coding life much more pleasant
- Use indenting
- Use spaces around binary operators and after commas
- Wrap your lines
- Use comments (and indent properly)
- Develop naming conventions for yourself
- Lots of this is automatic or very very easy with the right editor/IDE setup, e.g. Emacs Speaks Statistics or Rstudio ... they didn't build this stuff in for jollies, people! It's useful.

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 ...  
## $ 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 3 3 ...  
## $ 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          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 (iAlwaysDoltLikeThis)

```
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 ...  
## $ 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 3 3 ...  
## $ 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 ☺

## Homework before next class:

Read *at least two* of the documents suggested about R style (or locate and suggest others!).

Develop a modest goal for partial implementation of “good” R style and start trying to achieve that.

Write a short blurb or review (a couple sentences is OK) about each piece you read, describing it or critiquing it or recounting how easy/hard/valuable/useless implementation seems to be. Is it about nitty-gritty code style or is it more about an approach programming? Is it fun or boring to read? You get the idea.

**Read this bit on your own!**

Sidebar: How to comment, reflect on an activity or reading

Taken from introductory lecture from UBC CS Prof Tamara Munzner for CPSC 533 Information Visualization. which, by the way, looks like a cool course.

View as a chance to demonstrate what you've learned and/or the thoughtfulness with which you approached the task/reading

Also an opportunity to share information with your fellow students

## Questions

- questions or comments
- fine to be less formal than written report
  - correct grammar and spelling expected nevertheless
  - be concise: a few sentences good, one paragraph max!
- should be thoughtful, show you've read and reflected
  - poor to ask something trivial to look up
  - ok to ask for clarification of genuinely confusing section

## Question Examples: Poor

- Well, what exactly Pad++ is? Is it a progarmming library or a set of API or a programming language? how can we use it in our systems, for xample may be programming in TCL or OpenGL may be ?
- I learned some from this paper and got some ideas of my project.

## Question Examples: OK

- This seems like something fun to play around with, are there any real implementations of this? Has a good application for this type of zooming been found? Is there still a real need for this now that scroll wheels have become prevalent and most people don't even use the scroll bar anymore?
- Playing with the applet, I find I like half of their approach. It's nice to zoom out as my scroll speed increases, but then I don't like the automatic zoom in when I stop scrolling. Searching the overview I found the location I wanted, but while I paused and looked at the overview, I fell back in to the closeup. I think they need to significantly dampen their curve.

## Question Examples: Good

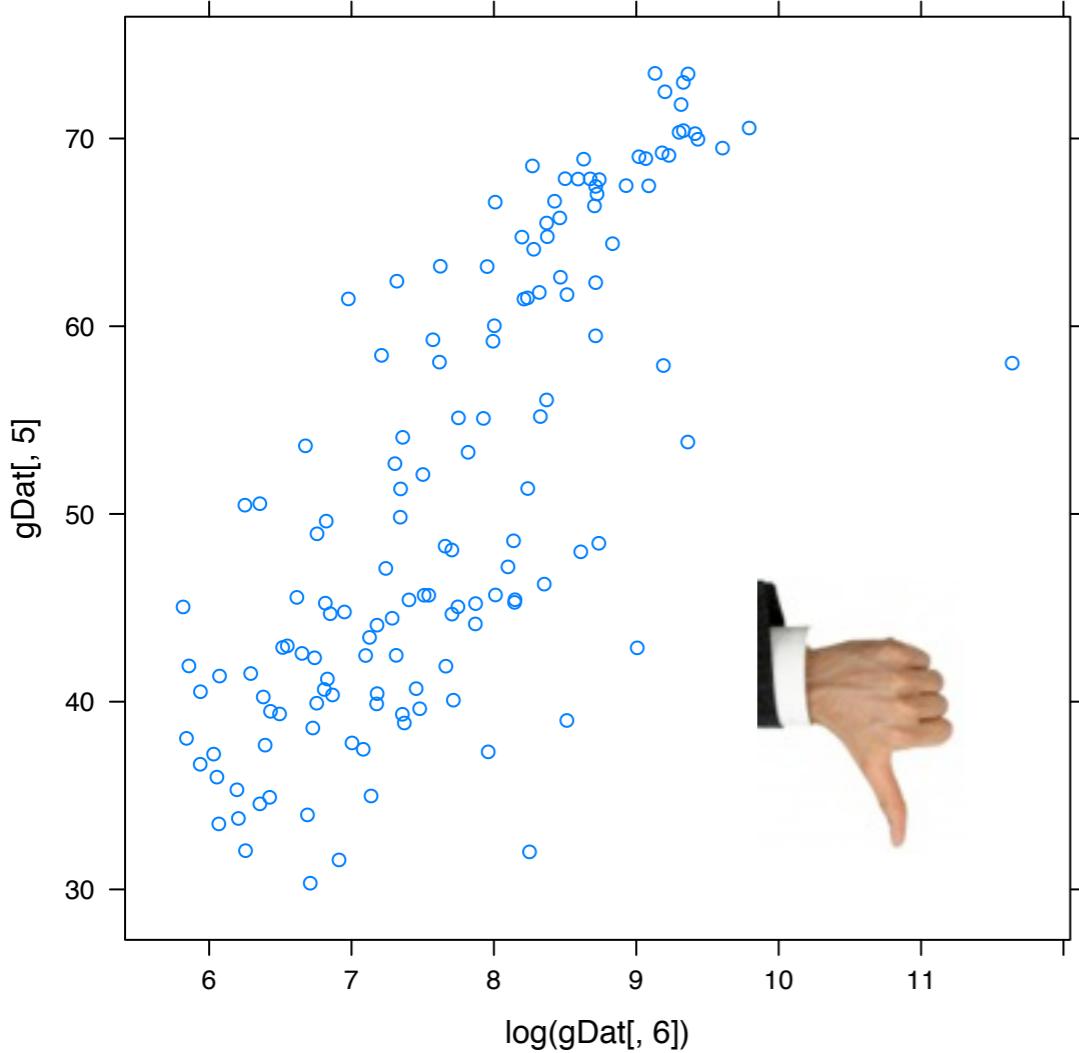
- It would be interesting to compare the approach in this paper to some other less-mathematically-thought-out zoom and pan solutions to see if it is really better. Sometimes "faking it" is perceived to be just as good (or better) by users.
- The space-scale diagrams provided a clear intuition of why zooming out, panning then zooming in is a superior navigation technique. However, I found the diagram too cumbersome for practical use, especially for objects with zoom-dependent representations (Figure 11).

## Question Examples: Great

- I'm curious as to what would have happened if the authors had simply preselected the values of the free parameters for the participants in their user study, and then had the users compare their technique to the standard magnification tools present in a 'normal' application (much like the space-scale folks did). Could it be that the users are 'manufacturing' a large standard deviation in the free parameter specifications by settling for values that merely produce a local improvement in their ability to manipulate the interface, instead of actively searching for an optimal valuation scheme?
- In a related vein, the speed-dependent automatic zooming met with mixed success on some applications. Isn't this success related to how "compressible" some information is? i.e. because zooming must necessarily throw out some information, it isn't obvious which information to keep around to preserve the navigable structure.

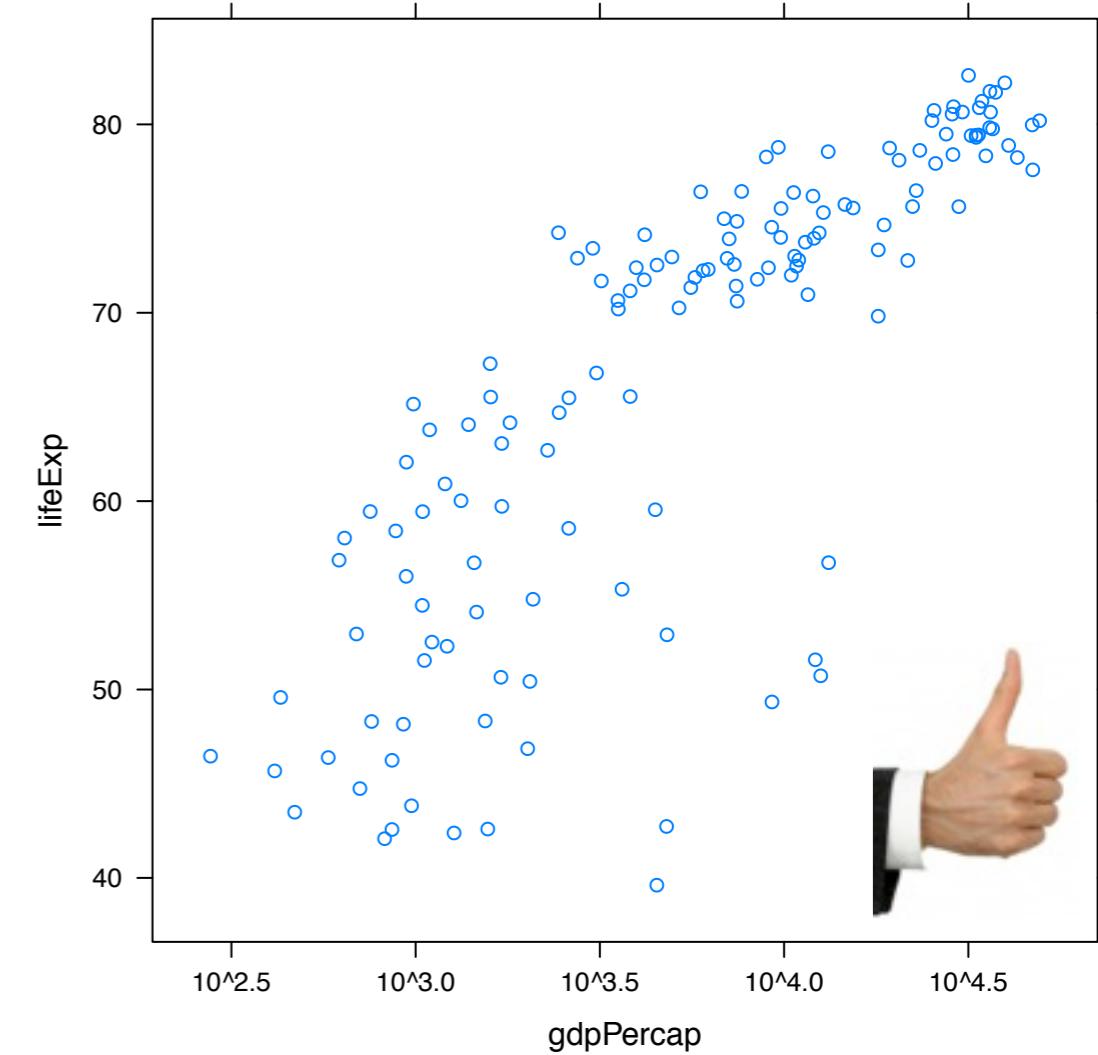
End of sidebar: How to comment, reflect on an activity or reading

