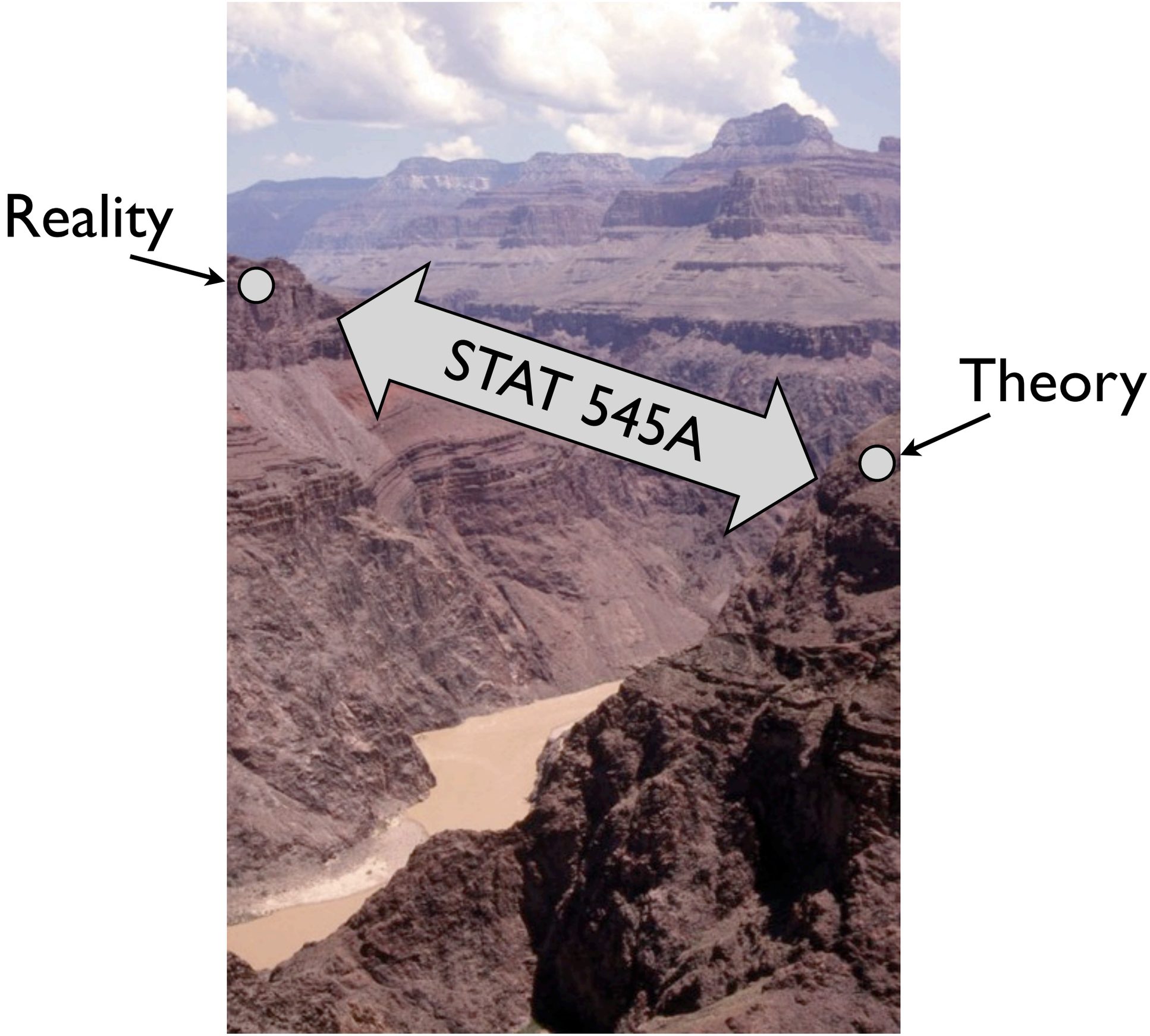
wordle from the previous assignment 1 reflections

STAT 545A
Class meeting #2
Monday, September 10, 2012

Dr. Jennifer (Jenny) Bryan

Department of Statistics and Michael Smith Laboratories


Reality
STAT 545A
Theory

Two inter-related goals for course

- Foster your development of a personal philosophy on data analysis.
- Strengthen your data analysis skills.

Two main goals for statistical graphics

- To facilitate comparisons.
- To identify trends.

Two key plot elements of data analytical stories

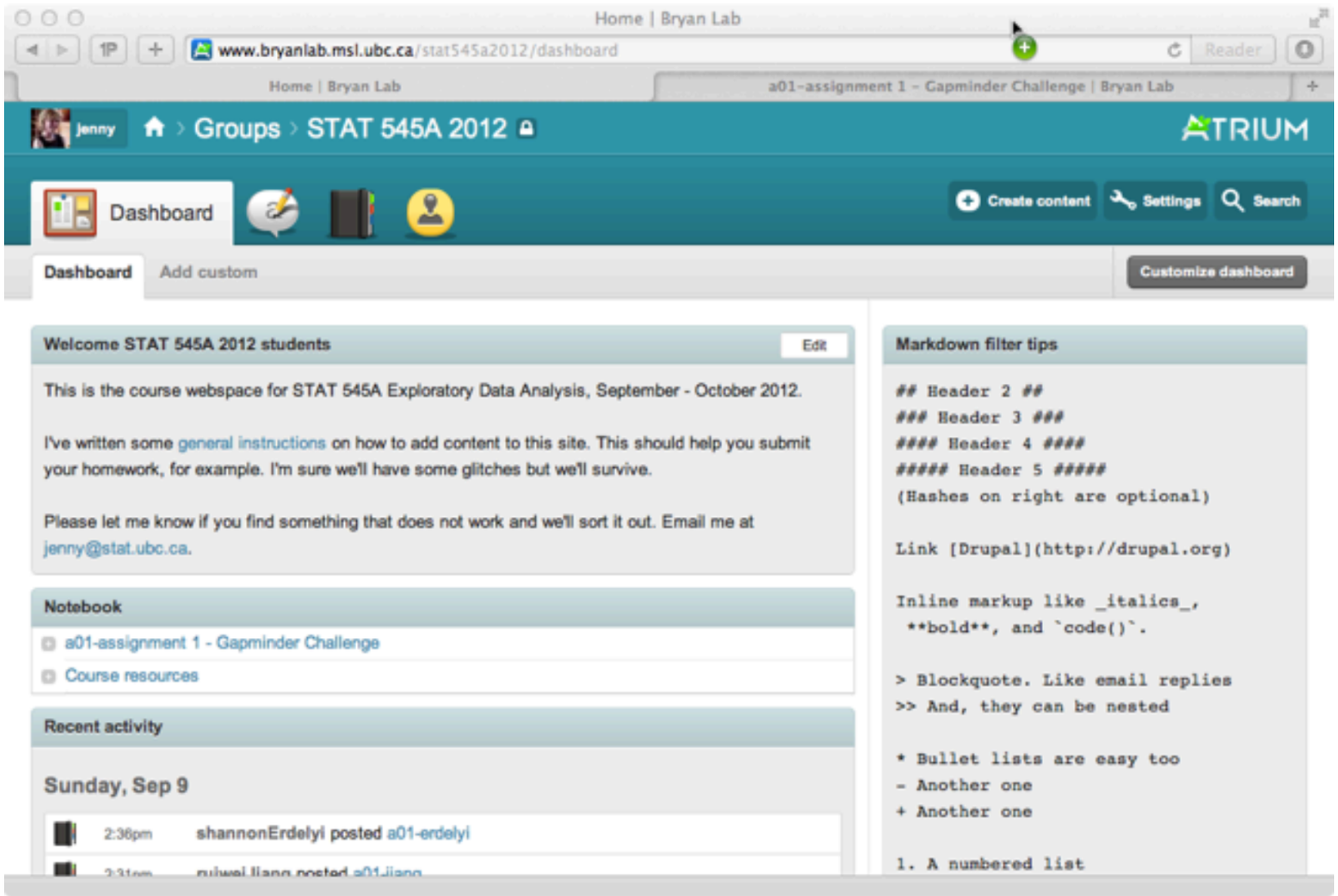
- Build trust by showing stuff that people expect.
- Generate excitement by showing stuff that they don't.

Admin & communication

Collaborative web space set up here:

http://www.bryanlab.msl.ubc.ca/stat545a2012/

I must add you as user, so contact me if you are not set up yet.



Excellent source of relevant eBooks: SpringerLink. Access via this UBC Library page

The Use R series is especially relevant for this course.
Also look at books listed under Statistics Computing and Software.

Another good repository of CRC Press books: STATSnetBASE. Access via this UBC Library page. Check out the Computer Science & Data Analysis series.

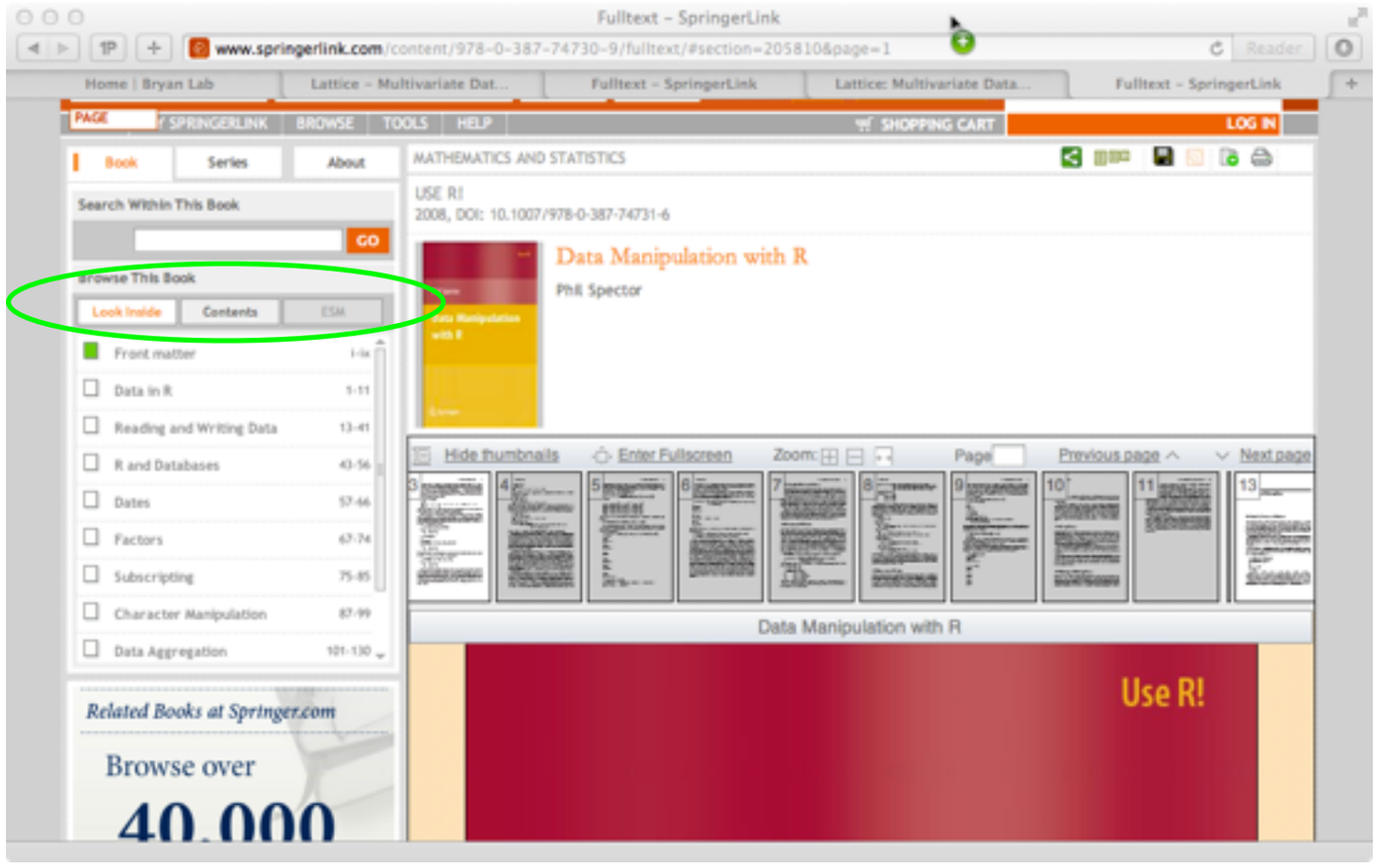
If accessing from a UBC network, you should automatically be recognized as a valid user.

