an introduction to R for epidemiologists

the basics

Charles DiMaggio, PhD, MPH, PA-C

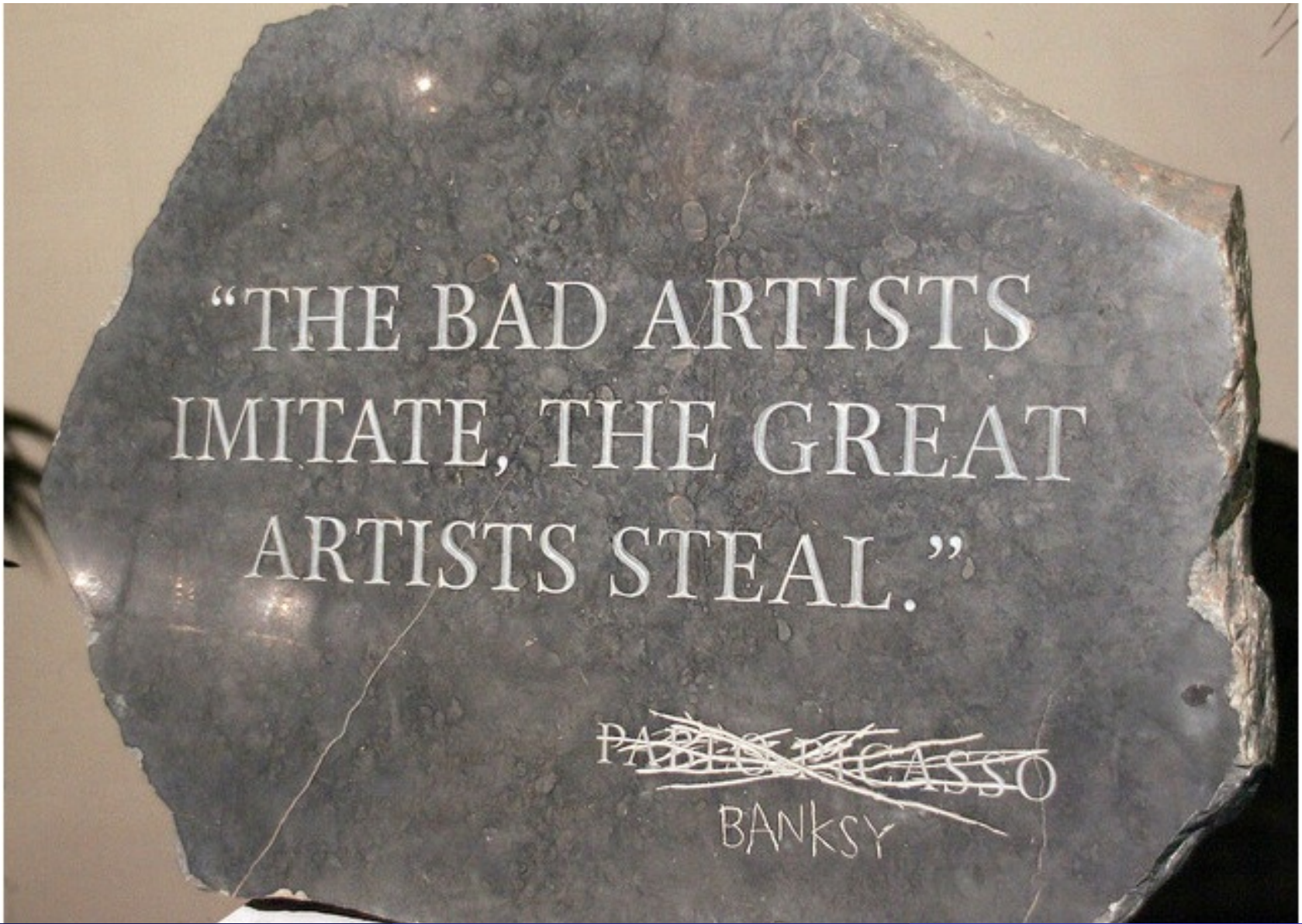
New York University Departments of Surgery and Population Health
NYU-Bellevue Division of Trauma and Surgical Critical Care
550 First Avenue, New York, NY 10016

Spring 2015

- http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/
- Charles.DiMaggio@nyumc.org

Outline

But first...



Credit where credit is due...

