an introduction to R for epidemiologists

the basics

Charles DiMaggio, PhD, MPH, PA-C

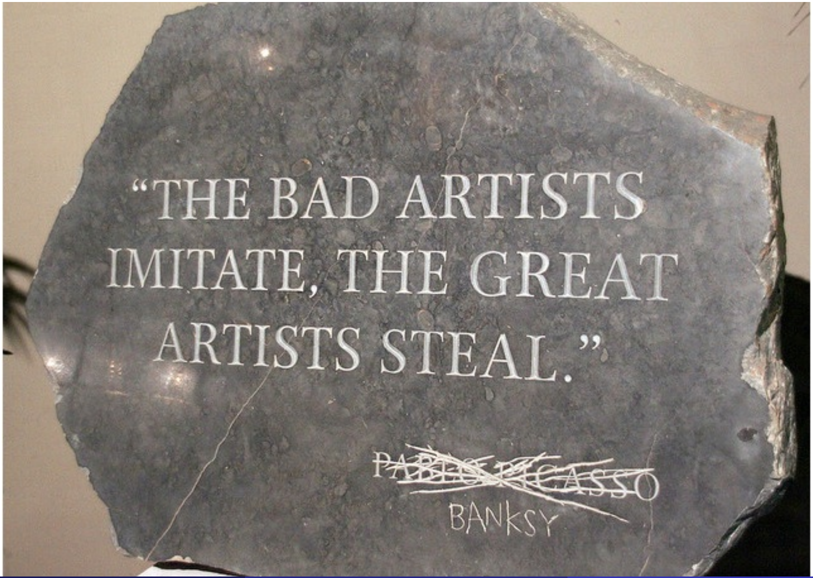
Professor of Surgery and and Population Health
New York University School of Medicine
Bellevue Hospital
Division of Trauma and Surgical Critical Care
462 First Avenue, New York, NY 10016

Fall 2024

- http://www.injuryepi.org/
- Charles.DiMaggio@nyumc.org

Outline

But first...


"THE BAD ARTISTS IMITATE, THE GREAT ARTISTS STEAL."
~~PABLO PICASSO~~
BANKSY

Credit where credit is due...

- Tomas Aragon, MD, DrPH
  - Applied Epidemiology Using R
  - `https://tbrieder.org/epidata/course_reading/e_aragon.pdf`
- John Fox, PhD
  - An Introduction to Statistical Computing in R
  - `https://facsocsci.mcmaster.ca/jfox/Courses/R/ICPSR/intro-to-R-notes.pdf`
- 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.

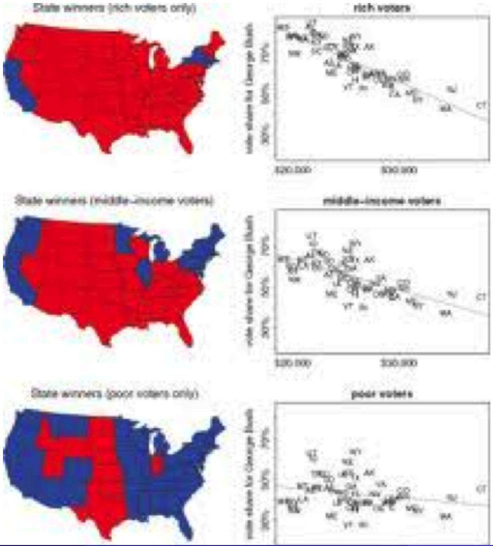
Making stunning graphics

Paul Butler

facebook
December 2010

Predicting Elections

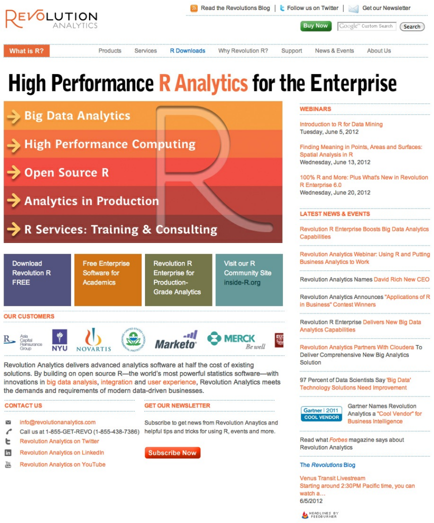
Andrew Gelman


State winners (rich voters only)
rich voters
State winners (middle-income voters)
middle-income voters
State winners (poor voters only)
poor voters
vote share for George Bush
$20,000
$30,000