```
xyplot(gDat[,5]~log(gDat[,6]),subset=gDat[,2]==1957)
```



Which figure and code would you rather try to decipher late at night?

Give good names, use them, use spaces, .....



```
xyplot(lifeExp ~ gdpPerCap, gDat,  
       subset = year == 1957,  
       scales = list(x = list(log = 10)))
```

# Don't use Magic Numbers

According to Wikipedia, Magic Numbers are “unique values with unexplained meaning or multiple occurrences which could (preferably) be replaced with named constants”

Why do we avoid them?

To make code more transparent.

To make code easier to maintain.

To make code more reusable.

## Two kinds of Magic Numbers and how to handle properly:

- constant(s) that can be **derived**, e.g. the number of observations in the data matrix or the row corresponding to “Afghanistan” in 1962 or the range of gdpPercap. Fix: derive them once transparently in a prominent place (top of script?) and store as well-named variable.
- constant(s) that need to be **set**, e.g. the size of a plotting symbol. Fix: set them once transparently in a prominent place (top of script?) and store as well-named variable.

```

> ## creating a matrix, so I can demo apply
> (jCountries <- sort(c('Canada', 'United States', 'Mexico'))))
[1] "Canada"           "Mexico"            "United States"

> tinyDat <- subset(gDat, country %in% jCountries)

> (nY <- length(unique(tinyDat$year)))      # 12 years
[1] 12

> jLifeExp <- matrix(tinyDat$lifeExp, nrow = nY)

> colnames(jLifeExp) <- jCountries

> rownames(jLifeExp) <- tinyDat$year[1:nY]

> jLifeExp
    Canada Mexico United States
1952 68.750 50.789       68.440
1957 69.960 55.190       69.490
1962 71.300 58.299       70.210
1967 72.130 60.110       70.760
1972 72.880 62.361       71.340
1977 74.210 65.032       73.380
1982 75.760 67.405       74.650
1987 76.860 69.498       75.020
1992 77.950 71.455       76.090
1997 78.610 73.670       76.810
2002 79.770 74.902       77.310
2007 80.653 76.195       78.242

```

nY is a constant that can be derived; learn important constants from the data, give them good names, use them downstream

```

## demo with just one year
(jYear <- max(gDatOrdered$year))

## the most basic plot
xyplot(lifeExp ~ gdpPercap, gDatOrdered, subset = year == jYear)

## take control of the axis limits
(i <- i + 1)
jXlim <- c(200, 58000)
jYlim <- c(21, 88)
xyplot(lifeExp ~ gdpPercap, gDatOrdered,
       subset = year == jYear,
       xlab = jXlab, ylab = jYlab,
       xlim = jXlim, ylim = jYlim)

## shade the plot symbols according to country BUT now use our color
## scheme (and change the plotting symbol to something fillable and
## make the symbol outline transparent)
## enact the color using the group argument
jDarkGray <- 'grey20'
jPch <- 21
jGapminderPars <-
  list(superpose.symbol = list(pch = jPch, col = jDarkGray, cex = 2,
                                fill = countryColors$color[match(levels(gDatOrdered$country),
                                countryColors$country)]))

(i <- i + 1)
xyplot(lifeExp ~ gdpPercap, gDatOrdered, subset = year == jYear,
       xlab = jXlab, ylab = jYlab,
       scales = list(x = list(log = 10)),
       xlim = jXlim, ylim = jYlim,
       xscale.components = xscale.components.log10,
       type = c("p", "g"),
       group = country,
       par.settings = jGapminderPars)

```

jYear, jXlim, jYlim, jPch, etc. are constants that need to be set. BUT it's easy to imagine using different values. So set them once, stored as a well-named variable, and it will be easy to change later.

**set.seed() is the proper way to make something random AND repeatable.**

```
> set.seed(1903)
> (countriesToKeep <- as.character(sample(levels(gDat$country),
+                                         size = nC)))
[1] "Norway"      "Chad"        "Eritrea"      "Germany"      "Costa Rica"
[6] "Jamaica"     "Cuba"        "Nepal"

> jYear <- 2007

> jDat <-
+   droplevels(subset(gDat, country %in% countriesToKeep & year == jYear))

> jDat <- jDat[order(jDat$gdpPerCap),]

> str(jDat)
'data.frame': 8 obs. of 6 variables:
 $ country : Factor w/ 8 levels "Chad","Costa Rica",...: 4 7 1 6 3 2 5 8
 $ year    : int  2007 2007 2007 2007 2007 2007 2007 2007
 $ pop     : num  4906585 28901790 10238807 2780132 11416987 ...
 $ continent: Factor w/ 4 levels "Africa","Americas",...: 1 3 1 2 2 2 4 4
 $ lifeExp  : num  58 63.8 50.7 72.6 78.3 ...
 $ gdpPerCap: num  641 1091 1704 7321 8948 ...