- Tomas Aragon, MD, DrPH
  - Applied Epidemiology Using R
  - `http://medepi.com/`
- John Fox, PhD
  - An Introduction to Statistical Computing in R
  - `http://socserv.mcmaster.ca/jfox/Courses/UCLA/index.html`
- Bill Venebles, PhD
  - An Introduction to R
  - `cran.r-project.org/doc/manuals/R-intro.pdf`
- Phil Spector, PhD
  - Data Manipulation with R

Outline

R

what is R?

A flexible, scalable, **free** tool for the description, analysis, visual display, exploration and interpretation of data.

- a calculator
- a suite of statistical tools
- a graphics creator
- a programming language
- a simulation lab
- a means of scientific documentation and discourse

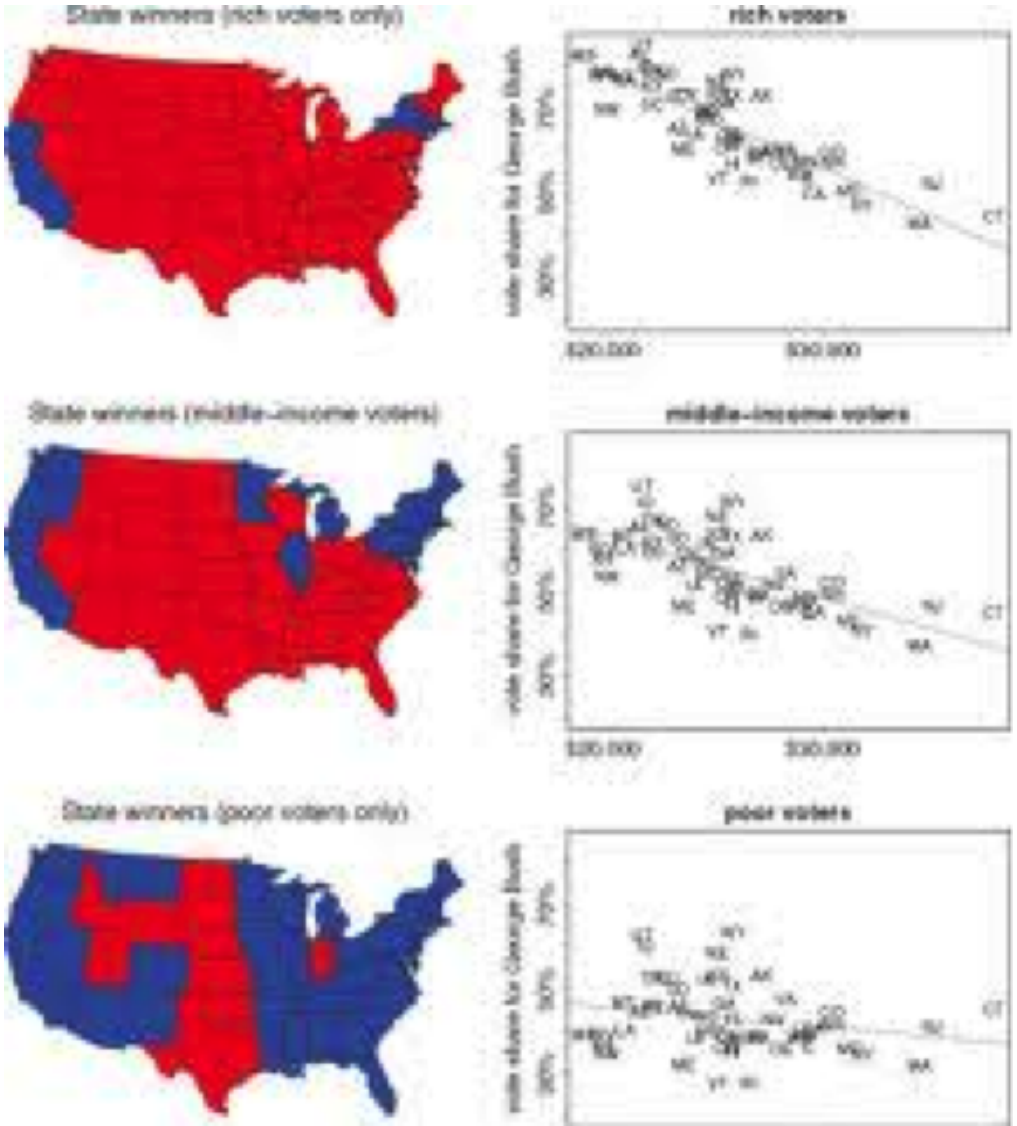
It is uniquely suited to epidemiological analysis.