If accessing from off campus, read UBC Library's page about "Connect from Home".

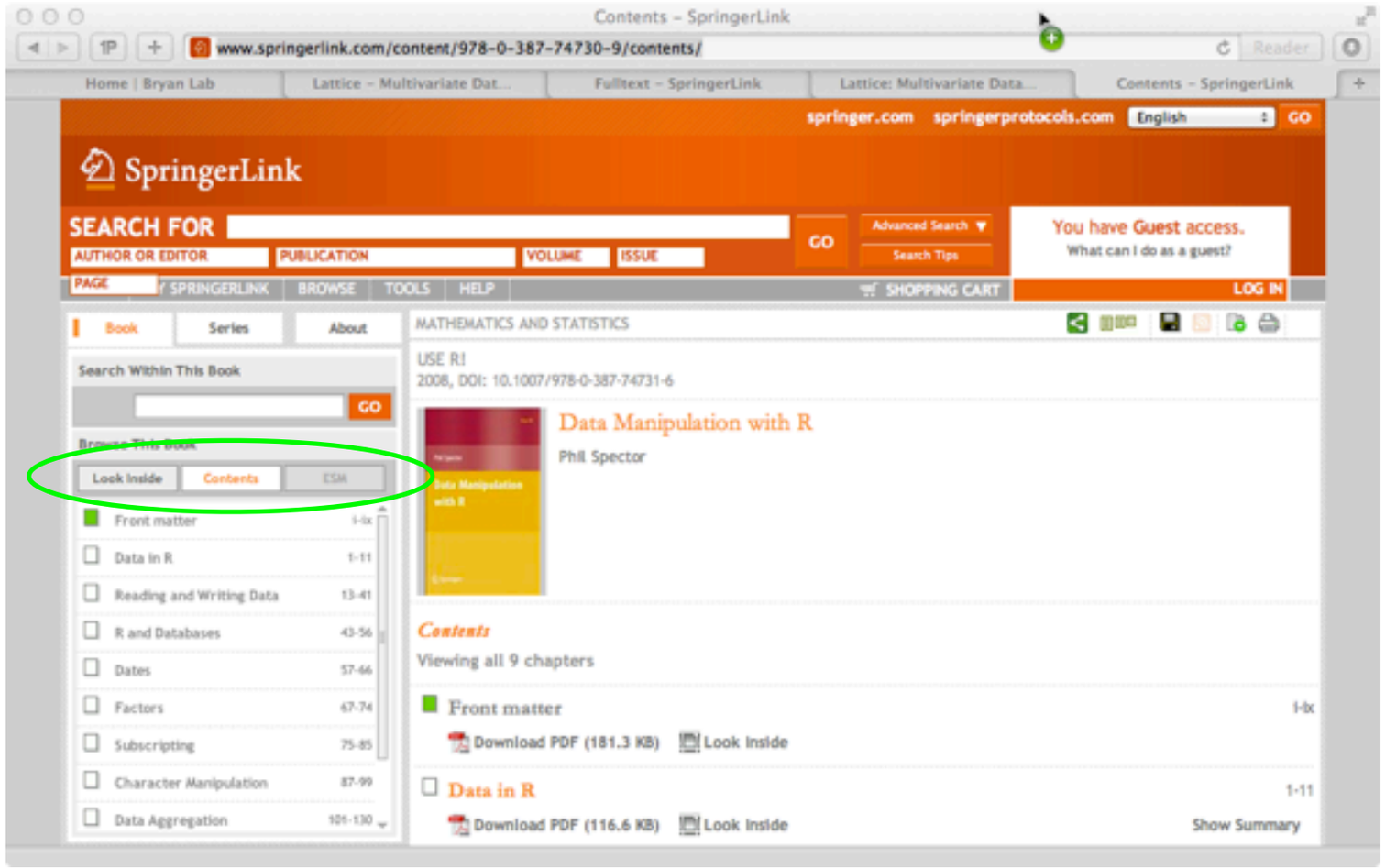
Personal recommendations:

Phil Spector's book "Data Manipulation with R"
author's webpage (lots of great material here)
on SpringerLink:
- Look Inside view (read book online)
- Contents view (facilitates downloading chapters as PDFs)
Google books search inside of it

Deepayan Sarkar's book "Lattice: Multivariate Data Visualization with R"
webpage w/ all book's figures and associated code
on SpringerLink:
- Look Inside view
  - Contents view
Google books search inside of it



Different views of Springer eBooks.



Personal recommendations:

Paul Murrell's book "R Graphics"
author's webpages for 1st edition and 2nd edition
(gives R code to produce all figures)
STATSnetBASE (read 1st edition online; JB owns both in hard copy if want to borrow)
Google books search

Venables and Ripley "Modern applied statistics with S"
authors' webpage
Springer 4th edition 2002
sadly not available via SpringerLink; getting a little old; JB owns hard copy if want to borrow

Also look promising

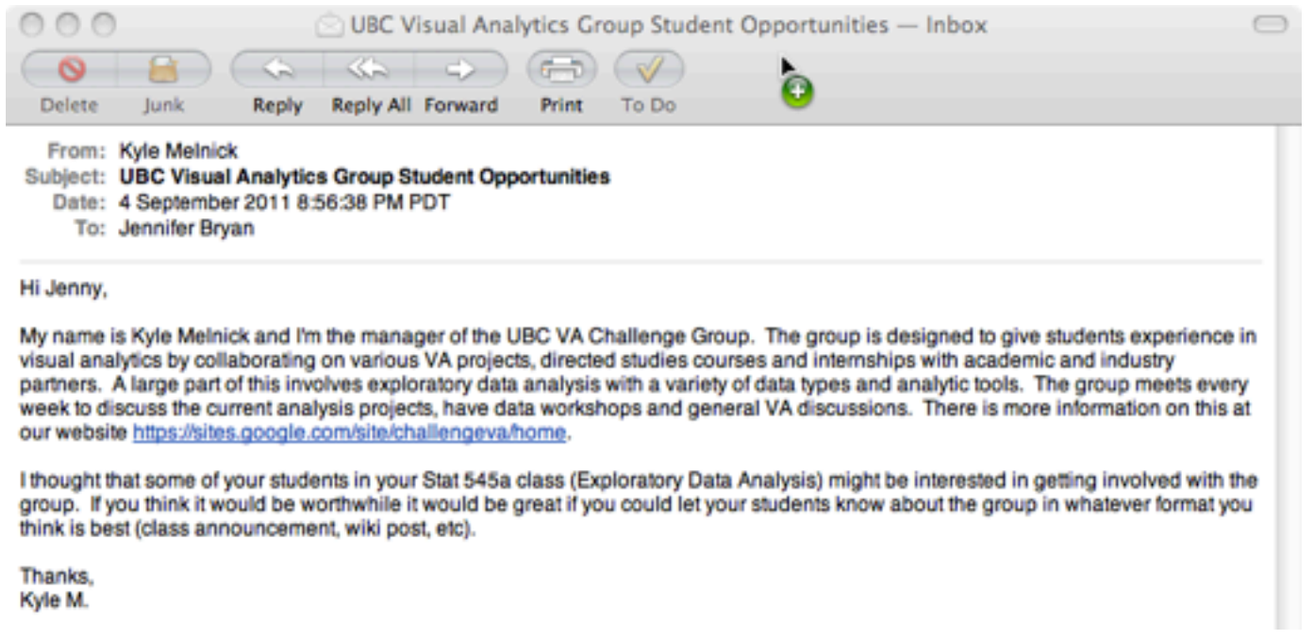
- Zuur, Ieno, Meesters (2009) A Beginner's Guide to R. Springer. Available via SpringerLink.
- Wickham, H (2009) ggplot2 Elegant Graphics for Data Analysis. Springer. Available via SpringerLink.

More sources & books of interest

- Cleveland, William S (1993). Visualizing Data. AT&T Bell Laboratories.
- STATSnetBASE (CRC Press)
  - A Handbook of Statistical Analyses Using R by Brian Everitt and Torsten Hothorn.
  - Handbook of Statistical Analyses using S-Plus, Second Edition by Brian Everitt.
  - R Programming for Bioinformatics by R Gentleman.

Andrew Wade VA Challenge Program

https://sites.google.com/site/challengeva/


UBC Visual Analytics Group Student Opportunities — Inbox

Delete Junk Reply Reply All Forward Print To Do

From: Kyle Melnick
Subject: **UBC Visual Analytics Group Student Opportunities**
Date: 4 September 2011 8:56:38 PM PDT
To: Jennifer Bryan

Hi Jenny,

My name is Kyle Melnick and I'm the manager of the UBC VA Challenge Group. The group is designed to give students experience in visual analytics by collaborating on various VA projects, directed studies courses and internships with academic and industry partners. A large part of this involves exploratory data analysis with a variety of data types and analytic tools. The group meets every week to discuss the current analysis projects, have data workshops and general VA discussions. There is more information on this at our website https://sites.google.com/site/challengeva/home.

I thought that some of your students in your Stat 545a class (Exploratory Data Analysis) might be interested in getting involved with the group. If you think it would be worthwhile it would be great if you could let your students know about the group in whatever format you think is best (class announcement, wiki post, etc).