> jDat
    country year    pop continent lifeExp gdpPerCap
504    Eritrea 2007 4906585    Africa  58.040  641.3695
1080    Nepal  2007 28901790    Asia   63.785 1091.3598
276     Chad  2007 10238807    Africa  50.651 1704.0637
792    Jamaica 2007 2780132  Americas 72.567 7320.8803
396     Cuba  2007 11416987  Americas 78.273 8948.1029
360  Costa Rica 2007 4133884  Americas 78.782 9645.0614
576  Germany 2007 82400996    Europe  79.406 32170.3744
1152  Norway 2007 4627926    Europe  80.196 49357.1902
```

# How I organize my work

Contents of /Users/jenny/research/conibear

The screenshot shows a Mac OS X Finder window titled "conibear". The sidebar on the left lists "DEVICES" (Macintosh HD, iDisk), "SHARED", and "PLACES" (STAT545A, vanPel, stoepel, haggarty, pheNorm, plosOne, CV, jenny, Applications, Downloads, baetz, conibear, hartman, peterson). The main pane displays a file list for the "conibear" folder. The columns are "Name", "Date Modified", "Size", and "Kind". The "Name" column lists various subfolders: assortedProjects, easyEpistasis, invertase, preprocessingAndMisoingSchluter, quenneville, schluter, and webSupp. The "Date Modified" column shows dates ranging from 3-Jul-08 to 13-Aug-08. The "Size" column shows all items as "--". The "Kind" column shows all items as "Folder". A status bar at the bottom indicates "20 items, 96.64 GB available".

Name	Date Modified	Size	Kind
assortedProjects	13-Aug-08, 1:25 PM	--	Folder
easyEpistasis	3-Jul-08, 1:11 PM	--	Folder
invertase	13-Aug-08, 1:38 PM	--	Folder
code	13-Aug-08, 3:48 PM	--	Folder
data	13-Aug-08, 2:22 PM	--	Folder
figs	13-Aug-08, 2:29 PM	--	Folder
preprocessingAndMisoingSchluter	13-Aug-08, 1:17 PM	--	Folder
code	13-Aug-08, 1:23 PM	--	Folder
data	13-Aug-08, 1:26 PM	--	Folder
figs	13-Aug-08, 2:23 PM	--	Folder
prose	10-Apr-08, 10:01 AM	--	Folder
results	10-Apr-08, 12:28 PM	--	Folder
quenneville	28-May-08, 9:13 AM	--	Folder
schluter	13-Aug-08, 1:14 PM	--	Folder
code	11-Apr-08, 1:27 PM	--	Folder
data	25-Jul-08, 11:03 AM	--	Folder
figs	13-Aug-08, 1:17 PM	--	Folder
prose	9-Apr-08, 4:33 PM	--	Folder
results	10-Apr-08, 1:08 PM	--	Folder
webSupp	9-Apr-08, 3:09 PM	--	Folder

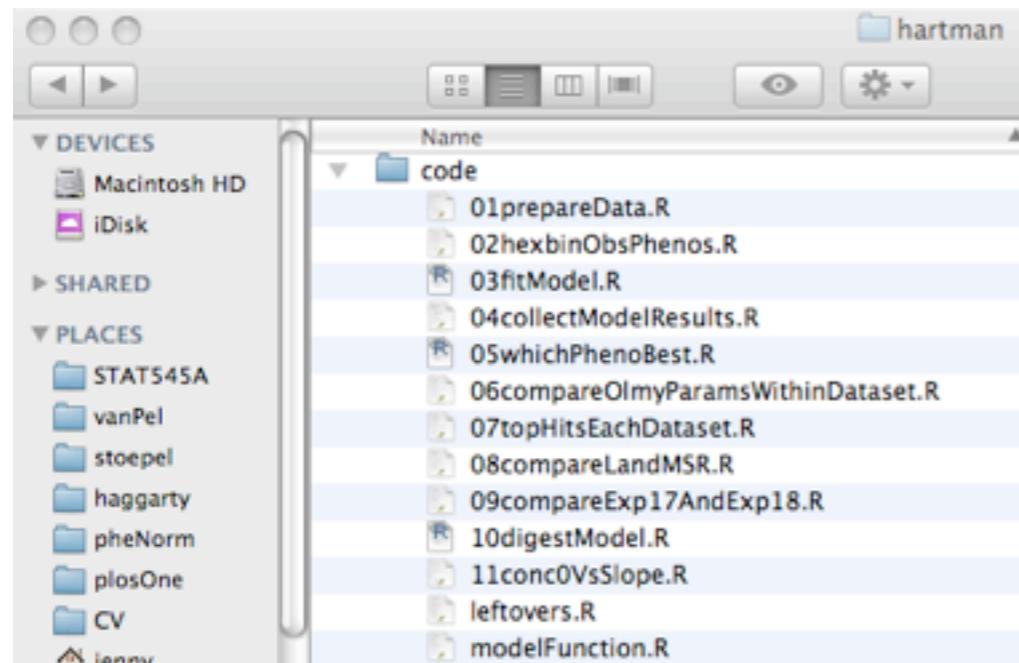
# How I organize my work

- 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)

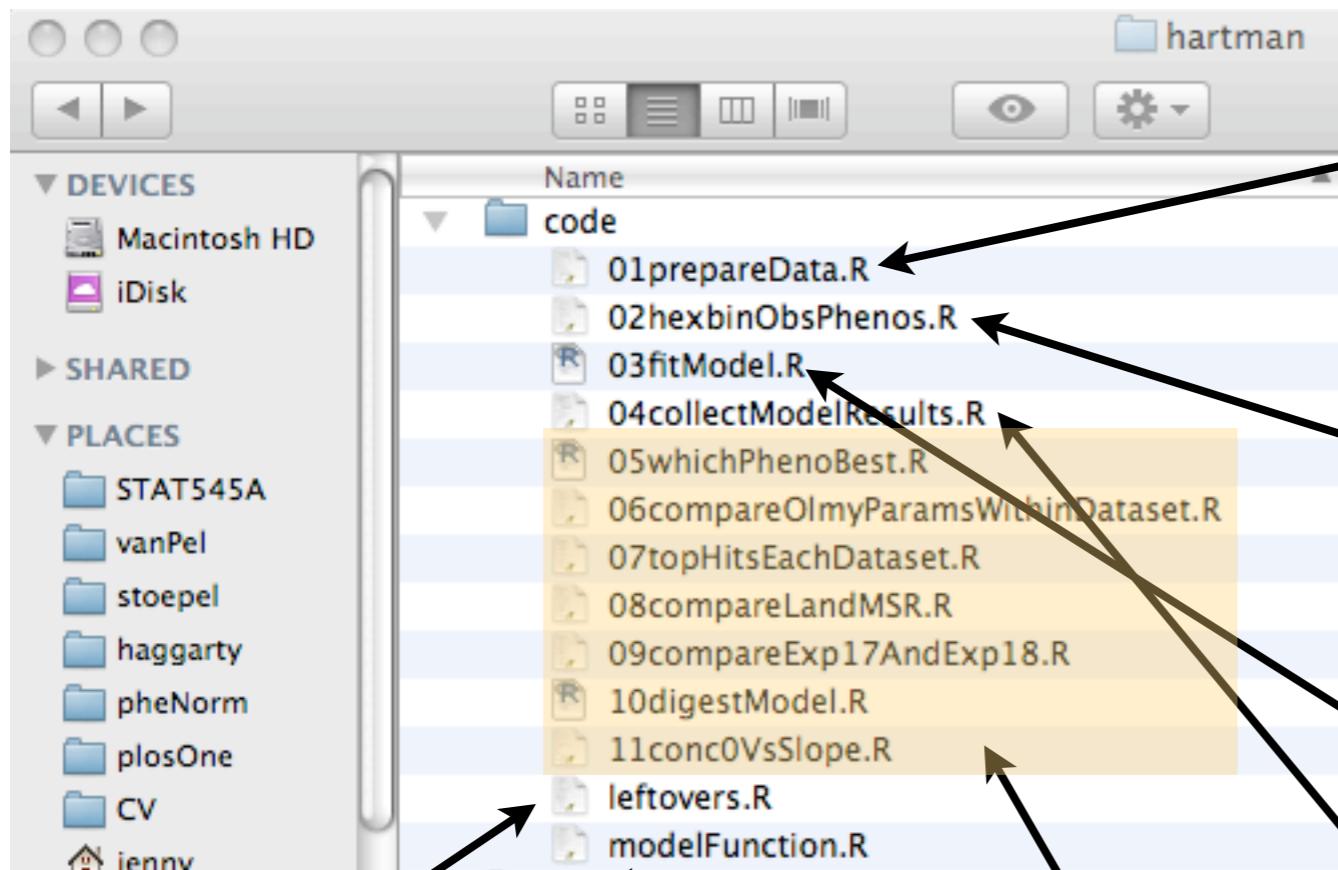
# the code subdirectory



```
/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



Code snippets you aren't  
currently using

Key functions

Data cleaning and prep

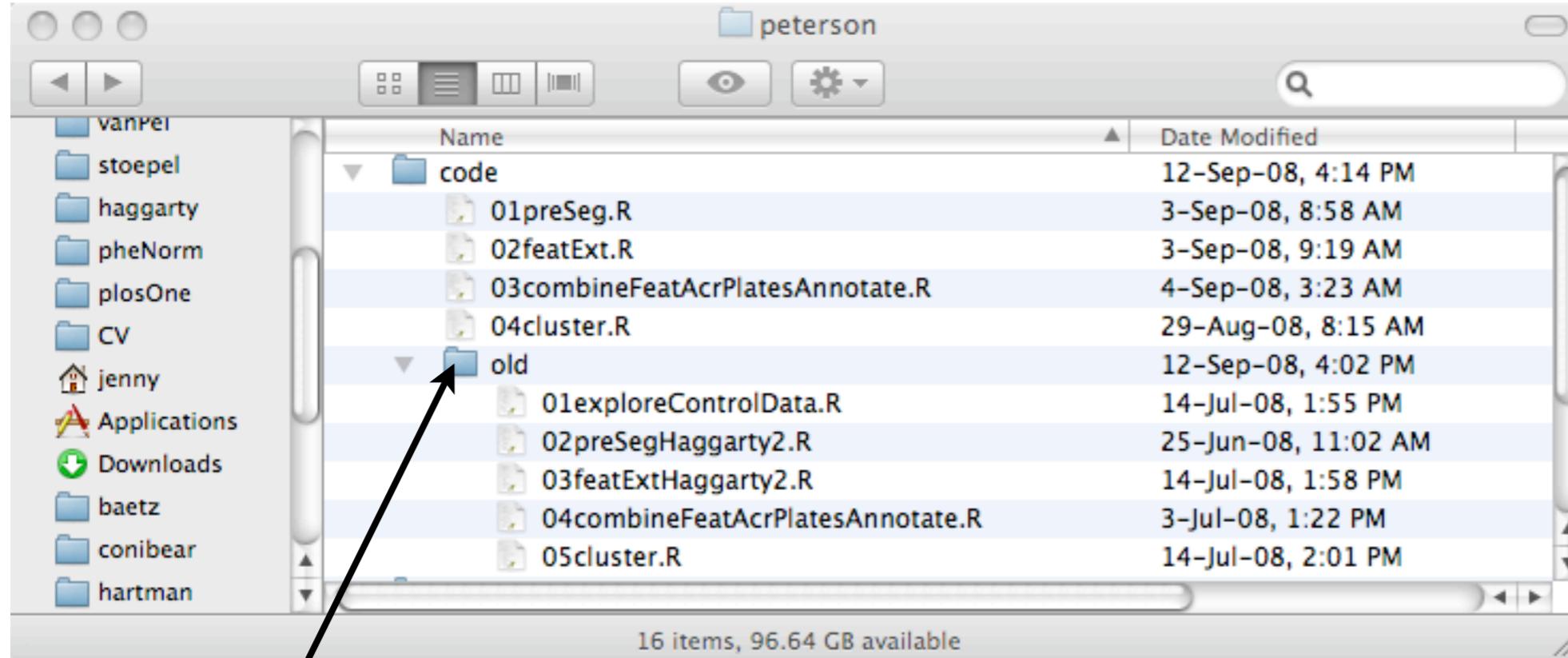
“Getting to know you”  
figures

First real analysis

Getting analytical results  
into a useful form

Analyzing the analytical  
results, including making  
more figures

# 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.

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

# Low-tech documentation of an analysis



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](#)