Making stunning graphics

Paul Butler

Predicting Elections

Andrew Gelman, "Red State, Blue State, Rich State, Poor State"



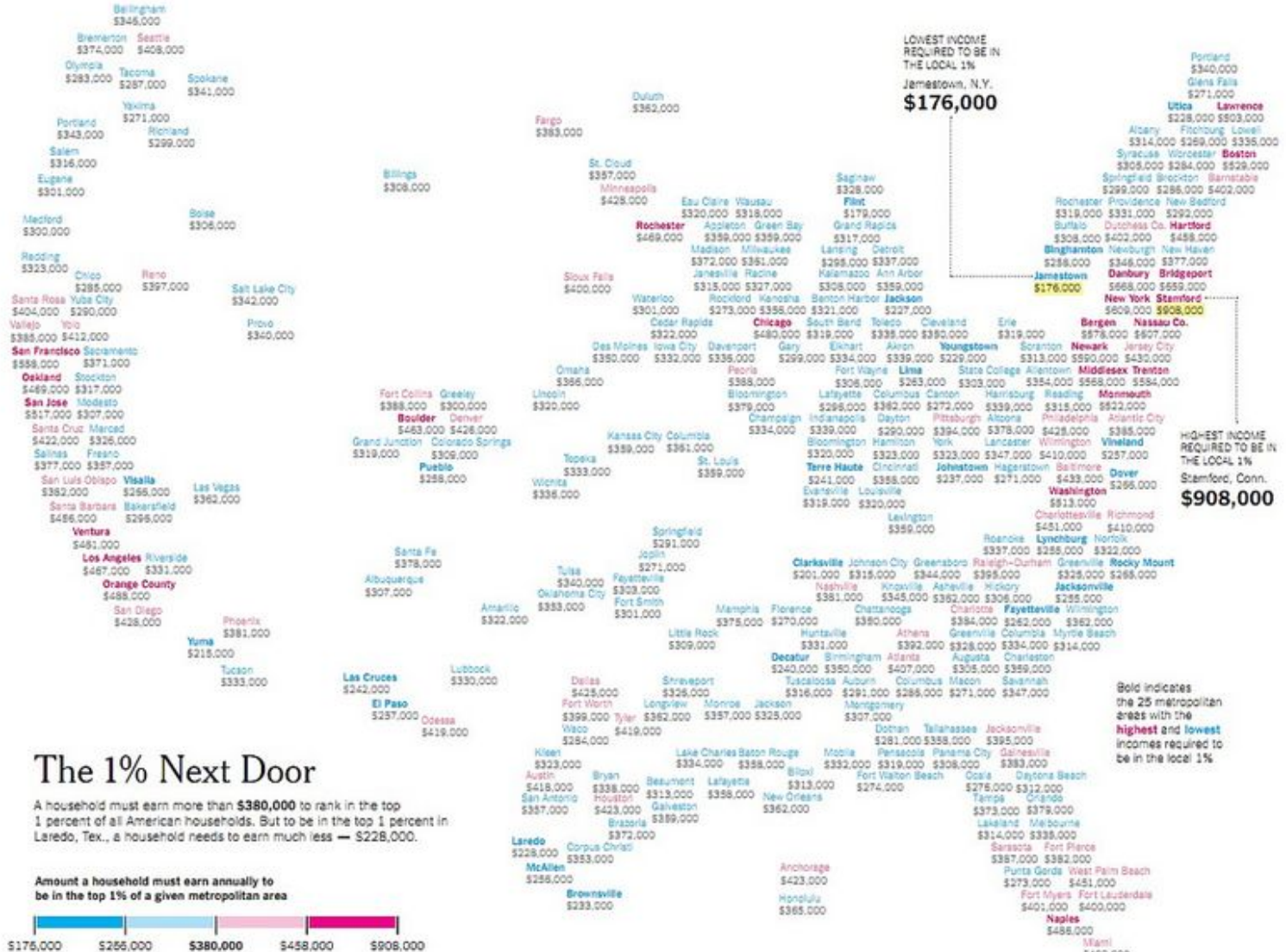
Crawling and Scraping the Web

John Muschelli, Andrew Jaffe, Jeffrey Leek. Simply Statistics Blog

Making Money