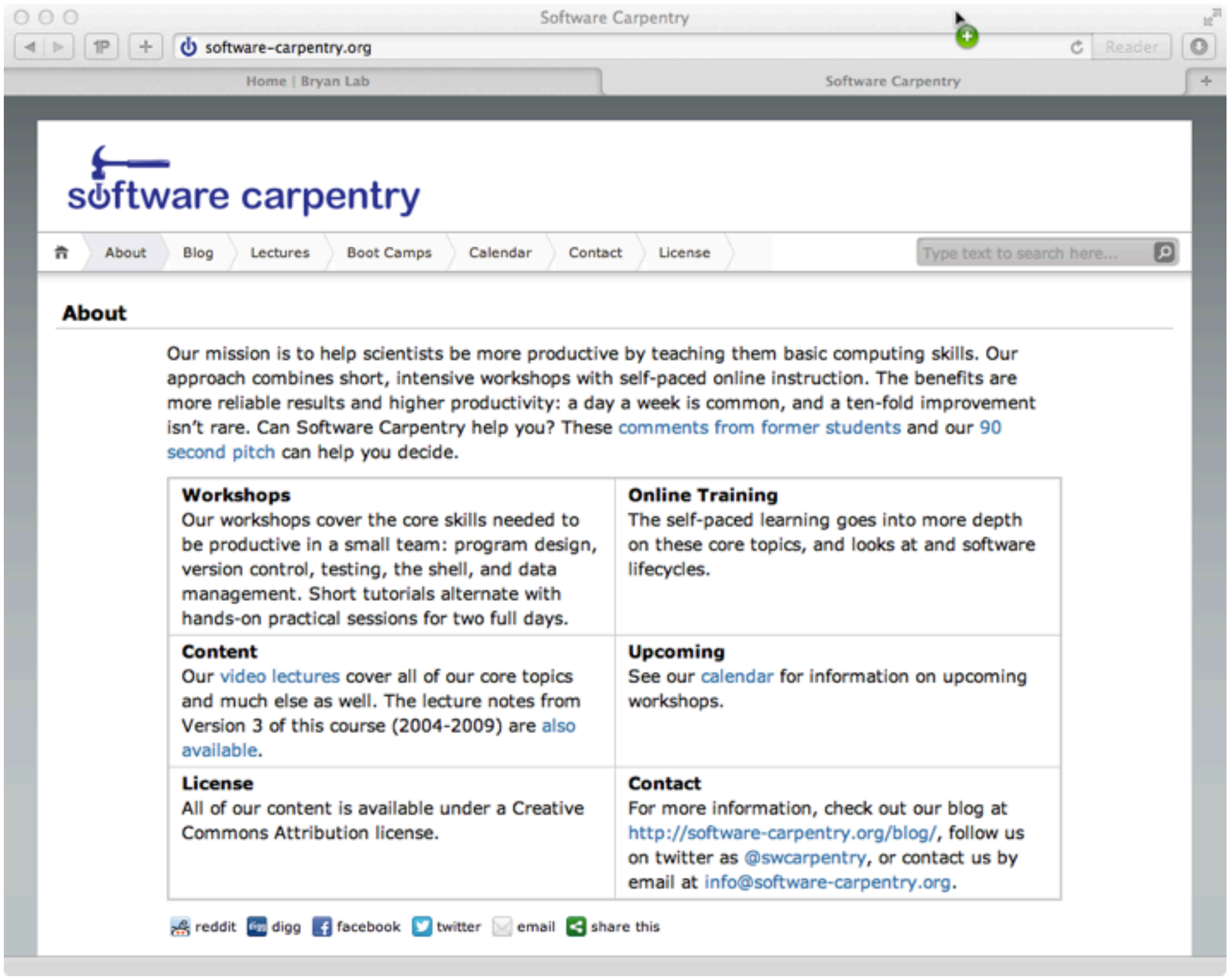
Thanks,
Kyle M.

Some of us UBC Stat faculty met with folks from VIVA (Vancouver Institute for Visual Analytics) this summer -- this program is affiliated with VIVA.
Great resource for inspiration, project ideas, etc. in this area. I'd like to hear about it if you get involved.

Software Carpentry workshop at UBC Oct 18 - 19

first (?) use of R, in addition to python, for the programming bits

http://software-carpentry.org



Sept

	5 course intro
10 Gapminder shock therapy	12
17	19
24	26

Oct

1	3
Thanksgiving	10
15	17

12 class meetings

Breakdown of methods

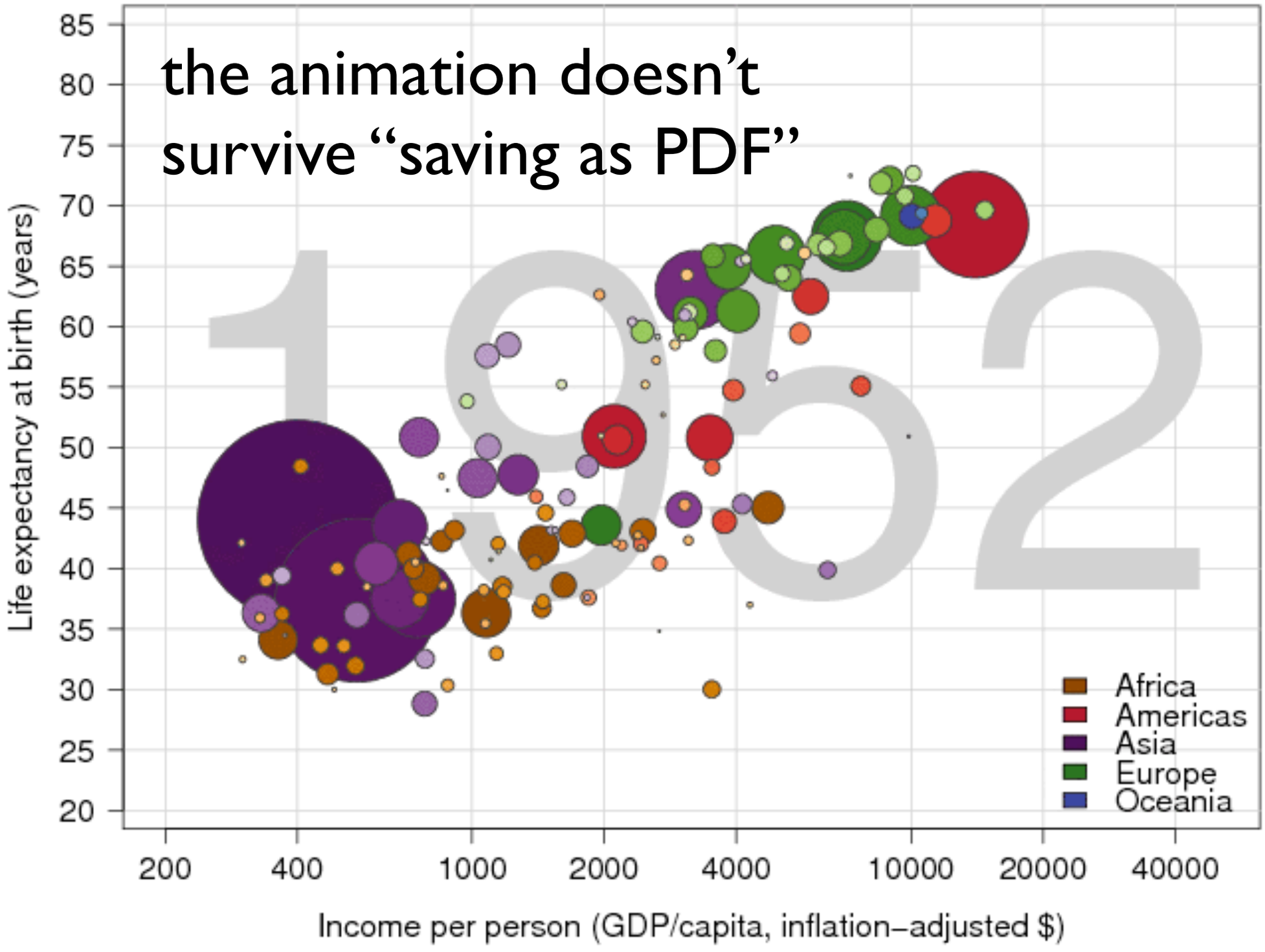
- univariate data (single quantitative variable) and, optionally, a categorical variable
- bivariate data (two quantitative variables) and, optionally, a categorical variable
- multivariate data (3 or more quantitative variables)
- multiway data (single quantitative variable, two or more categorical variables)

STAT 545A Assignment #1

Spend two hours trying to replicate this sequence of graphs from Gapminder, using R:

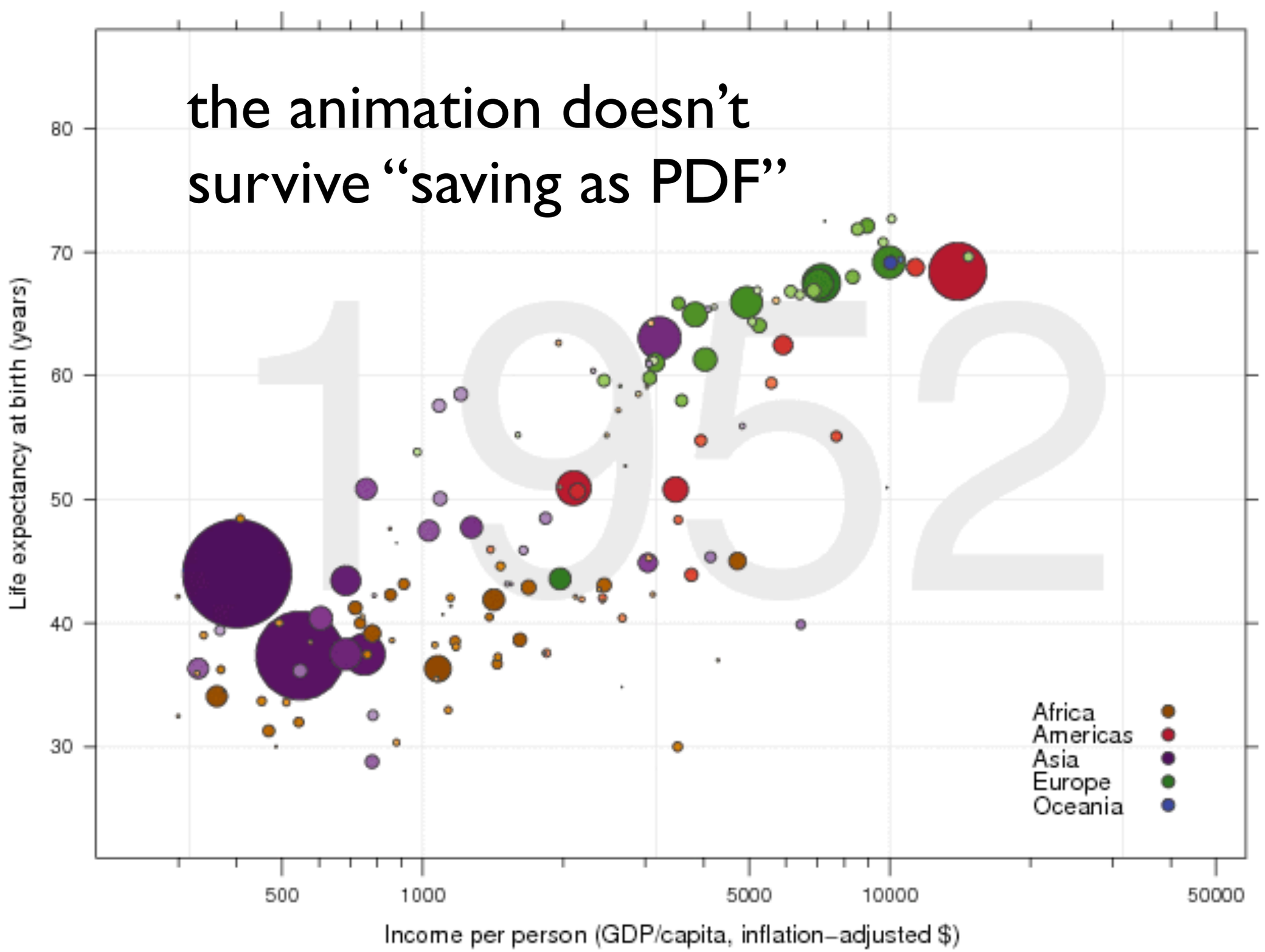
Go to Gapminder World.
Using the pull-down menu, set the x-axis for “Income per person (GDP/capita)”.
Set the y-axis for “Life expectancy at birth”.
Set the circle size (look in lower right corner) for “Population, total”.
Hit Play.