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

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  
knitr

How JB is currently leaning ....

collaboration / open science

github  
Rforge  
sourceforge

## The Trifecta of Vexing Issues in Scientific Statistical Computing

git

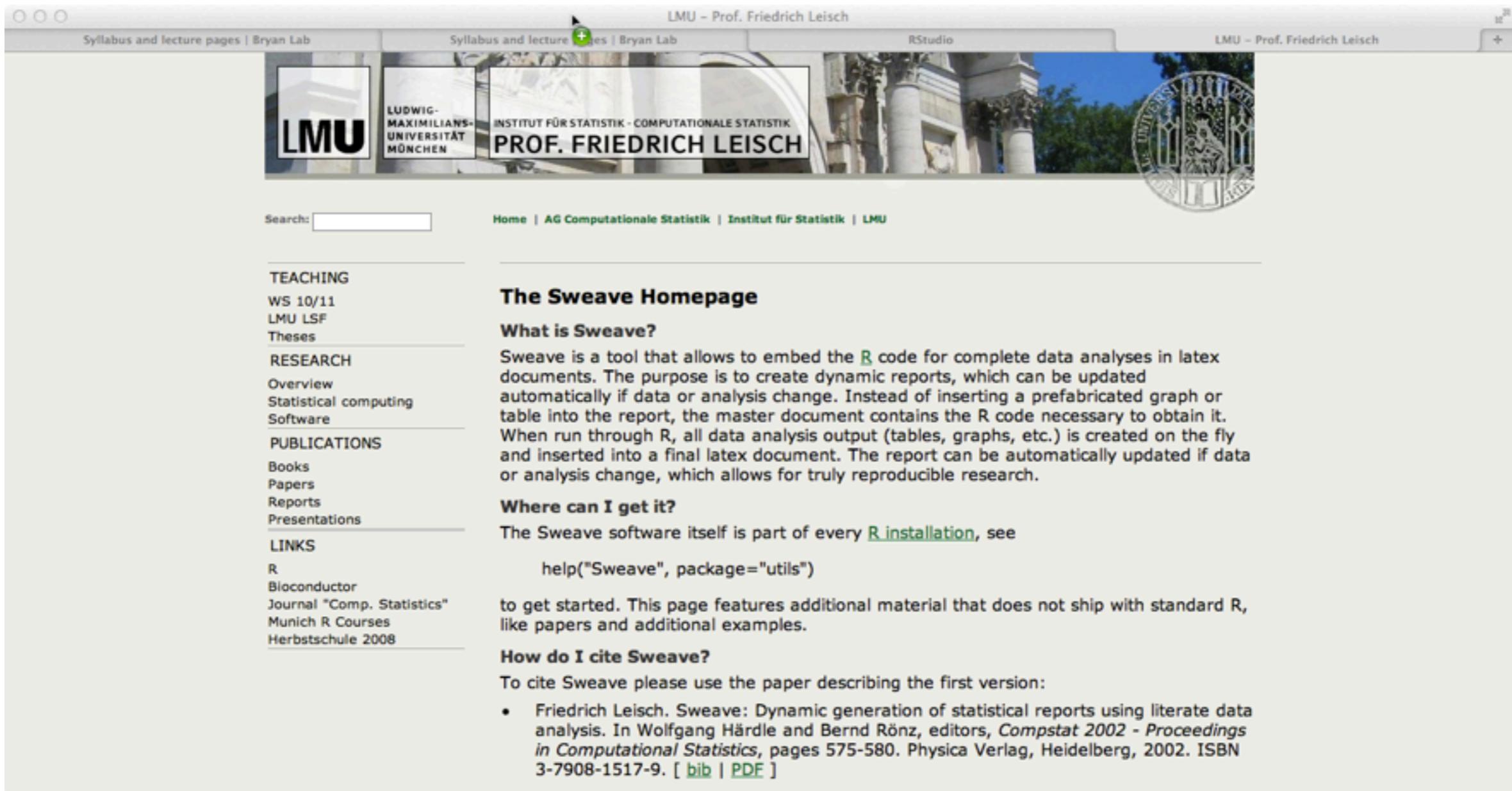
subversion

mercurial

version control / back up / archive

# project organization / literate programming / reproducible research

## Sweave



The screenshot shows a web browser window with four tabs open:

- Syllabus and lecture pages | Bryan Lab
- Syllabus and lecture + | Bryan Lab
- RStudio
- LMU – Prof. Friedrich Leisch

The main content area displays the "The Sweave Homepage". It features the LMU logo and a banner with the text "INSTITUT FÜR STATISTIK - COMPUTATIONALE STATISTIK" and "PROF. FRIEDRICH LEISCH". Below the banner is a search bar and a navigation menu with links to "Home", "AG Computationale Statistik", "Institut für Statistik", and "LMU".

**The Sweave Homepage**

**What is 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.

**Where can I get it?**

The Sweave software itself is part of every `R` installation, see

```
help("Sweave", package="utils")
```

to get started. This page features additional material that does not ship with standard R, like papers and additional examples.

**How do I cite Sweave?**

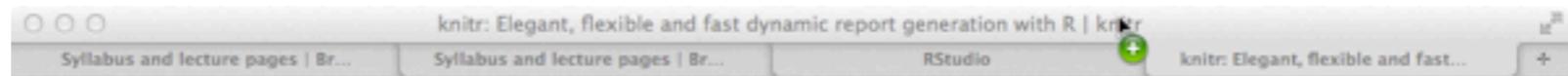
To cite Sweave please use the paper describing the first version:

- Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 - Proceedings in Computational Statistics*, pages 575-580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9. [ [bib](#) | [PDF](#) ]

“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.”

# project organization / literate programming / reproducible research

## knitr



Home

Objects

Options

Hooks

Patterns

Demos

# knitr

Elegant, flexible and fast  
dynamic report generation with R



## Overview

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`).

- Transparency means that the user has full access to every piece of the input and output, e.g., `1 + 2` produces [1] 3 in an R terminal, and `knitr` can let the user decide whether to put `1 + 2` between `\begin{verbatim}` and `\end{verbatim}`, or `<div class="rsource">` and `</div>`, and put [1] 3 in `\begin{Routput}` and `\end{Routput}`; this kind of freedom even applies to warning messages, errors and plots (e.g. decorate error messages with red bold fonts); see the [hooks](#) page for details
- `knitr` tries to be consistent with users' expectations by running R code as if it were pasted in an R terminal, e.g., `qplot(x, y)` directly produces the plot (no need to `print()` it), and *all* the plots in a code chunk will be written to the output by default; `knitr` also added options like `out.width` to set the width of plots in the output document (think `.8\textwidth` in LaTeX), so we no longer need to `hack in LaTeX`

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**  $\approx$  **Sweave** + **cacheSweave** + **pgfSweave** + **weaver** + **animation**::**saveLatex** + **R2HTML**::**RweaveHTML** + **highlight**::**HighlightWeaveLatex** + 0.2 \* **brew** + 0.1 \* **SweaveListingUtils** + more).

# 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 time I assume you 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 posting, this is 0.96)
- Install the latest version of the knitr package:

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

To run the basic working example that produced this blog post:

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

```
install.packages("ggplot2"); install.packages("lattice")
```
- Paste in the contents of [the gist](#) (which contains the R Markdown file

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](#) and email subscription options.

[Overview of Blog Content](#)

[Academic Publications](#)

[Teaching Resources](#)

[Google+ Profile: JeromyAnglim](#)

[Twitter @JeromyAnglim](#)

[GitHub @JeromyAnglim](#)

[Follow on](#)

[Search This Blog](#)

[FeedBurner FeedCount](#)

901 readers  
BY FEEDBURNER

[Subscribe via RSS](#)

[Posts](#)   
[Comments](#)

[Categories](#)

ability academia APAStyle Article  
Deconstruction Australia basic  
analyses bayesian Beamer  
BibTeX binary variable  
blogging book review  
bootstrapping calculus  
causation CFA cluster analysis  
computers correlation data  
mining data sharing descriptive  
statistics design difference  
scores discriminant function  
analysis dyads Eclipse Endnote  
Excel experiments factor  
analysis focus groups formatting  
fun GEE general advice ggplot2  
I/O Psych Inquisit internet  
interviews introduction JabRef  
... [LaTeX](#)

The screenshot shows the RStudio interface with two main windows. On the left, the 'example.Rmd' file is open in the code editor. The code contains R Markdown syntax, including headers, lists, and code blocks. On the right, the 'Preview HTML' window shows the rendered output of the Markdown code, displaying the formatted text, lists, and code blocks as they would appear in a web browser.

R Markdown enables easy authoring of reproducible web reports from R. It offers:

- Easy creation of web reports from R that can be automatically regenerated whenever underlying code or data changes.
- A highly accessible syntax (markdown) which lower the barriers to entry for reproducible research.
- Output of a standalone HTML file (with images embedded directly in the file) that is easy to share using email, Dropbox, or by deploying to a web server.
- Support for publishing dynamic and interactive web content.