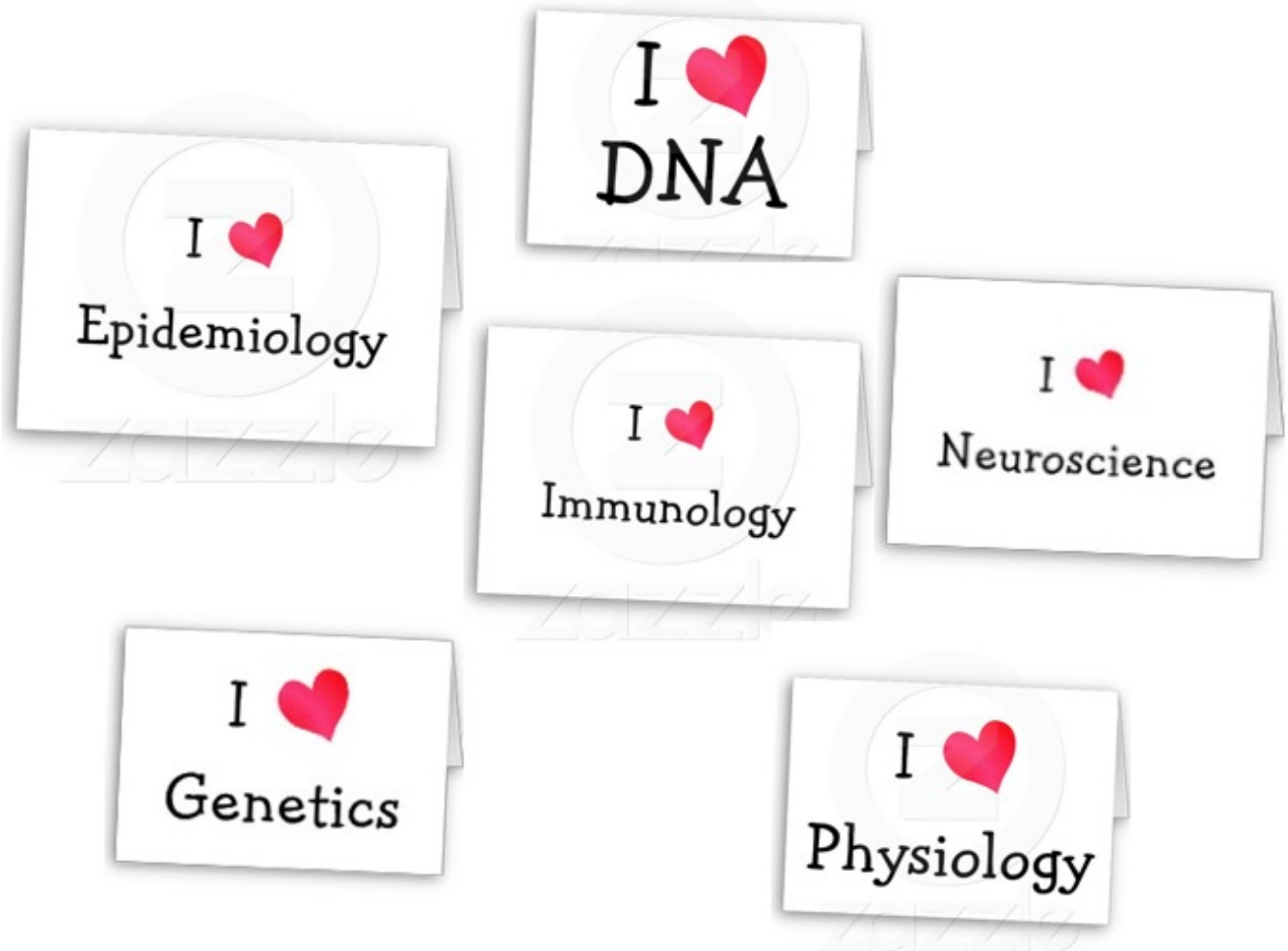
Publishing Newspapers

New York Times is an R Shop



Doing Science

Developed by Scientists for Scientists



so, why learn R?

Many options for epidemiological computing: SAS, STATA, SPSS, Mathematica, Excel....

But, if you want to ...