www.gapminder.org (via YouTube)

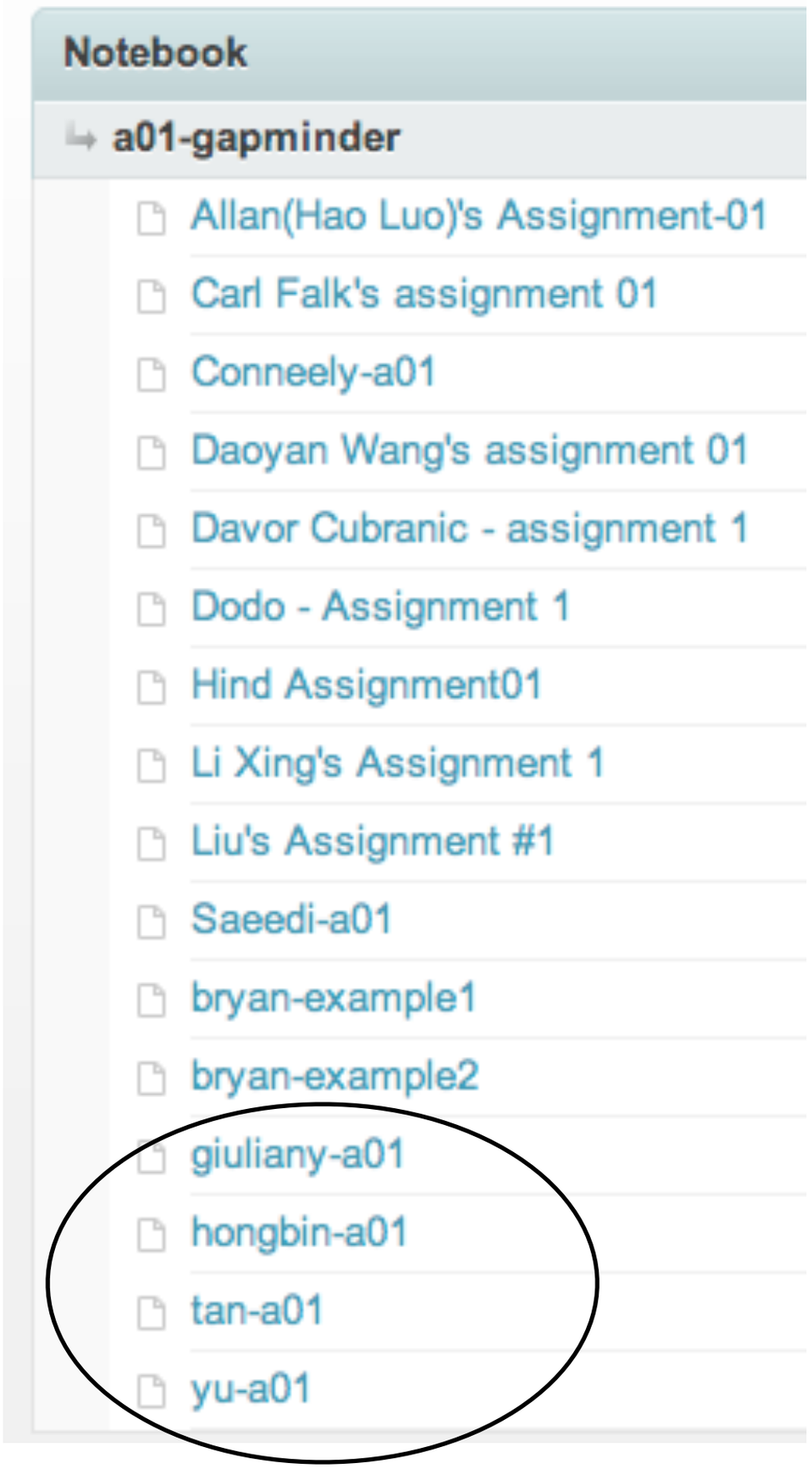


A JB base R graphics 'solution' (but I took longer than 2 hours!)

the animation doesn't survive "saving as PDF"



A JB lattice 'solution' (but I took longer than 2 hours!)



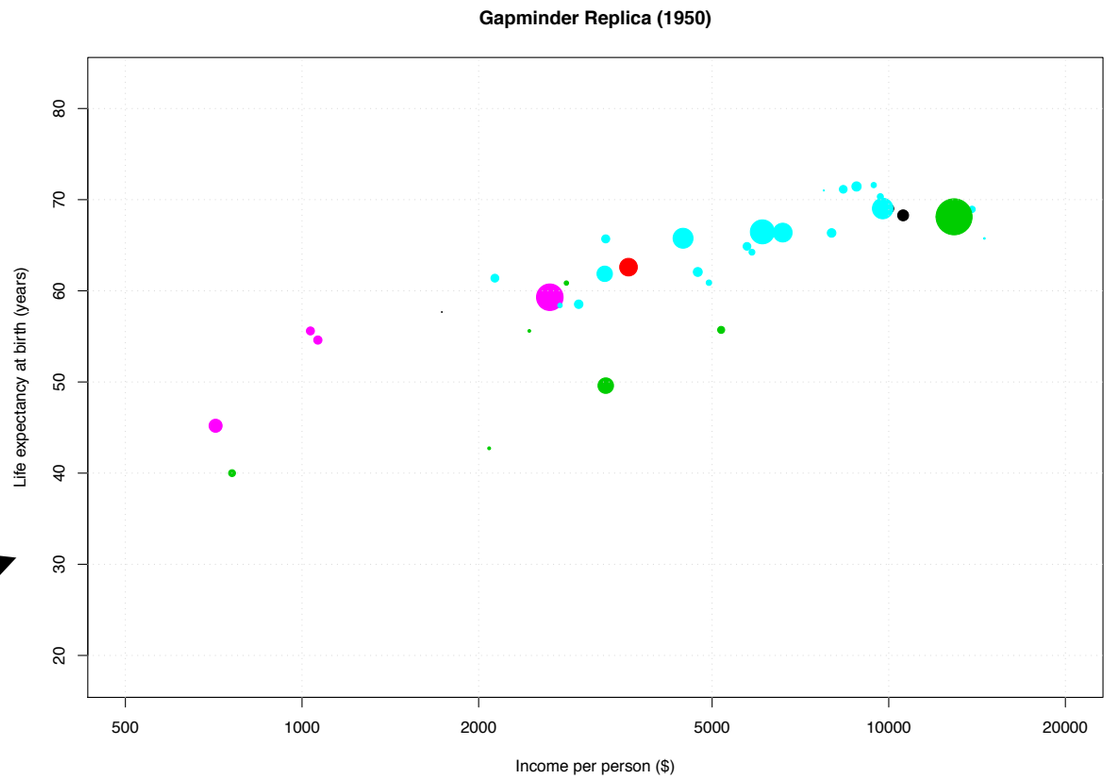
Please name the pages like so. Start practicing eliminating spaces, apostrophes, etc. from your directory and filenames. You will be a happier person.

2012 students doing well here so far!

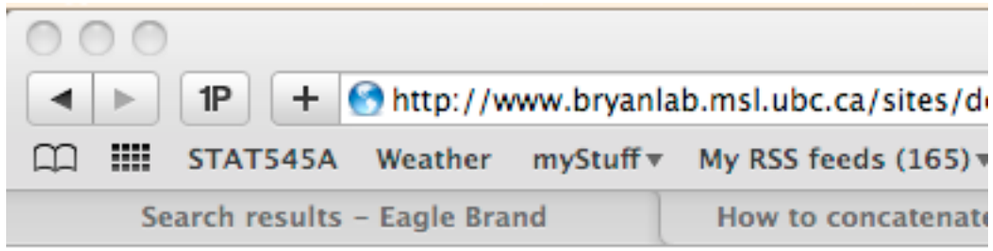
Please post/link to graphics files that are easily clicked & viewed. No Word documents.

Oh well, here's the best I could manage :).