This article includes an overview of how to use R Markdown within RStudio. For more specific details on syntax and implementation, see the [R Markdown](#) article.

## Markdown Basics

Markdown is a simple markup language designed to make authoring web content easy for everyone. Rather than writing HTML and CSS code, Markdown enables the use of a syntax much more like plain-text email. For example the file on the left shows basic Markdown and the resulting output as an HTML file on the right:

```
example.Rmd
```

```
1 Header 1
2 -----
3 This is an R Markdown document. Markdown is a
4 simple formatting syntax for authoring web pages.
5 Use an asterisk mark, to provide emphasis such as
6 *italics* and **bold**.
7 Create lists with a dash:
8 - Item 1
9 - Item 2
10 - Item 3
11
12 You can write `in-line` code with a back-tick.
13 ...
14
15 Code blocks display
16 with fixed-width font
17 ...
18
19 > Blockquotes are offset
```

RStudio: Preview HTML

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark, to provide emphasis such as *italics* and **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

You can write `in-line` code with a back-tick.

Code blocks display  
with fixed-width font

Blockquotes are offset



## R Markdown

### Overview

R Markdown is a format that enables easy authoring of reproducible web reports from R. It combines the core syntax of [Markdown](#) (an easy-to-write plain text format for web content) with embedded R code chunks that are run so their output can be included in the final document.

The concept of R Markdown is similar to [Sweave](#) (a system built-in to R for combining R with LaTeX). In Sweave, Rnw files are "weaved" into TeX files that are then compiled into PDFs. In R Markdown, Rmd files are similarly "weaved" into plain markdown (.md) files that are then compiled into HTML documents.

The R Markdown syntax is described in detail below. At a high-level it is a combination of the following:

- The core [Markdown](#) syntax
- The ability to embed R code and the results of its execution
- Additional extensions provided by the [Sundown](#) library
- Support for LaTeX and MathML equations
- Bundling of images within generated HTML files

### Implementation

The implementation of R Markdown is provided by two packages:

- **knitr** — Weaves Rmd files into plain markdown (.md) files
- **markdown** — Converts markdown files into HTML documents

For example, to run the R code chunks inside an Rmd file and then convert the resulting markdown file into HTML you would execute the following:

```
library(knitr)
library(markdown)
knit("Foo.Rmd")
markdownToHTML("Foo.md", "Foo.html")
```

Note that this can also be accomplished in one step by calling `knitr::knit2html`. However, the fact that converting from Rmd to HTML is broken into two steps allows for the use of alternate markdown rendering programs (e.g. [Pandoc](#)).

RStudio also implements support for rendering R Markdown files into HTML using the Knit HTML commands. This produces a result equivalent to the above code.

### Syntax

The following is a detailed description of the R Markdown syntax. Note that the first two sections covering *Core Markdown* and *R Code Chunks* are always applicable. The remaining sections apply to the rendering of markdown to HTML as implemented by

project organization / literate programming /  
reproducible research

Sweave  
knitr

How JB is currently leaning ....

collaboration / open science

github  
Rforge  
sourceforge

## The Trifecta of Vexing Issues in Scientific Statistical Computing

git

subversion

mercurial

version control / back up / archive

# 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 time I am 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 post)
- Install the latest version of the knitr package:

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

To run the basic working example that produced this blog post:

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

```
install.packages("ggplot2"); install.packages("lattice")
```
- Paste in the contents of [the gist](#) (which contains the following R code):

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  
BY FEEDBURNER

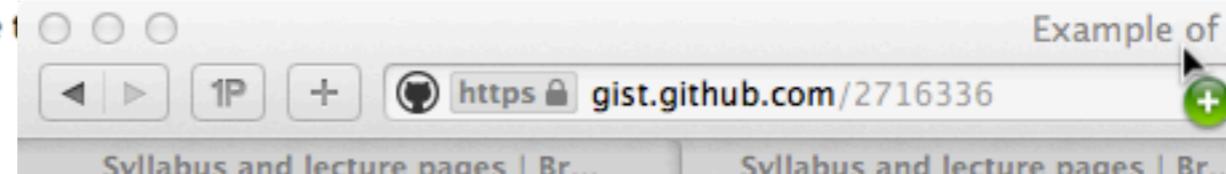
### Subscribe via RSS

[Posts](#) ▾  
[Comments](#) ▾

### Categories

ability academia APAStyle Article

Example of using R Markdown — Gist



# github:gist

New Gist My Gists Starred Gists All Gists

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/authoring/rmarkdown.html) using [knitr] (http://yihui.name/knitr/) in Rstudio 0.96.
2 This combination of tools provides an exciting improvement in usability for
3 reproducible analysis (http://rstudio.github.com/r/15006/102).
```

The screenshot shows a GitHub interface with three main project pages displayed side-by-side, each highlighted by an orange circle.

**Top Project (yihui/knitr):**

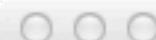
- Header:** yihui/knitr
- Project Name:** yihui / knitr
- Description:** A general-purpose tool for dynamic report generation.
- Code tab:** Selected tab.
- Network tab:**
- Pull Requests:** 11
- Issues:** 114
- Clone In Mac:** button
- Branch:** master
- Latest commit:** bump version by yihui
- File Tree:** R, inst, man, tests, .Rbuildignore, .Rinstignore, .gitignore

**Middle Project (hadley/ggplot2):**

- Header:** yihui/knitr
- Project Name:** hadley / ggplot2
- Description:** A general-purpose tool for dynamic report generation.
- Code tab:** Selected tab.
- Network tab:**
- Pull Requests:** 11
- Issues:** 114
- Clone In Mac:** button
- Branch:** master
- Latest commit:** hadley/plyr by hadley

**Bottom Project (hadley/plyr):**

- Header:** yihui/knitr
- Project Name:** hadley / plyr
- Description:** A R package for splitting, applying and combining large problems into simpler problems — [Read more](#)
- Code tab:** Selected tab.
- Network tab:**
- Pull Requests:** 15
- Issues:** 54
- Clone In Mac:** button
- ZIP:** button
- HTTP:** selected tab
- Git Read-Only:** button
- Read-Only access:** button
- Branch:** master
- Files tab:** Selected tab.
- Commits tab:**
- Branches tab:** 8 branches
- Tags tab:** 20 tags
- Latest commit:** bump version by hadley
- Version bump:**
- Author:** hadley authored 9 months ago



<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.).

Posted on 06

May 2012.

[previous](#)

[next](#)

[history](#)

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.

## 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

project organization / literate programming /  
reproducible research

Sweave  
knitr

How JB is currently leaning ....

collaboration / open science

github  
Rforge  
sourceforge

## The Trifecta of Vexing Issues in Scientific Statistical Computing

git

subversion

mercurial

version control / back up / archive

git – R and version control for the solo data analyst – Stack Overflow

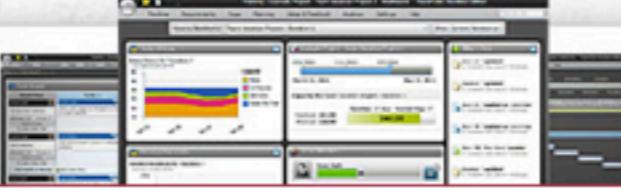
stackoverflow.com/questions/2712421/r-and-version-control-for-the-solo-data-analyst

Syllabus and lecture pages | Br... Syllabus and lecture pages | Br... RStudio git – R and version control for t...

StackExchange log in | careers | chat | meta | about | faq search

 Questions Tags Users Badges Unanswered Ask Question

## R and version control for the solo data analyst

 TeamPulse Project management inspired by Agile best practices  Try It for Free

Many data analysts that I respect use version control. For example:

**43** • <http://github.com/hadley/>  
• See comments on <http://permut.wordpress.com/2010/04/21/revision-control-statistics-blog/>

However, I'm evaluating whether adopting a version control system such as git would be worthwhile.

**30** A brief overview: I'm a social scientist who uses R to analyse data for research publications. I don't currently produce R packages. My R code for a project typically includes a few thousand lines of code for data input, cleaning, manipulation, analyses, and output generation. Publications are typically written using LaTeX.

With regards to version control there are many benefits which I have read about, yet they seem to be less relevant to the solo data analyst.

- **Backup:** I have a backup system already in place.
- **Forking and rewinding:** I've never felt the need to do this, but I can see how it could be useful (e.g., you are preparing multiple journal articles based on the same dataset; you are preparing a report that is updated monthly, etc)
- **Collaboration:** Most of the time I am analysing data myself, thus, I wouldn't get the collaboration benefits of version control.

There are also several potential costs involved with adopting version control:

- Time to evaluate and learn a version control system
- A possible increase in complexity over my current file management system

However, I still have the feeling that I'm missing something. General guides on version control seem to be addressed more towards computer scientists than data analysts.

tagged  
[git](#) × 19828  
[r](#) × 17640  
[version-control](#) × 6656

asked 2 years ago  
viewed 5150 times  
active 2 months ago

**Community Bulletin**  
blog Join the Stack Exchange team – we're hiring!

 Apptivate.ms  
Develop great apps for Windows 8 for a chance to win \$5,000!

 Microsoft

The screenshot shows the RStudio website with the URL [rstudio.org/docs/version\\_control/overview](http://rstudio.org/docs/version_control/overview) in the address bar. The page title is "Using Version Control with RStudio". The navigation menu includes Home, Screenshots, Download, Docs (which is selected), Support, Development, and Blog. A large blue "R" logo is on the left. The main content discusses version control benefits, supported systems (Git, Subversion), and RStudio Projects.

## 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

How JB is currently leaning ....

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
- 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)
```

## 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](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 filesystem

[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"
```

# Getting data out of R

See Chapter 2 of  
Spector (2008).