- accomplish many tasks with a single tool
- better understand the methods you use
- use methods not available in any other program
- develop and share your own methods
- collaborate with wide community of scientific colleagues

...R might be for you.

and did I mention it's free?

what R is not

- a GUI experience
- initially easy and intuitive
- warrantied in any way (if it runs, it can be on CRAN)
- a DBMS
- (*traditionally*) well suited to enormous data sets (*but that is changing...*)
  - historical 32-bit limit ($2^{31} - 1$) on size of a vector
  - in R, objects like matrices are actually vectors
  - R stores everything in RAM
  - old rule of thumb: 100,000 rows, 20 variables (*very* conservative), single object 10% of RAM, etc...
  - but, things have changed
    - 64 bit computing, increased RAM
    - interest in 'big data' in R
      `http://www.cybaea.net/Blogs/Data/Big-data-for-R.html`
    - packages like *optimize read.csv, ff, bigmemory, data.table, rsqlite, python* take advantage of "paralellism" (Hadoop, MapReduce)

how to install R

1. go to `http://www.r-project.org/`
2. select CRAN (Comprehensive R Archive Network) from left menu
3. link to nearby geographic site (e.g., http://software.rc.fas.harvard.edu/mirrors/R/)
4. select your operating system
5. chose "Base" installation
6. save R-X.X.X-win32.exe (windows) or R-X.X.X-mini.dmg (Mac OS X)
7. run the installation program accepting defaults

Outline

R is a calculator

math operators and functions

```
arithmetic    + , - , * , /
power         ^
```

convert 68 degrees Fahrenheit to Celsius ($C^0 = \frac{5}{9}(F^0 - 32)$)

```
5/9*(68-32)
```

assignment operator

'memory' key

<-

```
y <- 5^3 #assignment (no display )
y
(y <- 5^3) #assignment (display)
```

concatenation *function*

combine or "vectorize"

c()

,

```
x <- c(1,2,3,4,5)
x
y<-c("a", "b", "c", "d")
y
```

functions

R "apps"

fx()

math operators and functions

```
mathematical functions -  sqrt, log, exp, sin, cos, tan
simple functions -   max, min, length, sum, mean, var, sort
```

```
abs(-23) #absolute value
exp(8) # exponentiation
log(exp(8)) # natural logarithm
sqrt(64) # square root
```


YOUR
TURN

- create a vector object called "my.numbers" that consists of the numbers 2,4,6 and 8.
- what is the square root of the sum of "my.numbers"?

about R functions

and their arguments

- function name without parentheses returns source code
  - useful if want to write own code or functions
- *args(function)* returns brief argument syntax
- some arguments have default values
  - if entered in correct order need not be named

```
args(sample)
data<-1:30
sample(s = 18, x = data, r = T)
 # with replacement (spell enough to identify)
sample(s = sample(1:100, 1), x = sample(1:10, 5), r=T)
# arg any valid R expression
```

write your own function

R is a programming language

```
 my.function<-function(x){
5/9*(x-32)
}
my.function(68)
[1] 20

a<-c(134,156,222)
my.function(a)
[1]  56.66667  68.88889 105.55556
```

base R comes with many handy statistical functions

summary statistics

- summary(), fivenum(), stem() - examine the distribution of a data set
- qqnorm(), qqline() normal plots
- boxplots() (a, b)

test statistics

- t.test() 2-sample t test, (a, b),
  - R does not by default assume equality of variances, (can use an F test to examine this assumption)
  - var.test() returns an F test, (a,b)
- wilcox.test() returns a two-sample non-parametric Wilcoxon (aka Mann-Whitney) or one-sample Wilcoxon (specify "paired=TRUE") test

Some statistics with R

```
myDat<-data.frame(cbind(outcome1=rnorm(1000,20,5),
outcome2=rpois(1000,5),
grp=factor(sample(c("a","b","c"), 1000, replace=T))))

summary(myDat$outcome1)
fivenum(myDat$outcome1)
stem(myDat$outcome1)

boxplot(myDat)
boxplot(outcome1~grp, data=myDat)

myDat2<-cbind(rnorm(1000,20,5), rpois(1000,5))
boxplot(myDat2)

qqnorm(myDat$outcome1)
qqline(myDat$outcome1)

t.test(myDat$outcome1, myDat$outcome2)

wilcox.test(myDat$outcome1, myDat$outcome2)
wilcox.test(myDat$outcome1, myDat$outcome2, paired=T)
```


YOUR
TURN

- run the following code to create a small data set:

```
crashDat<-data.frame(age=rnorm(n=100,mean=22, sd=2),
crash=sample(x=c(0,1),size=100,replace=T, prob=c(.2,.8)))
```