Preview	Attachment
	giuliany-a01.pdf
	giuliany-a01.R



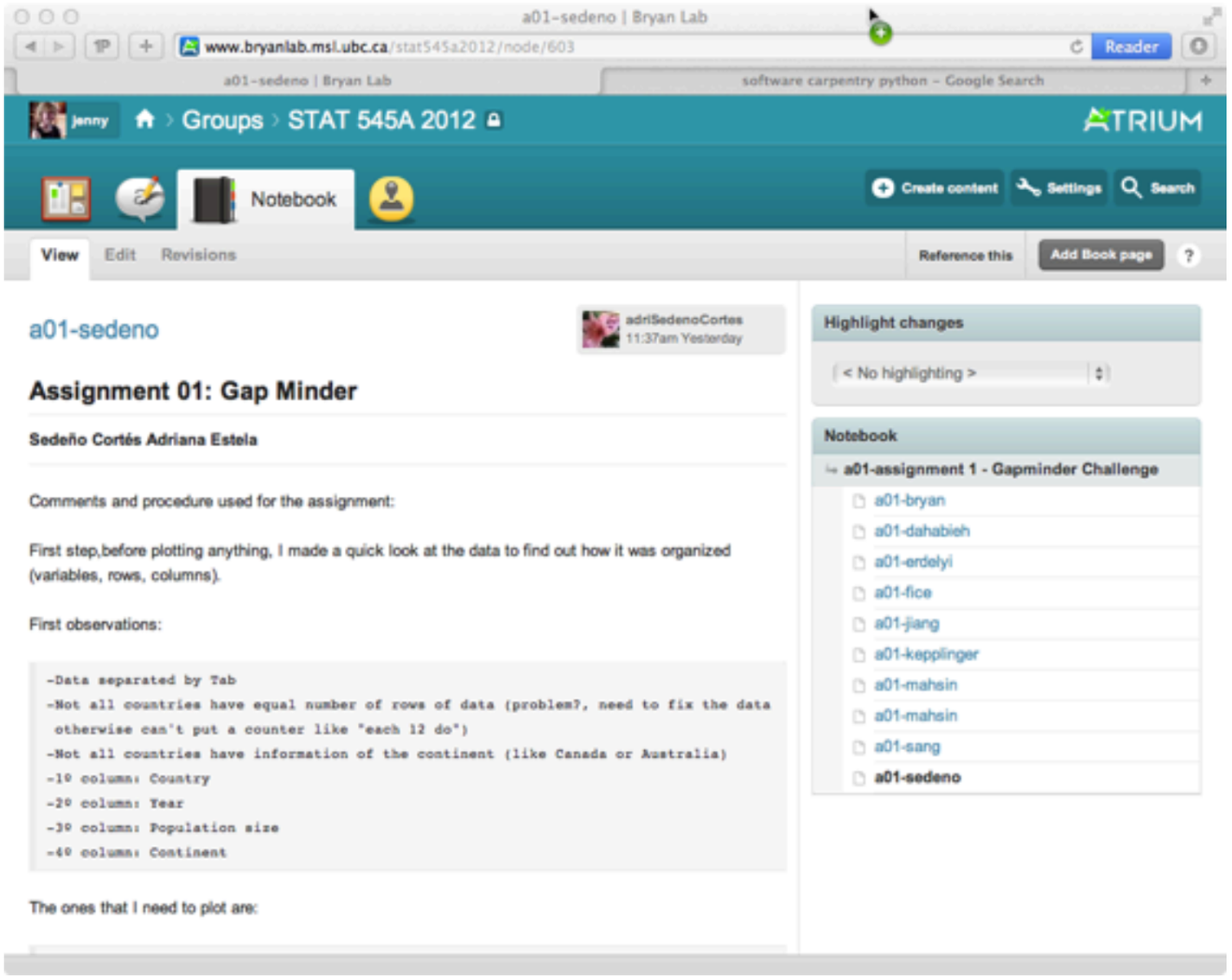
Please post/link to actual, runnable R code. Not a transcript of an R session. Not a mix of English prose and code.




```
data = read.table("C:/Users/Ryan/Documents/UBC/S
                  sep="\t", header=T)
life.exp = data$lifeExp
pop = data$pop
gdp = data$gdpPercap
year = data$year
cont = data$continent

data.1950.entries = year == 1950

life.exp.1950 = life.exp[data.1950.entries]
pop.1950 = pop[data.1950.entries]
gdp.1950 = gdp[data.1950.entries]
cont.1950 = cont[data.1950.entries]

#pulled colour codes from a a handy chart:
```

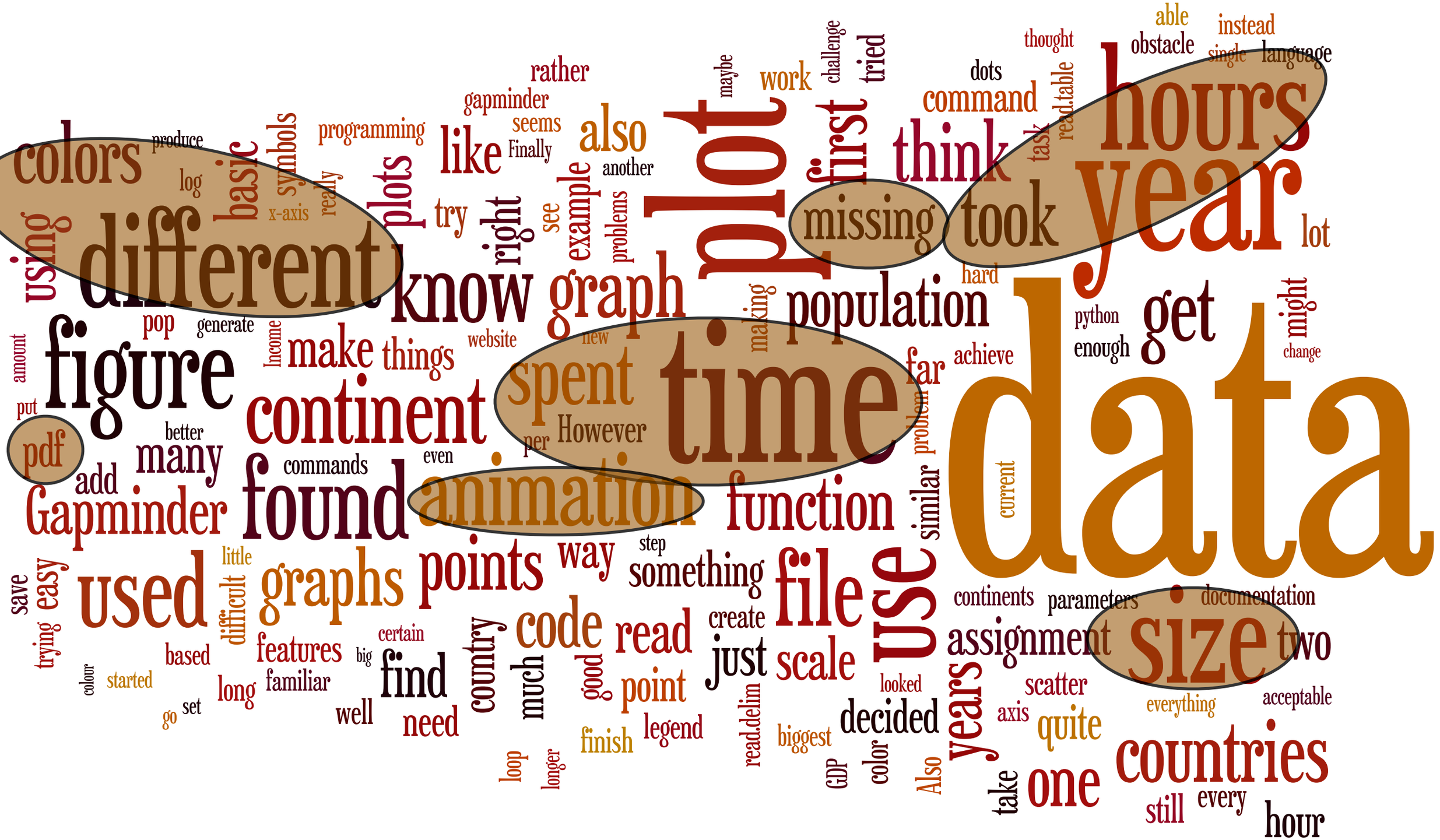
Nice job by Adri re: using the mark up and recording what she did, why, how it went. These sorts of notes are a great practice, even just for yourself!

http://www.bryanlab.msl.ubc.ca/stat545a2012/node/603



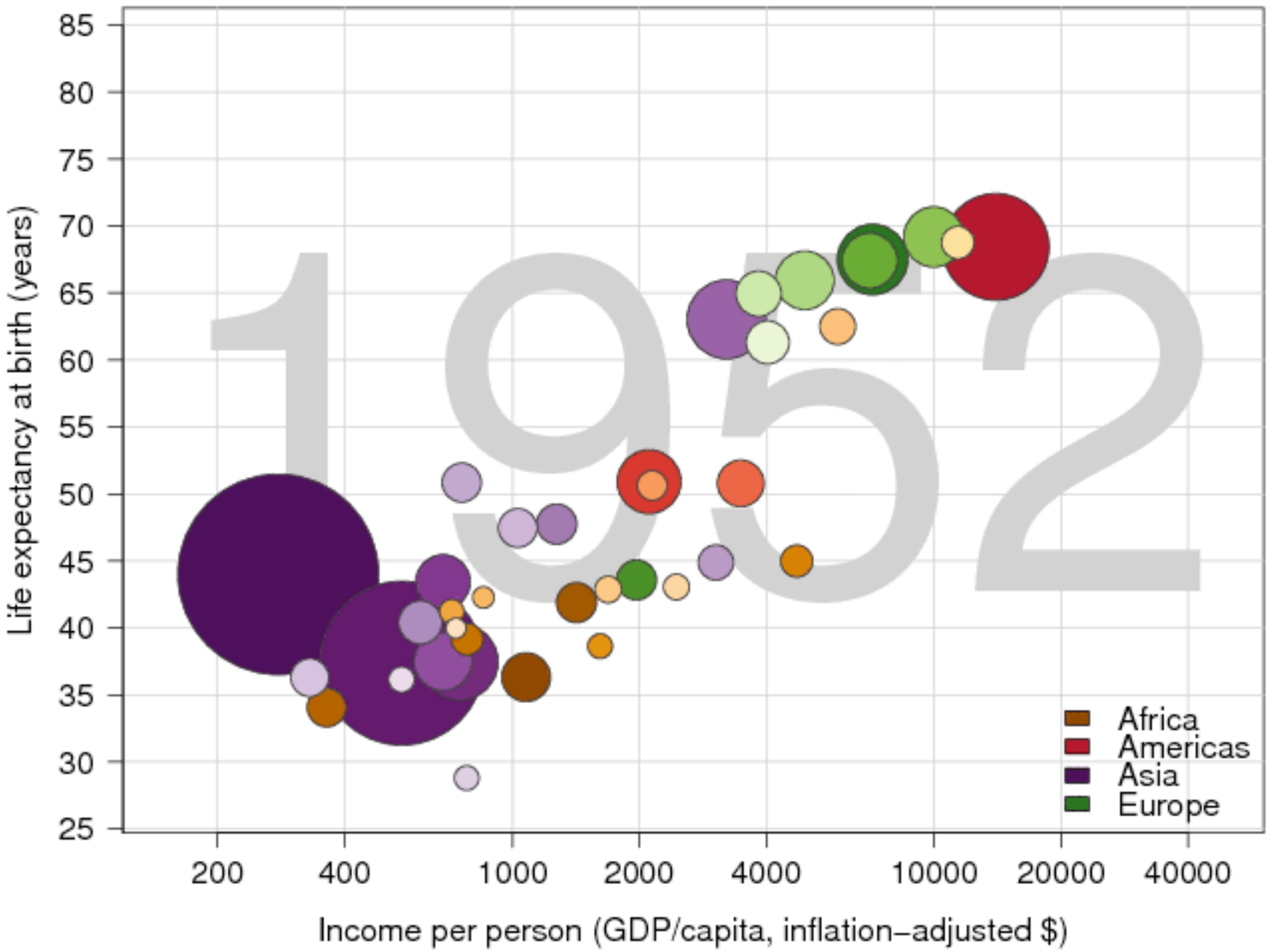
Please do post your prose in the body text. Live links, formatting, etc. are greatly appreciated by your readers. If you're not already comfortable with mark up, this is a good time to start.