```
write.table(gDat,  
            jPaste(whereAmI, "data/gapminderDataFiveYear.txt"),  
            quote = FALSE, sep = "\t", row.names = FALSE)
```

- **write.table will be your main function for this; it writes plain text, human-readable files.**
- I like to use the args above, by default (~~why don't I package that as a handy function? now I have!~~)

country	year	pop	continent	lifeExp	gdpPerCap
Afghanistan	1952	8425333	Asia	28.801	779.4453145
Afghanistan	1957	9240934	Asia	30.332	820.8530296
Afghanistan	1962	10267083	Asia	31.997	853.10071
Afghanistan	1967	11537966	Asia	34.02836.1971382	
Afghanistan	1972	13079460	Asia	36.088	739.9811058
Afghanistan	1977	14880372	Asia	38.438	786.11336
Afghanistan	1982	12881816	Asia	39.854	978.0114388
Afghanistan	1987	13867957	Asia	40.822	852.3959448
Afghanistan	1992	16317921	Asia	41.674	649.3413952
Afghanistan	1997	22227415	Asia	41.763	635.341351
Afghanistan	2002	25268405	Asia	42.129	726.7340548
Afghanistan	2007	31889923	Asia	43.828	974.5803384
Albania	1952	1282697	Europe	55.231601.056136	
Albania	1957	1476505	Europe	59.281942.284244	
Albania	1962	1728137	Europe	64.822312.888958	
Albania	1967	1984060	Europe	66.222760.196931	

# Getting data out of R

See Chapter 2 of  
Spector (2008).

```
write.table(gDat,  
            jPaste(whereAmI, "data/gapminderDataFiveYear.txt"),  
            quote = FALSE, sep = "\t", row.names = FALSE)
```

- I only use “exotic” import and export functionality with a very good reason. What do I mean by exotic? importing/exporting from/to .xls files, databases, etc.
- Why am I such a Luddite? I’ve been burned with R changing, add-on R packages changing or vanishing, Excel changing, Excel not being installed, maddening Mac/Windows incompatibility issues in Excel, blah blah blah and now I maniacally save all important input, intermediate, and output files in the plainest form possible. Learn from me so that I did not suffer in vain.

# Getting stuff out of R

See Chapter 2 of  
Spector (2008).

- In certain situations, it is advisable to save R objects or, very rarely, an entire R workspace. Examples:
  - a vital, non-rectangular object, for example, a fitted nonlinear model object or a classification & regression tree
  - an object you will continue to need that also took a nontrivial amount of compute time to create
  - a vital classed object you will continue to need whose creation required some add-on software that might change dramatically in the next version or that might become “abandonware”

# Getting stuff out of R

See Chapter 2 of  
Spector (2008).

- In certain situations, it is advisable to save R objects or, very rarely, an entire R workspace. Examples cont'd:
  - a `data.frame` which holds factors for which you exerted yourself to set the order of the levels, i.e. to something other than the alphanumeric default ordering (In this case I usually save as plain text with `write.table()` AND as an R object with `save()` -- I wear a belt and suspenders!)
- Relevant functions: `save()`, `save.image()`, `load()`

# dput() especially helpful for creating self-contained examples when asking for help

```
> jDat
      country year      pop continent lifeExp gdpPercap
504     Eritrea 2007 4906585     Africa  58.040   641.3695
1080    Nepal 2007 28901790     Asia   63.785 1091.3598
<snip, snip>

> dput(jDat)
structure(list(country = structure(c(4L, 7L, 1L, 6L, 3L, 2L,
5L, 8L), .Label = c("Chad", "Costa Rica", "Cuba", "Eritrea",
"Germany", "Jamaica", "Nepal", "Norway"), class = "factor"),
year = c(2007L, 2007L, 2007L, 2007L, 2007L, 2007L, 2007L,
2007L), pop = c(4906585, 28901790, 10238807, 2780132, 11416987,
4133884, 82400996, 4627926), continent = structure(c(1L,
3L, 1L, 2L, 2L, 4L, 4L), .Label = c("Africa", "Americas",
"Asia", "Europe"), class = "factor"), lifeExp = c(58.04,
63.785, 50.651, 72.567, 78.273, 78.782, 79.406, 80.196),
gdpPercap = c(641.3695236, 1091.359778, 1704.063724, 7320.880262,
8948.102923, 9645.06142, 32170.37442, 49357.19017)), .Names =
c("country", "year", "pop", "continent", "lifeExp", "gdpPercap"),
row.names = c(504L, 1080L, 276L, 792L, 396L, 360L, 576L, 1152L),
class = "data.frame")

> rm(jDat)

> jDat
Error: object 'jDat' not found
```

# dput() especially helpful for creating self-contained examples when asking for help

```
> jDat <- structure(list(country = structure(c(4L, 7L, 1L, 6L, 3L, 2L,
+ 5L, 8L), .Label = c("Chad", "Costa Rica", "Cuba", "Eritrea",
+ "Germany", "Jamaica", "Nepal", "Norway"), class = "factor"),
+ year = c(2007L, 2007L, 2007L, 2007L, 2007L, 2007L,
+ 2007L), pop = c(4906585, 28901790, 10238807, 2780132, 11416987,
+ 4133884, 82400996, 4627926), continent = structure(c(1L,
+ 3L, 1L, 2L, 2L, 4L, 4L), .Label = c("Africa", "Americas",
+ "Asia", "Europe"), class = "factor"), lifeExp = c(58.04,
+ 63.785, 50.651, 72.567, 78.273, 78.782, 79.406, 80.196),
+ gdpPercap = c(641.3695236, 1091.359778, 1704.063724, 7320.880262,
+ 8948.102923, 9645.06142, 32170.37442, 49357.19017)), .Names =
c("country",
+ "year", "pop", "continent", "lifeExp", "gdpPercap"), row.names =
c(504L,
+ 1080L, 276L, 792L, 396L, 360L, 576L, 1152L), class = "data.frame")
```

```
> jDat
  country year    pop continent lifeExp gdpPercap
504   Eritrea 2007 4906585     Africa  58.040  641.3695
1080   Nepal 2007 28901790      Asia  63.785 1091.3598
276     Chad 2007 10238807     Africa  50.651 1704.0637
792   Jamaica 2007  2780132 Americas  72.567  7320.8803
396     Cuba 2007 11416987 Americas  78.273  8948.1029
360 Costa Rica 2007  4133884 Americas  78.782  9645.0614
576 Germany 2007  82400996   Europe  79.406 32170.3744
1152 Norway 2007  4627926   Europe  80.196 49357.1902
```

I literally copied the previous output from dput() to create this assignment statement. Would allow someone else to recreate jDat from just a \*.R file, versus sending a \*.R file and some raw, importable data.

dput

package:base

R Documentation

Write an Object to a File or Recreate it

Description:

Writes an ASCII text representation of an R object to a file or connection, or uses one to recreate the object.

Usage:

```
dput(x, file = "",  
     control = c("keepNA", "keepInteger", "showAttributes"))  
  
dget(file)
```

Arguments:

**x**: an object.

**file**: either a character string naming a file or a connection. '""' indicates output to the console.

**control**: character vector indicating deparsing options. See '`.deparseOpts`' for their description.

Details:

'`dput`' opens '`file`' and deparses the object '`x`' into that file. The object name is not written (unlike '`dump`'). If '`x`' is a function the associated environment is stripped. Hence scoping information can be lost.

# Getting tables out of R

**Do not** type statistical results into tables in LaTeX or Word or ... (or, at least, that should be the exception, not the rule).

You will make mistakes. You will grow weary of it each time you need to update.

Automate.

Where might you want to put a table computed by R?

In Excel. Easy. Write to a delimited file and import. Done.

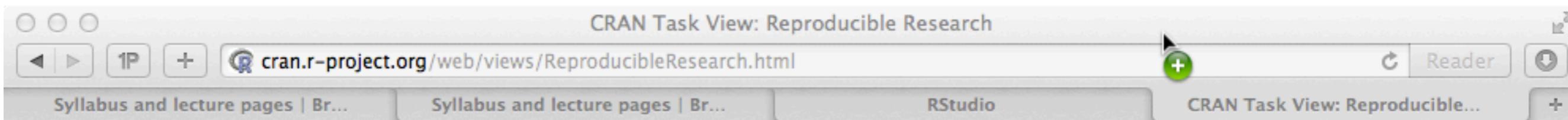
In a Word or Pages document, i.e. word processors that have a real concept of a table.

In a Powerpoint or Keynote document. Ditto.

On the web.

In a LaTeX document.

For more than you ever wanted to know about this subject, check out the [CRAN Task View for Reproducible Research](#).



## CRAN Task View: Reproducible Research

**Maintainer:** Max Kuhn

**Contact:** max.kuhn at pfizer.com

**Version:** 2012-07-11

The goal of reproducible research is to tie specific instructions to data analysis and experimental data so that scholarship can be recreated, better understood and verified.

R largely facilitates reproducible research using literate programming; a document that is a combination of content and data analysis code. The `Sweave` function (in the base R `utils` package) can be used to blend the subject matter and R code so that a single document defines the content and the algorithms.

Basic packages can be structured into the following groups:

- *LaTeX Markup* : The `Hmisc`, `xtable` and `tables` packages contain functions to write R objects into LaTeX representations. `Hmisc` also includes methods for translating strings to proper LaTeX markup (e.g., "`>=`" to "`\geq`"). Animations can be inserted into LaTeX documents being converted to PDF via the `animation` package. The `pictex` function in the base `grDevices` package is a PicTeX graphics driver. The `makesweave` package for Linux streamlines the generation of Sweave files using `make`.

**Low-tech solution for Word and Keynote and probably other similar programs ....**

**Write table to tab-delimited file  
open in text editor, copy contents to clipboard  
paste into table-type receptacle in work processing /  
presentation software OR convert text to table**

**details will differ but should be workable in many contexts**

**figure out how to do this with the software you use!**

Examples in excruciating detail:

In the script `01-initialClassListProcessing.R` I tabulate the students in this class by subject and degree:

```
## cross-tabulate subject by degree
(jTab <- addmargins(table(cDat$specSubj, cDat$degree)))

write.table(jTab,
            file = paste(whereAmI,"results/whosInHere.txt",sep=""),
            quote = FALSE, sep = "\t")
```

Here's what that file looks like in Emacs for me:

EXCH	MSC	PHD	UNCL	Sum					
1	1	0	0	1	3				
Experimental Medicine		0	0	1	0	0	1		
Forestry	0	0	1	0	0	1			
Lib, Arch and Info Stud		0	0	0	1	0	1		
Mathematics	0	0	1	0	0	1			
Mechanical Engineering		0	0	0	1	0	1		
Resource Mgmt/Envirn Stud		0	0	0	1	0	1		
Statistics	0	0	13	2	0	15			
Sum	1	1	16	5	1	24			

I see it has 10 rows and 7 columns. If counting that's too annoying, just guess high!

For Keynote: I create a new empty table with enough rows and columns.

For Keynote: I copy the contents of my plain text, tab delimited file into the clipboard.

I select the *upper left cell* of the empty table and paste.

I do not select the whole table.

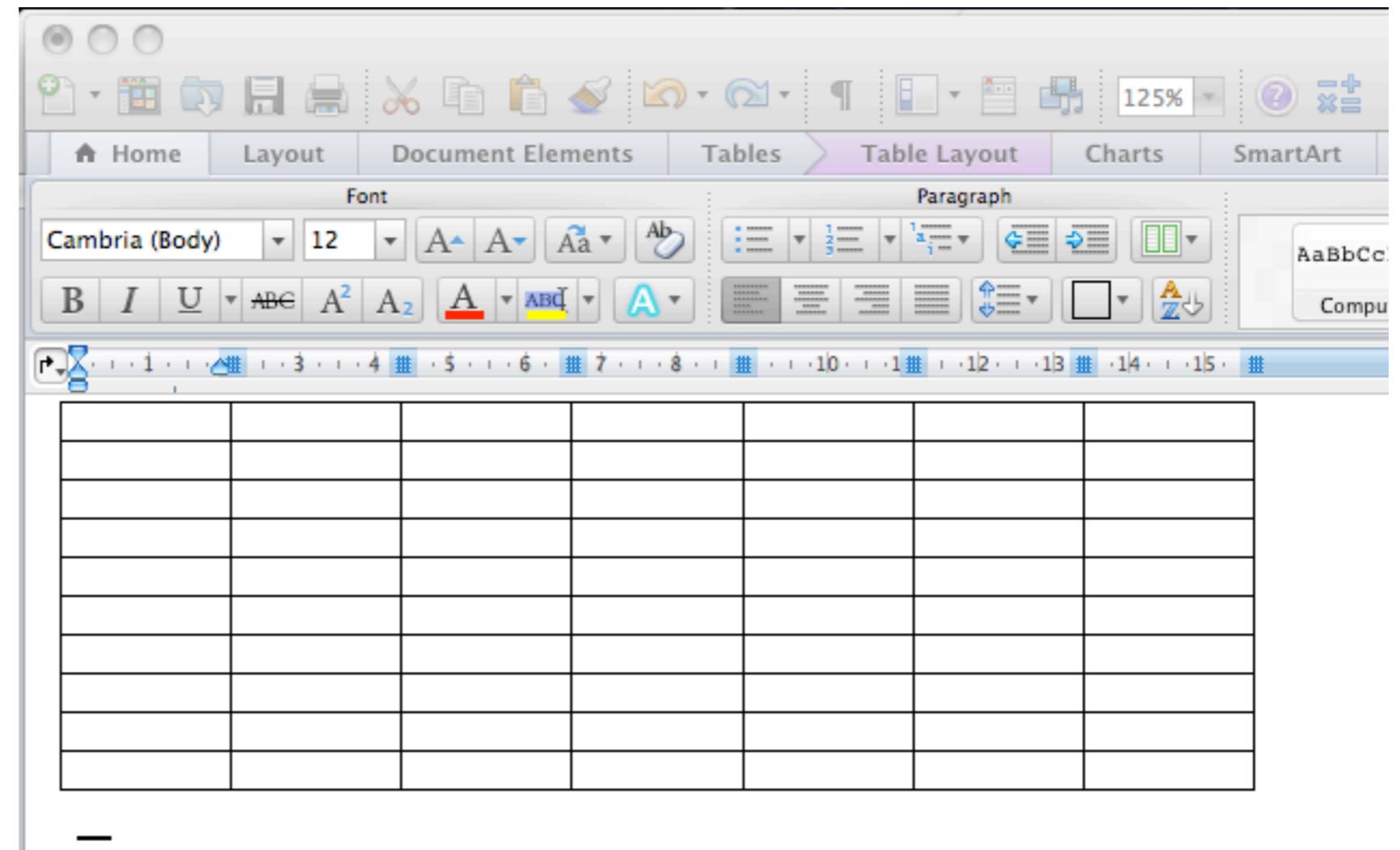
I do not double click into the cell, thereby putting the cursor there.

I repeat: I select the upper left cell and then paste.

Voilà!

	EXCH	MSC	PHD	UNCL	Sum	
	1	1	0	0	1	3
Exper	0	0	1	0	0	1
Fores	0	0	1	0	0	1
Lib,	0	0	0	1	0	1
Mathe	0	0	1	0	0	1
Mecha	0	0	0	1	0	1
Resou	0	0	0	1	0	1
Stati	0	0	13	2	0	15
Sum	1	1	16	5	1	24

For Word: I create a new empty table with enough rows and columns.



I copy the contents of my plain text, tab delimited file into the clipboard.....

For Word:

I select the whole table and paste.

I do not select a single cell.

I do not click into a cell, thereby putting the cursor there.

I repeat: I select the whole table and then paste.

Voilà!

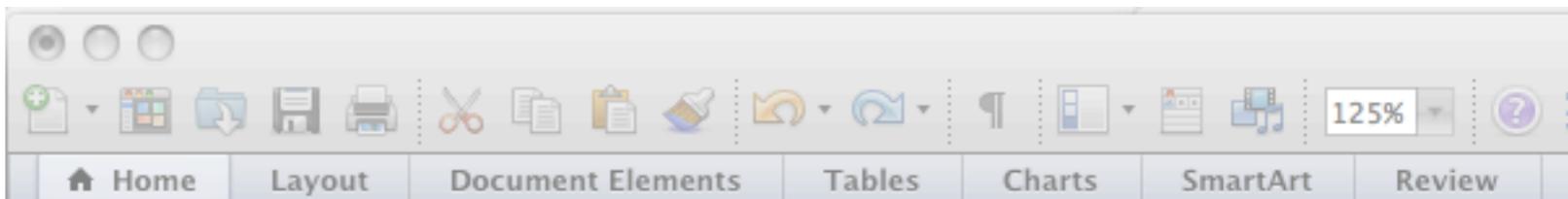
	EXCH	MSC	PHD	UNCL	Sum	
	1	1	0	0	1	3
Experimental Medicine	0	0	1	0	0	1
Forestry	0	0	1	0	0	1
Lib, Arch and Info Stud	0	0	0	1	0	1
Mathematics	0	0	1	0	0	1
Mechanical Engineering	0	0	0	1	0	1
Resource Mgmt/Envirn Stud	0	0	0	1	0	1
Statistics	0	0	13	2	0	15
Sum	1	1	16	5	1	24

For Word, version 2: I copy the contents of my plain text, tab delimited file into the clipboard.

I paste into Word. I select all of what I pasted.

With the mouse (!), Table --> Convert --> Convert Text to Table ...  
(dialog box where you can adjust number of rows/columns, specify  
the delimiter, etc.) --> OK

Voilà!



The screenshot shows the Microsoft Word ribbon with the 'Tables' tab selected. Below the ribbon is the standard toolbar with icons for file, print, and other document operations. The main area displays a table with data from a tab-delimited file. The table has columns labeled EXCH, MSC, PHD, UNCL, Sum, and an unnamed column at the end. The rows represent different academic fields and their counts across these categories.

	EXCH	MSC	PHD	UNCL	Sum	
	1	1	0	0	1	3
Experimental Medicine	0	0	1	0	0	1
Forestry	0	0	1	0	0	1
Lib, Arch and Info Stud	0	0	0	1	0	1
Mathematics	0	0	1	0	0	1
Mechanical Engineering	0	0	0	1	0	1
Resource Mgmt/Envirn Stud	0	0	0	1	0	1
Statistics	0	0	13	2	0	15
Sum	1	1	16	5	1	24

# Getting tables out of R

I've showed you a low-tech solution for Word and Keynote. Can someone work on PowerPoint?

Good but old thread on how to copy from R to the clipboard; first time I've ever seen something work better on Windows! Still relevant? I don't know know; am on Mac.

The R2wd package looks intriguing but I would worry about fiddliness: “R2wd:Write MS-Word documents from R. This package uses the statconnDCOM server to communicate with MS-Word via the COM interface.”

# How to create an HTML table

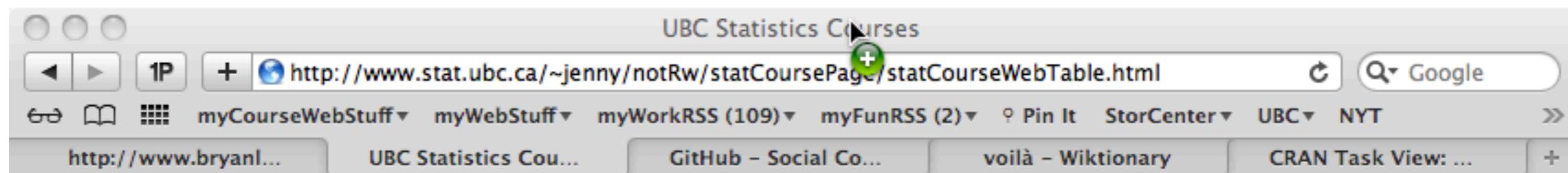
Use one of these packages to write HTML tables: xtable,  
Hmisc, R2HTML, hwriter

I use R2HTML .....

Example of using R2HTML to take a class list as a data.frame, find student photos and student work automatically, and create this table.

who	photo	a04Remote
bretschneider, hannes MSC Statistics (M)		<ul style="list-style-type: none"><li><a href="#">a04-bretschneider-fig1.pdf</a></li><li><a href="#">a04-bretschneider-fig2.pdf</a></li><li><a href="#">a04-bretschneider-fig3.pdf</a></li><li><a href="#">a04-bretschneider-fig4.pdf</a></li><li><a href="#">a04-bretschneider-functions.r</a></li><li><a href="#">a04-bretschneider-functions.r.html</a></li><li><a href="#">a04-bretschneider-output.txt</a></li><li><a href="#">a04-bretschneider.pdf</a></li><li><a href="#">a04-bretschneider.r</a></li><li><a href="#">a04-bretschneider.r.html</a></li><li><a href="#">a04-bretschneider.tex</a></li></ul>
chen, jason MSC Statistics (M)		<ul style="list-style-type: none"><li><a href="#">a04-chen-fig01.pdf</a></li><li><a href="#">a04-chen-fig02.pdf</a></li><li><a href="#">a04-chen.R</a></li><li><a href="#">a04-chen.R.html</a></li><li><a href="#">a04-chen.txt</a></li></ul>
cormier, eric MSC Statistics (M)		<ul style="list-style-type: none"><li><a href="#">a04-Cormier.R</a></li><li><a href="#">a04-Cormier.R.html</a></li><li><a href="#">a04-cormier-fig1.pdf</a></li><li><a href="#">a04-cormier-fig2.pdf</a></li><li><a href="#">a04-cormier-fig3.pdf</a></li><li><a href="#">a04-cormier-fig4.pdf</a></li><li><a href="#">a04-cormier.pdf</a></li></ul>
fu, eric MSC Statistics (M)		<ul style="list-style-type: none"><li><a href="#">a04-fu-fig.pdf</a></li><li><a href="#">a04-fu.R</a></li><li><a href="#">a04-fu.R.html</a></li><li><a href="#">a04-fu.RData</a></li><li><a href="#">a04-fu.pdf</a></li></ul>

# I used R2HTML to create this web table of STAT courses



Courses offered by the UBC Department of Statistics  
(draft of what one should see at <http://www.stat.ubc.ca/Courses/> in the future??)

## General information on our courses

- [STAT courses as listed in the Academic Calendar](#)
- Enhanced course listing that includes links to extras like course outlines, course webpages, and more (the table Jenny is developing; see draft below)

## Current offerings of our courses

- [STAT course schedule as provided by UBC Student Service Centre](#)
- [Courses running right now -- a list and a timetable view](#)
- [SLATE -- find course webpages for many current courses here](#)

courseNo	title	desc	links
100	Statistical Thinking 3 credits [3-0-1]	Explores the development and use of statistical thinking in the modern world. The aim is to develop statistical literacy and demonstrate applications of statistics in research and society. Students who obtain credit for any of STAT 306, 307, 308, 344 cannot in the same or later term gain credit for STAT 100. Prerequisite: Principles of Mathematics 12 or Pre-Calculus 12	<a href="#">Course overview</a> <a href="#">STAT100-2011-gustafson.pdf</a>
200	Elementary Statistics for Applications 3 credits	Classical, nonparametric, and robust inferences about means, variances, and analysis of variance, using computers. Emphasis on problem formulation, assumptions, and interpretation. See the Faculty of Science Credit Exclusion Lists: <a href="http://www.students.ubc.ca/calendar/proof/edit/index.cfm?tree=12,215,410,414">www.students.ubc.ca/calendar/proof/edit/index.cfm?tree=12,215,410,414</a> .	<a href="#">STAT200-2011-yu.pdf</a>