- what is the mean age
- create a box plot comparing age by crash status

Outline

R packages

a quick example

- collections of user-written functions
- *install.package("my.package")* - copies the package from CRAN to your installation of R
- *library(my.package)* - brings the package into RAM so you can use it

```
# is jello associated with diarrhea?
library(epitools) #load 'epitools' package
data(oswego) \#load Oswego dataset
names(oswego) #get some info
epitab(oswego$jello,oswego$ill) #use epitab for RR
# "pretty" up the results (everything is an object)
round(epitab(oswego$jello, oswego$ill,
method = "riskratio")$tab, 3)
 # same function, different method for OR
with(oswego,round(epitab(jello, ill, method = "oddsratio")$tab, 3))
```

the scan() and cbind() functions

R as a spreadsheet

```
weight <- scan()
1: 134 156 222
4:
Read 3 items
height <- scan()
 1: 60 63 72
 4: Read 3 items
bmi <- (weight*703)/height^2
cbind(weight, height, bmi)
     weight height      bmi
[1,]    134     60 26.16722
[2,]    156     63 27.63114
[3,]    222     72 30.10532
```

NB: to scan in character variables use scan(, what = "")

getting "real" data into R

"there's a function for that"

- *read.table() is how you get data into R*
- optimized version *read.csv()* even better

```
cars<-read.table("http://www.columbia.edu/~cjd11/
charles_dimaggio/DIRE/resources/R/cars.txt",
header=T, stringsAsFactors=F)
```

using read.csv

```
dig<-read.csv("http://www.columbia.edu/~cjd11/charles_dimaggio
DIRE/resources/R/dig.csv", header=T,
 stringsAsFactors=F) #digitalis data
 str(dig)
 table(dig$TRTMT,dig$DEATH)
```


YOUR
TURN

- go to `http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/styled-4/styled-6/`
  - DIRE $\rightarrow$ epidemiology $\rightarrow$ R
- click to download the data file called "sparcs" to your machine
- read the file into R using read.csv()
  - remember to save the dataframe to a named object
- what is the mean age?
- how many males are in county "59" (Brooklyn)?

Outline

about data and procedures

- folks come to R from programs like SAS, SPSS and (gasp) Excel
- *data manipulation* steps or procs are followed by *analytic* steps or procs
- these two activities are fairly-well demarcated and differentiated
- data are mutable, procs are immutable

R is different

functional programming, and R objects

- functional programming - data are immutable, functions return new "objects"
  - could be data, could be something else, e.g. a regression object, a table object
  - you only see minimal information about the new object on your screen
  - you can save the results of a function as a new object
- everything in R (including functions) is an object
  - some objects you will learn about: vectors, matrices, arrays, lists, dataframes
  - objects have "characteristics" that determine how they "behave"

practical implications

- same function will return different results depending on the argument object type
- e.g. plot(numeric.data) returns scatterplot, plot(sp.class.data) returns a map
- you can (and often must) supply a function as an argument to another function
- e.g. plot(table(a,b))
- you will write a lot of parentheses
- "magrittr" allows pipeline of operations

```
plot %>%
    table(a,b)
```

Outline

your workspace

and how to work in it

- R session stores objects, data etc ... in a .RData file
  - at end of session will be prompted to "save workspace"
  - you may or may not want to
  - do save your source document (we'll discuss)
- *getwd()* to find location current workspace
- *setwd()* to set it
- *ls()* or *objects ()* to list current objects in workspace
- *rm()* to remove objects
- *save.image()* to save *current* space
- *save()* more control
- *load()* place .RData file back into workspace
- *q()* - quit

objects in your workspace

identifying, assessing, removing

- ls() / objects() - view the objects in the current workspace,
  pattern = search for object names that contain phrase
  eg ls(pattern = "dat")
- rm() / remove() - remove workspace objects
  eg rm(dat, dat2, dat3, dat4)
  CAUTION - rm(list = ls()) will remove everything
- str() / mode() / class() - "go to" functions to assess objects
- *data()* - displays available data sets

```
data(Titanic)
Titanic
str(Titanic)
```

Outline

getting help

online community (`http://r.789695.n4.nabble.com/`), tutorials (`http://www.ats.ucla.edu/stat/r/`), search sites (`http://www.r-project.org/search.html`), books by folks like *Venebles* and *Aragon*, and built-in help:

- *help()* opens help page
- *apropos()*displays all objects matching topic
- *library(help=packageName)* help on a specific package
- *example()* ; *demo()*
- *vignette(package="packageName")*; *vignette(package="topic")*
- RSiteSearch("packageName")

```
help(sample) ; ?sample ; ??sample
apropos("sam")
example(sample)
demo(graphics)
```