wordle from the previous assignment 1 reflections

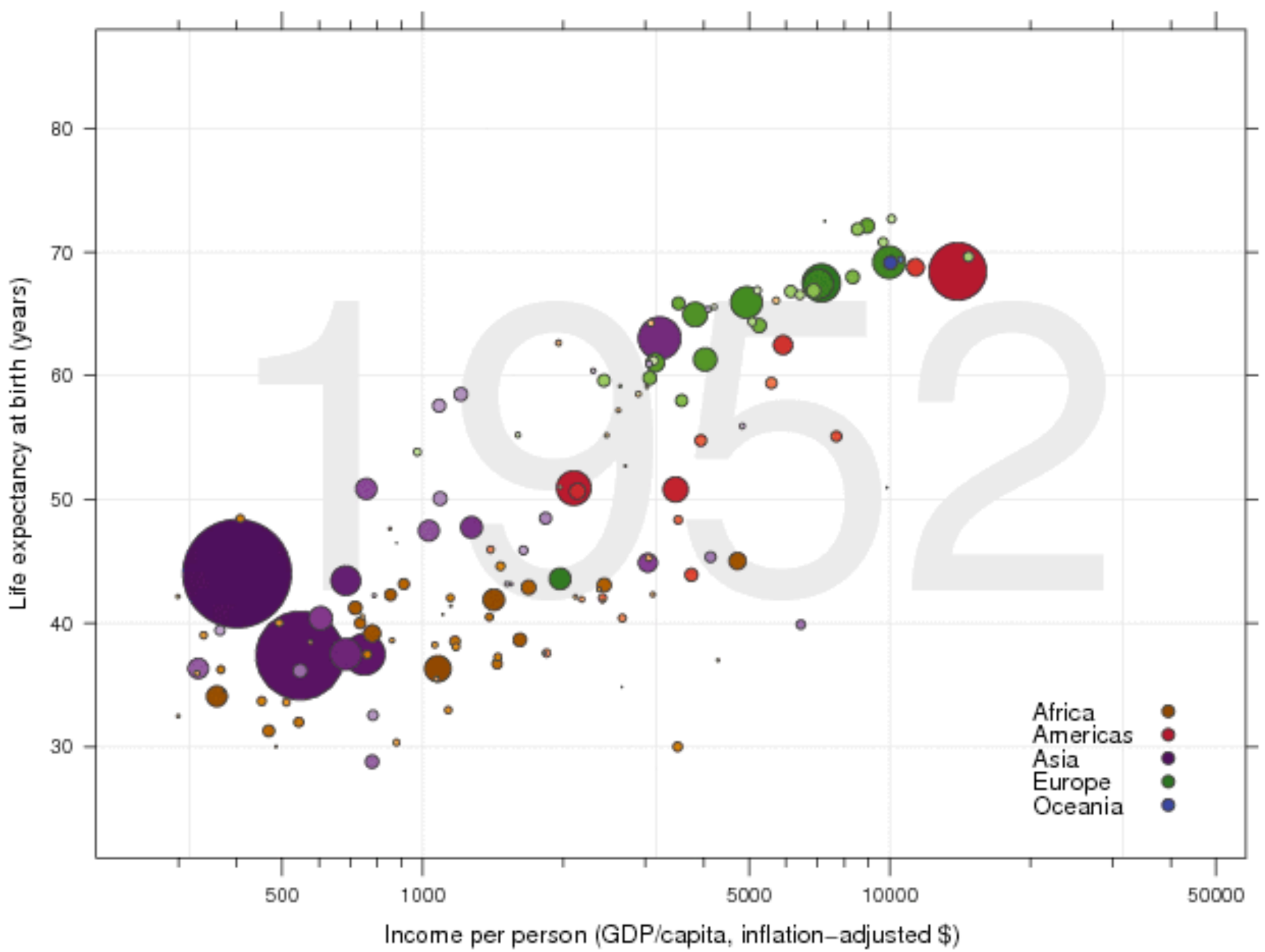


wordle from the previous assignment 1 reflections

<We discussed student work and approaches by browsing around the wiki.>



A JB ‘solution’ (but I took longer than 2 hours!)



A JB lattice ‘solution’ (but I took longer than 2 hours!)

Let’s put out some fires.

First, some informal quick tips to help people solve some very common, very frustrating problems .....

Problem: Working directory, paths, etc.

> "However, my problem is that, usually some small problems which are not about the programming itself bothered me much. For example, changing the directory of workspace, etc."

You will generally bring data into R from a file.

You will often bring information -- numerical or graphical -- out of R to a file.

You will need to communicate *where* these files live on your computer. How to handle?

"working directory"

See the working directory of an R session:

```
> getwd()
[1] "/Users/jenny/teaching/2012-2013/STAT545A/classMeet/cm02"
```

One could set the working directory like so ....

```
> setwd("/Users/jenny/tmp/")
> getwd()
[1] "/Users/jenny/tmp"
```

.... and then provide only filenames when reading / writing.

This will work BUT I take a different approach.

Alternative approach I prefer:
I always provide the specific, full path when I read or write a file.

I'll justify and demonstrate this more in a later lecture.

For now, here's an example of what I mean:

```
gDat <-
  read.delim(file = "/Users/jenny/teaching/2012-2013/STAT545A/
examples/gapminder/data/gapminderData.txt")
```

(though I achieve this differently in real life for obvious reasons ... above is ugly and cumbersome)

getting data back out of R ...

will cover later but if you are desperate, read up on ‘write.table’ for rectangular, spreadsheet-y objects and ‘save’ for other things

‘read.table’ & ‘write.table’ are related, as are ‘save’ and ‘load’

Problem: Getting data into R

```
*. at first, when I encounted that R could not read-in the data
        file, I tried to go to Gapminder website to download its data;
        however, I could not read the .xlsx formated file as I have only
        MS Office 2003 installed; I then downloaded a convertor; but after
        I see the converted data file, it is actully not usable.
```

My first roadblock came while attempting the load the data; many of the entries were missing a continent. At first, I decided to try to go through, line by line, and fill in the missing data. Having spent 15 minutes and not completed a tenth of the work, I decided that there was no way that anyone else went through this trouble.

I used the data supplied in the text file on the website by pasting it into an excel document and saving it as a .csv file. I was able to read my data and create a simple plot with little problems.

Later on I found that the data in the table are not complete. I know there are some command to run R with incomplete data, but I don not know how to use it. Finally I am happy to know read.delim() works.

some countries have a missing value for the "continent" variable,

Getting data into R

See Chapter 2 of Spector (2008).

```
> whereAmI <-
+     "/Users/jenny/teaching/2012-2013/STAT545A/examples/gapminder/"
> gDat <- read.delim(paste0(whereAmI, "data/gapminderData.txt"))
```

- read.table() -- and friends read.csv() and read.delim() -- are the functions you will use most often
- **Worth reading the documentation!** Educate yourself about the arguments -- with careful use of arguments, can often get the data.frame you want ***on import***, eliminating lots of post-import fussing around or ill-advised hand-editing of the input data.

* Caveat: I have recently discovered file.path(), which may be the more correct way to build paths, especially when writing for multiple platforms, e.g. in a package for distribution. Good to know.

```
> gDat <- read.table(paste0(whereAmI, "data/gapminderData.txt"))
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,  :
  line 62 did not have 6 elements
```

Let's look at line 62 and its neighborhood ....

	1	2	3	4	5	6
55	Argentina	1977	26983828	Americas	68.481	10079.02674
56	Argentina	1982	29341374	Americas	69.942	8997.897412
57	Argentina	1987	31620918	Americas	70.774	9139.671389
58	Argentina	1992	33958947	Americas	71.868	9308.41871
59	Argentina	1997	36203463	Americas	73.275	10967.28195
60	Argentina	2002	38331121	Americas	74.34	8797.640716
61	Argentina	2007	40301927	Americas	75.32	12779.37964
62	Armenia	1992	3378331	68.663	1442.937796	
63	Armenia	1997	3059000	70.377	1791.34719	
64	Armenia	2002	3013818	71.403	2692.304039	
65	Armenia	2007	2971650	71.965	4942.543911	
66	Aruba	1972	59461	70.941	4939.758007	
67	Aruba	1977	59412	71.83	7390.359942	
68	Aruba	1982	61569	74.116	10874.91495	

	1	2	3	4	5

From the help file on read.table()

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
                   dec = ".", row.names, col.names,
                   as.is = !stringsAsFactors,
                   na.strings = "NA", colClasses = NA, nrows = -1,
                   skip = 0, check.names = TRUE, fill = !blank.lines.skip,
                   strip.white = FALSE, blank.lines.skip = TRUE,
                   comment.char = "#",
                   allowEscapes = FALSE, flush = FALSE,
                   stringsAsFactors = default.stringsAsFactors(),
                   fileEncoding = "", encoding = "unknown")

<snip, snip>

    sep: the field separator character.  Values on each line of the
         file are separated by this character.  If 'sep = ""' (the
         default for 'read.table') the separator is 'white space',
         that is one or more spaces, tabs, newlines or carriage
         returns.
```

read.delim() is a wrapper around read.table() ... mostly just read.table() with the separator set to tab

```
read.delim(file, header = TRUE, sep = "\t", quote="\"", dec=".",
                   fill = TRUE, comment.char="", ...)
```

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
                   dec = ".", row.names, col.names,
                   as.is = !stringsAsFactors,
                   na.strings = "NA", colClasses = NA, nrows = -1,
                   skip = 0, check.names = TRUE, fill = !blank.lines.skip
                   strip.white = FALSE, blank.lines.skip = TRUE,
                   comment.char = "#",
                   allowEscapes = FALSE, flush = FALSE,
                   stringsAsFactors = default.stringsAsFactors(),
                   fileEncoding = "", encoding = "unknown")
```

```
> gDat <- read.table(paste0(whereAmI, "data/gapminderData.txt"))
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings,  :
  line 62 did not have 6 elements

> gDat <- read.table(paste0(whereAmI, "data/gapminderData.txt"), sep = "\t")

> gDat <- read.delim(paste0(whereAmI, "data/gapminderData.txt"))
```

Case in point:
With the Gapminder data, simply specifying tab as the delimiter is all it takes to go from a fatal error to perfect success. This happens a lot. Read error messages and try to work the problem. Don't shut down, freak out, guess wildly.

Great quote clip from Apollo 13 movie. Disaster begins to unfold on the space craft. After a few minutes of blaming each other, leadership emerged. Ed Harris, portraying NASA flight director Gene Krantz, said: "Let's work the problem people. Let's not make things worse by guessing."