```

whereAmI <- "/Users/jenny/adminService/2011-06-statWebCourseTable/"
library(R2HTML)

cDat <- read.table(file = jPaste(whereAmI, "data/academicCalendarEnhanced.txt"),
                     sep = "|", colClasses = "character", header = TRUE,
                     quote = "\'")

targDir <- jPaste(whereAmI, "results/R2HTML/")
dir.create(targDir)

file.symlink(from = jPaste(.Library, "/R2HTML/output/R2HTML.css"),
             to = targDir)
## if get warnings about files and directories existing ... ignore

target <- HTMLInitFile(outdir = targDir,
                       filename = "statCourseWebTable",
                       Title = "UBC Statistics Courses",
                       useLaTeX = FALSE, useGrid = FALSE)

## insert basic content for a new course page
system(paste("cat",
             jPaste(whereAmI, "data/coursePageInnards.html"),
             ">>",
             jPaste(targDir, "statCourseWebTable.html")))

HTMLhr(file = target, Size = "1")

HTML(cDat, file = target,
      Border = 2, innerBorder = 1, row.names = FALSE)

HTMLEndFile()

```

**Code that created web table of STAT courses**

# How to create a LaTeX table

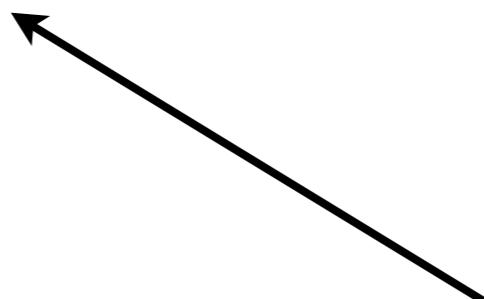
Use one of these packages to write LaTeX tables: xtable,  
Hmisc

I don't use these because I've abandoned LaTeX (!), at least for now.

# Saving analytical results, when write.table() isn't appropriate

- sink() will divert your R output to a file

```
> sink(jPaste(whereAmI, "sinkDemo.txt"))
> t.test(pheno ~ chromo, kDat)
> wilcox.test(pheno ~ chromo, kDat)
> ks.test(kDat$pheno[kDat$chromo == 6], kDat$pheno[kDat$chromo == 7])
> sink()
```



Notice the  
results aren't  
showing up!

# Contents of “sinkDemo.txt”

Welch Two Sample t-test

```
data: pheno by chromo
t = 1.4982, df = 158.612, p-value = 0.1361
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.0714768 0.5206902
sample estimates:
mean in group 6 mean in group 7
8.767099    8.542492
```

Wilcoxon rank sum test with continuity correction

```
data: pheno by chromo
W = 29063, p-value = 0.1896
alternative hypothesis: true location shift is not equal to 0
```

Two-sample Kolmogorov-Smirnov test

```
data: kDat$pheno[kDat$chromo == 6] and kDat$pheno[kDat$chromo == 7]
D = 0.0964, p-value = 0.3933
alternative hypothesis: two-sided
```

# `sink()`

- Not a great general purpose, long-run strategy, but useful sometimes
- Must write and debug your code first, then implement `sink()`, since you “fly blind” while the sink is in place
- Reminds me of the ‘correct’, but annoying way to make PDF files
- Nice when you are ‘source()’ing code and/or running R non-interactively
- Helpful for writing key facts and numbers to file that must be incorporated into written English (vs. a table)