When in doubt, choose data.frame

- read.table returns your data as a certain kind of R object: a data.frame
- kind of like an Excel spreadsheet (but not really)
- data.frame > matrix, because can handle variables of different modes (use as.data.frame, as.matrix to convert)
- data.frame > separate vectors/variables
  - data.frames prevent parallel variables from becoming 'out of sync' in terms of ordering or length
- data.frame is accepted by many functions for modeling and graphing via a 'data' argument
- data.frame is a very special *list* (in the technical R sense) that also quacks like a matrix ... offers the best of both worlds

Many data analyses revolve around the idea of a dataset, a collection of related values which can be treated as a single unit. For example, you might collect information about different companies; for each company you would have a name, an industry type, the number of employees, type of health care plans offered, etc. For each of the companies you study you would have values for each of these variables. If we store the data in a matrix, with rows representing observations and columns representing variables, it would be easy to access the data, but since the modes of the variables in a dataset will often not be the same, a matrix would force, say, numeric variables to be stored as character variables. To allow the ease of indexing that a matrix would provide while accommodating different modes, R provides the data frame. A data frame is a list with the restriction that each element of the list (the variables) must be of the same length as every other element of the list. Thus, the mode of a data frame is list, and its class is data.frame. While there is some overhead for storing data in a data frame as opposed to a matrix, data frames are the preferred method for working with "observations and variables"-style datasets

from Chapter 1 of
Spector (2008).

after a (seemingly?) successful data import, top priority is to inspect the new object ideally ... without printing the whole thing to screen

Type an object's name at the command line to have it print to screen

Tempting to do this all the time because, unlike a spreadsheet, R objects aren't staring you in the face.

```
> gDat
                country year      pop continent  lifeExp   gdpPercap
1           Afghanistan 1952  8425333      Asia 28.80100    779.4453
2           Afghanistan 1957   9240934      Asia 30.33200    820.8530
3           Afghanistan 1962  10267083      Asia 31.99700    853.1007
4           Afghanistan 1967  11537966      Asia 34.02000    836.1971

... 3000 lines of data scrolled by .... waiting, waiting .... yawn

3309           Zimbabwe 1992  10704340    Africa 60.37700    693.4208
3310           Zimbabwe 1997  11404948    Africa 46.80900    792.4500
3311           Zimbabwe 2002  11926563    Africa 39.98900    672.0386
3312           Zimbabwe 2007  12311143    Africa 43.48700    469.7093
```

BUT, except in ridiculously small examples, listing the entire object isn't a great idea.

Reach out and touch your data

```
> str(gDat)
'data.frame':  3312 obs. of  6 variables:
 $ country  : Factor w/ 187 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      : int  8425333 9240934 10267083 11537966 13079460 14880372 1288181..
 $ continent: Factor w/ 7 levels "","Africa","Americas",..: 4 4 4 4 4 4 4 4 4 ..
 $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
 $ gdpPercap: num  779 821 853 836 740 ...
```

- 'str' gives a concise overview of an object
  - so named because it aims to reveal internal *str*ucture of an object
- What to look for
  - Are the mode and size what I expect?
  - For data.frames, are the constituent variables what I expect / want?
  - Are the first few observations plausible?

Reach out and touch your data

```
> head(gDat)
      country year      pop continent lifeExp gdpPercap
1 Afghanistan 1952  8425333      Asia  28.801  779.4453
2 Afghanistan 1957  9240934      Asia  30.332  820.8530
3 Afghanistan 1962 10267083      Asia  31.997  853.1007
4 Afghanistan 1967 11537966      Asia  34.020  836.1971
5 Afghanistan 1972 13079460      Asia  36.088  739.9811
6 Afghanistan 1977 14880372      Asia  38.438  786.1134
```

- ‘head’ shows the first bits of an object
  - so named because it’s like the Unix command

- ‘tail’ exists too and shows the last bits

```
> tail(gDat)
      country year      pop continent lifeExp gdpPercap
3307 Zimbabwe 1982  7636524    Africa  60.363  788.8550
3308 Zimbabwe 1987  9216418    Africa  62.351  706.1573
3309 Zimbabwe 1992 10704340    Africa  60.377  693.4208
3310 Zimbabwe 1997 11404948    Africa  46.809  792.4500
3311 Zimbabwe 2002 11926563    Africa  39.989  672.0386
3312 Zimbabwe 2007 12311143    Africa  43.487  469.7093
```

I wrote peek(), inspired by head() and tail(), because the funny stuff never seems to happen in the first or last 5 observations. Better to look at some randomly drawn observations.

```
> peek(gDat)
            country year       pop continent lifeExp  gdpPercap
70            Aruba 1997     68341                73.011 26483.6686
197         Bahamas 1997    281577                68.472 20990.8325
1767      Lithuania 2001   3645747       FSU      71.740 10272.3655
1866      Maldives  1972    122681                51.440   719.3214
2187      Nigeria   1962  41871351    Africa      39.360  1150.9275
2794      Sri Lanka 1997  18698655      Asia      70.457  2664.4773
3214 United States 2003 290342554  Americas      77.470 39747.3764
```

```
peek <- function(x, n = 7) {
  if(is.matrix(x) | is.data.frame(x)) {
    nX <- nrow(x)
    print(x[sort(sample(nX, size = n)),])
  } else {
    cat("'peek' only anticipates matrices and data.frames.\n")
  }
}
```

... the funny stuff ...

Case in point: this peek happened to alert me to the fact that the continent data is missing sometimes.

If I hadn't learned that earlier at import, I would learn that now.

Practice Defensive Coding.

```
> peek(gDat)
            country year      pop continent lifeExp  gdpPercap
70            Aruba 1997    68341           73.011 26483.6686
197         Bahamas 1997   281577           68.472 20990.8325
1767      Lithuania 2001  3645747       FSU 71.740 10272.3655
1866       Maldives 1972   122681           51.440   719.3214
2187        Nigeria 1962 41871351    Africa 39.360  1150.9275
2794      Sri Lanka 1997 18698655      Asia 70.457  2664.4773
3214 United States 2003 290342554  Americas 77.470 39747.3764
```

See wikipedia page for "Defensive programming" or "Good Programming Practices in Healthcare Creating Robust Programs," a conference report that is mostly about SAS but is specific to (bio)statistical analysis -- I especially like the rules listed on pages 3 and 4.

```
> summary(gDat)
        country           year         pop                continent
Czech Republic:  58   Min.   :1950   Min.   :5.941e+04            : 301
Denmark        :  58   1st Qu.:1967   1st Qu.:2.679e+06   Africa   : 613
Finland        :  58   Median :1982   Median :7.557e+06   Americas: 343
Iceland        :  58   Mean   :1980   Mean   :3.161e+07   Asia     : 557
Japan          :  58   3rd Qu.:1996   3rd Qu.:1.959e+07   Europe   :1302
Netherlands    :  58   Max.   :2007   Max.   :1.319e+09   FSU      : 122
(Other)        :2964                                      Oceania  :  74
   lifeExp        gdpPercap
Min.   :23.60   Min.   :   241.2
1st Qu.:58.34   1st Qu.:  2514.6
Median :69.61   Median :  7838.5
Mean   :65.25   Mean   : 11317.1
3rd Qu.:73.66   3rd Qu.: 17357.9
Max.   :82.67   Max.   :113523.1
```

‘summary’ is another good option that generally does something intelligent and informative for most objects

‘summary’ marks a transition from sanity checking to data exploration ...

Reach out and touch -- but do not print to screen - your data

```
str()
head()
tail()
peek() -- not built-in

summary()
```

now we look at getting information back out of R -- specifically saving a figure

Problem: Getting figures out of R

In an attempt to save face, I won't go into the amount of time it took me to figure out how to save something as a pdf

The next issue that I faced was figuring out how to save each individual graph as a pdf with a different file name. After much researching, I finally came across the paste function and I was able to save each file automatically within my loop. I had to manually combine the pdf files into one file using Adobe Acrobat.

Output I found pdf() and dev.off() to be useful,

(Getting ready to) make a figure

- First draft (and the next ~100 attempts!) should be displayed in a window; for example, in a X11() or quartz() device
- When ready to preserve for posterity, I advise you default to PDF. Most generally useful format to have around.
  - Educate yourself about vector versus raster graphics here (or Google it yourself). Short version: default to *vector*, which PDF is an example of.
- Save the figure w/ a line of R code, not with the mouse.
- If preparing a figure for the web and/or a figure that presents a very large number of data points -- png() is a good option to consider. The web winds are blowing towards SVG these days.

(Getting ready to) make a figure

- If you rigorously produce your figures from code you save -- versus via typing at the command line or (shudder) with a mouse -- it will always be easy to remake a figure natively on a different device. This will come up!

- Additionally, it will be easy to find the R code that generated any figure by searching for the figure filename on your computer, filtering for *.R files if possible.

- The following methods for writing to PDF should work for other devices.

Two good blog posts on saving R graphics

10 tips for making your R graphics look their best
Revolutions blog post
http://blog.revolutionanalytics.com/2009/01/10-tips-for-making-your-r-graphics-look-their-best.html <-- highly recommended

High-quality R graphics on the Web with SVG
Revolutions blog post
http://blog.revolutionanalytics.com/2011/07/r-svg-graphics.html

Save a figure -- Method 1

- Open a PDF device by calling pdf(), execute all your graphics commands, close the PDF by calling dev.off()
  - Pros: Most 'correct' method. Possible to create multi-page files.
  - Cons: pdf() and dev.off() commands clutter up your R file and require repeated (de-)commenting out. It's annoying to re-submit all those commands, when you're staring at a beautiful figure on the screen.

```
pdf(paste0(whereAmI,"figs/rawPlotsByORF.pdf"),
    width = 9.5, height = 7)
for(i in seq(along = levels(kDat$ORF)))
   print(xyplot(pheno ~ tm | day, kDat,<blah, blah, ...>)
dev.off()
```

Save a figure -- Method 2

- Make the figure in a screen device, such as X11(). Call dev.print to copy that to a PDF file.
  - Pros: Immediate gratification. PDF-generating code is limited to one command, so easier to (de-)comment out.
  - Cons: Slightly 'incorrect'. Certain aspects of the figure depend on the graphics device (which would be X11, not PDF), therefore, in some cases, you can get different PDF files with Methods 1 and 2. It's a risk I'm often willing to take. Also, won't work for multi-page figures.

```
xyplot(lifeExp ~ gdpPercap | year, gDat)
dev.print(pdf,
          file = paste0(whereAmI,"figs/bryan-a01-latticeMinimal.pdf"),
          width = 9, height = 6)
```

Strategies for tackling difficult tasks, getting unstuck

break the task down into small pieces

walk before you run

make stuff up -- worry about dotting your i's and crossing your t's later

set aside the weird cases and just work with the "pretty" data at first

I decided to create a series of graphs (one for each year) which look as similar as possible to the gapminder graphics.

I was able to create scatterplot of a single year with most of the required features pretty quickly (maybe an hour), because I relied on parts of R that I already knew: reading a data frame from a file, filtering rows based on some condition (i.e., year), and drawing scatterplots. (Because basic R graphics could do everything I needed for the assignment, I didn't try using Lattice or GGPlot.)

I initially took a couple of shortcuts: I only drew a single year, did not colour points based on the country's continent, and used a random size of dots instead of making it proportional to population. Also, I started with linear instead of log scale for the GDP axis.

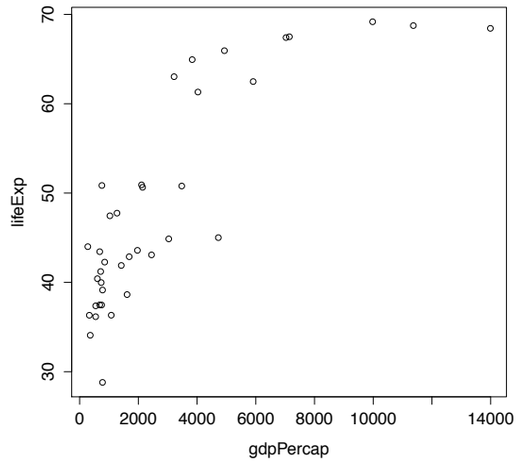
I then started adding the missing bits, and each took longer than expected..... In the end, the 'missing bits' that I avoided in my first pass at the code took longer than the basic functionality

Splitting up the task into components worked well to keep me focused instead of trying to think about everything at once. For instance, I started just graphing one year's worth of data and working on that first. Then, I tried to get different colors for data from each continent, change the size/shape of each data point, add a legend, etc.

```
gDat <- read.delim(file = paste0(whereAmI,
                                 "data/gapminderDataBiggest40.txt"))
jYear <- 1952
tinyDat <- subset(gDat, year == jYear)
plot(lifeExp ~ gdpPercap, tinyDat)
dev.print(pdf, paste0(whereAmI, "figs/walkingBeforeRunning.pdf"))
```

‘subset’ is generally the best way to filter a data.frame (or even other objects), i.e. to retain only certain rows or certain variables

another function worth reading the documentation for



subset()

Specific rows can be selected via subset argument which expects a logical vector

```
gDat <- read.delim(file = paste0(whereAmI,
                                 "data/gapminderDataBiggest40.txt"))
jYear <- 1952
tinyDat <- subset(gDat, year == jYear)

....

gDat <- subset(gDat, subset = year %% 5 == 2)
```

Specific variables (or columns) can also be retained (or dropped) via select argument which expects an expression

Advantages to ‘manual’ subsetting:

- code cleaner, more readable
- smart about looking for variables within the data.frame you’re operating on
- better handling of NAs (?)

filtering with logical vectors arising from regular expressions

```
> ## these accomplish the same thing

> yDat <-
+   subset(gDat,
+          subset = (year %in% jYears &
+                    grepl('^R[ow]',country)),
+          select = c('country','lifeExp','pop'))

> yDat <- gDat[(gDat$year %in% jYears &
+               grepl('^R[ow]',gDat$country)),
+              c('country','lifeExp','pop')]
```

keeping only variables 'country', 'lifeExp', and 'pop'

keeping only rows for countries starting with 'Ro' or 'Rw'

```
> str(yDat)
'data.frame':    16 obs. of  3 variables:
 $ country: Factor w/ 167 levels "Afghanistan",..: 127 127 ...
 $ lifeExp: num  61 64.1 66.8 66.8 69.2 ...
 $ pop    : int  16630000 17829327 18680721 19284814 ...

> yDat
     country lifeExp      pop
1494 Romania  61.050 16630000
1495 Romania  64.100 17829327
...
1502  Rwanda  40.000  2534927
1503  Rwanda  41.500  2822082
...
```

```
gDat <- read.delim(file = jPaste(whereAmI,
                                 "data/gapminderDataBiggest40.txt"))
jYear <- 1952
tinyDat <- subset(gDat, year == jYear)
xyplot(lifeExp ~ gdpPercap, tinyDat)
dev.print(pdf, jPaste(whereAmI, "figs/walkingBeforeRunning.pdf"))
```

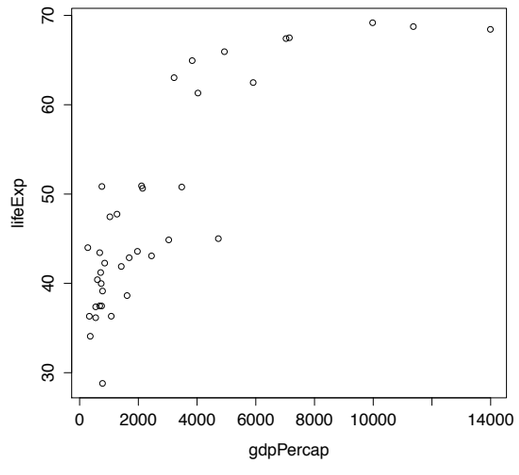
Don't hard-wire your small test cases.

Make it easy to change your test case.

Change it as you make progress.

You will avoid over-tuning.

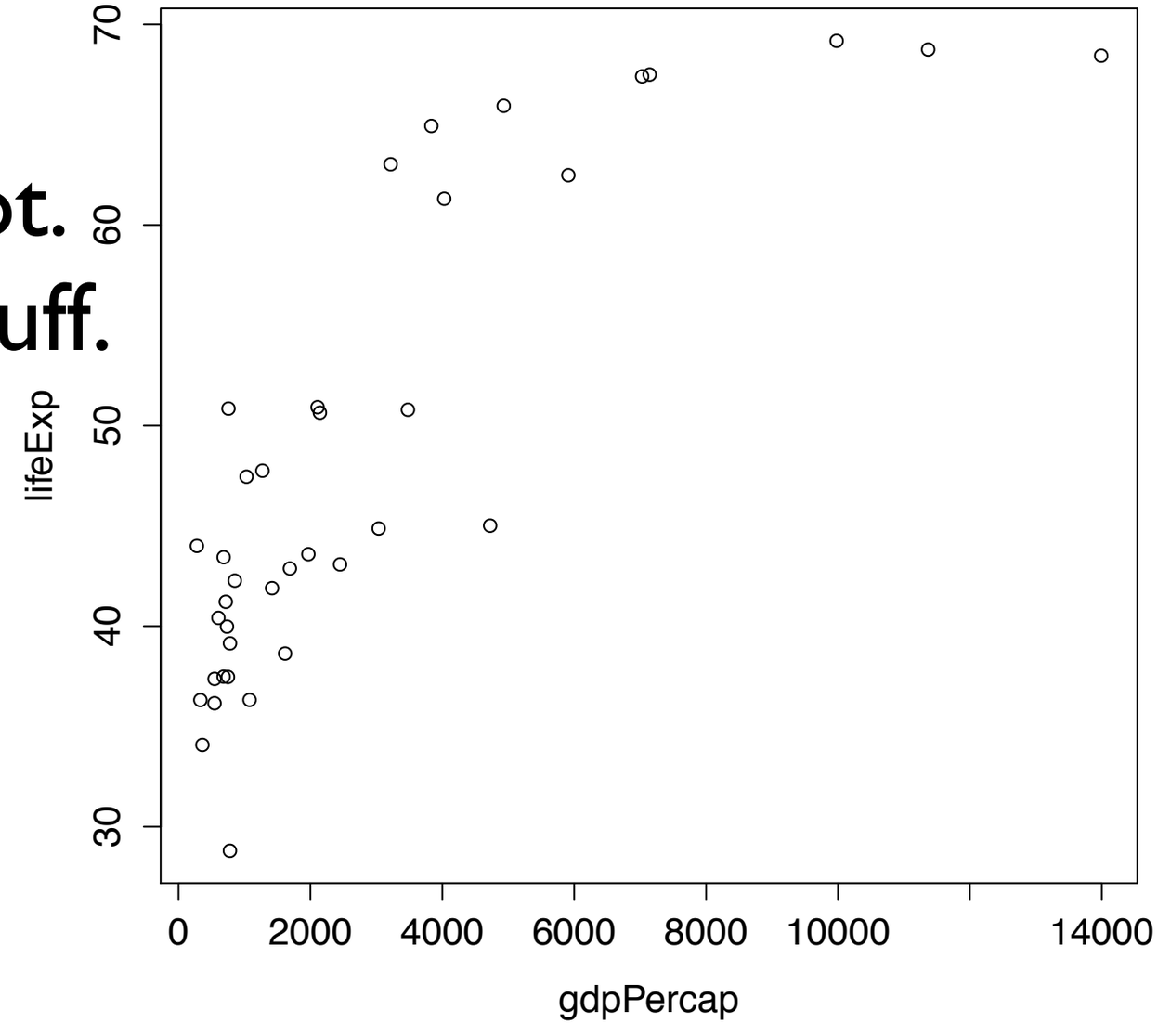
Naturally leads to scaling up / looping, building a function to make the plot for one year, etc.



```
gDat <- read.delim(file = jPaste(whereAmI,
                                 "data/gapminderDataBiggest40.txt"))
jYear <- 1952
tinyDat <- subset(gDat, year == jYear)
xyplot(lifeExp ~ gdpPercap, tinyDat)
dev.print(pdf, jPaste(whereAmI, "figs/walkingBeforeRunning.pdf"))
```

Start with the simplest plot.
Gradually add the fancy stuff.

Walk before you run.



less is more

not every country has data for each year

In some specific year, we have fewer countries involved in the survey

Also the Gapminder graphs seem to interpolate data for missing years, which seems like it would be troublesome to do properly (an average of the two closest years wouldn't be too hard maybe?).

Next, drawing all years caused another surprise about the data: not all countries have entries for each year! Manually checking the values showed that Africa is particularly bad, and only appears every fifth year. I then decided to plot only those years to avoid having a big chunk of dots appear every few pages and then disappear again.

```
gDat <- subset(gDat, subset = year %% 5 == 2)
```

Don't feel obligated to show every last piece of data you have.

But try to remain true to the full data, i.e. don't hide anything or warp the truth.

Figs that showed only one year tended to be more effective than figs that superposed all years or series where most data points disappeared / reappeared incessantly.

Other common issues -- will be covered

How to scale up, loop over many years

coloring by country/ continent

legend

logging an axis, axis limits, tick marks, reference grid

controlling the relationship between population size and the circle

Mini-Assignment #2:

Spend ~ 1 hour by end of this Friday September 7 making improvements to your Gapminder effort.

Tackle the most vexing issue you had, use class notes or other students' work for guidance (but give credit, especially if another student has helped you), etc.

JB has created a notebook page for this now:
a02-assignment 2 - Gapminder Resolution

Step 1 of Assignment #3:

Get an R-aware editor installed.

Just a preview!
Will be tweaked once I see how many using Rstudio.

R-aware = can send code to a running R process.*

Fire up R and the editor and get the system working.

(You will be glad you did. Trust me.)

*Or, at a very bare minimum, will help you enforce a coherent coding style on your R code, will do syntactical highlighting, etc..

Obsolete due to Rstudio?

Good leads for editors

- Emacs Speaks Statistics (what I use); also see the "R and Emacs" section of the R FAQ
- Specific Emacs installation I use (Mac OS X), includes ESS
- Vincent Goulet has a nice version of Emacs for Windows that includes ESS, AuCTeX, aspell.
- TextMate + R bundles (Mac)
- Tinn-R (Windows)
- WinEdt + R-WinEdt (helpful link?)
- vim, an improved version of vi, + R.Vim (helpful link?)
- NppToR: R Syntax Highlighting, Code Folding and Code-Passing for R in Notepad++

Obsolete due to Rstudio?

More links re: R & editors -- maybe more recent / fresh / current?

- http://www.sciviews.org/_rgui/projects/Editors.html
- http://stackoverflow.com/questions/1783254/r-text-editors-for-introductory-statistics-courses
- http://stackoverflow.com/questions/1439059/best-ide-texteditor-for-r
- http://www.rstudio.org

Just a preview!
I will give some prompts for this, i.e. be more concrete about what "start thinking" should mean.

Step 2 of Assignment #3:

Start thinking about a dataset you really want to analyze -- especially through a series of awesome figures.

Try to get it and clean it and make your first rough plots.

Do you need help w/ R stuff? Do you need help w/ ideas? Do you want to recruit others? It's never too early to start!