Web Scraping

John Muschelli, Andrew Jaffe, Jeffrey Leek

Making Money
Microsoft



Read the Revolutions Blog | Follow us on Twitter | Get our Newsletter

Buy Now | Google Custom Search | Search

What is R? | Products | Services | R Downloads | Why Revolution R? | Support | News & Events | About Us

High Performance R Analytics for the Enterprise

- Big Data Analytics
- High Performance Computing
- Open Source R
- Analytics in Production
- R Services: Training & Consulting

Download Revolution R FREE | Free Enterprise Software for Academics | Revolution R Enterprise for Production-Grade Analytics | Visit our R Community Site inside-R.org

OUR CUSTOMERS

Revolution Analytics delivers advanced analytics software at half the cost of existing solutions. By building on open source R—the world's most powerful statistics software—with innovations in big data analysis, integration and user experience, Revolution Analytics meets the demands and requirements of modern data-driven businesses.

CONTACT US

- info@revolutionanalytics.com
- Call us at 1-855-GET-REVO (1-855-438-7386)
- Revolution Analytics on Twitter
- Revolution Analytics on LinkedIn
- Revolution Analytics on YouTube

GET OUR NEWSLETTER

Subscribe to get news from Revolution Anaytics and helpful tips and tricks for using R, events and more.

Subscribe Now

WEBINARS

Introduction to R for Data Mining
Tuesday, June 5, 2012

Finding Meaning in Points, Areas and Surfaces: Spatial Analysis in R
Wednesday, June 13, 2012

100% R and More: Plus What's New in Revolution R Enterprise 6.0
Wednesday, June 20, 2012

LATEST NEWS & EVENTS

Revolution R Enterprise Boosts Big Data Analytics Capabilities

Revolution Analytics Webinar: Using R and Putting Business Analytics to Work

Revolution Analytics Names David Rich New CEO

Revolution Analytics Announces "Applications of R in Business" Contest Winners

Revolution R Enterprise Delivers New Big Data Analytics Capabilities

Revolution Analytics Partners With Cloudera To Deliver Comprehensive New Big Analytics Solution

97 Percent of Data Scientists Say 'Big Data' Technology Solutions Need Improvement

Gartner | 2011 COOL VENDOR — Gartner Names Revolution Analytics a "Cool Vendor" for Business Intelligence

Read what *Forbes* magazine says about Revolution Analytics

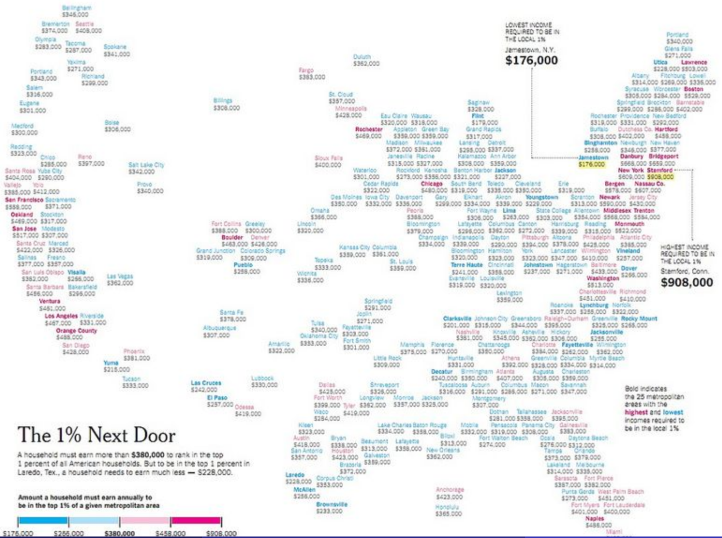
The *Revolutions* Blog

Venus Transit Livestream
Starting around 2:30PM Pacific time, you can watch a...
6/5/2012

HEADLINES BY FEEDBURNER

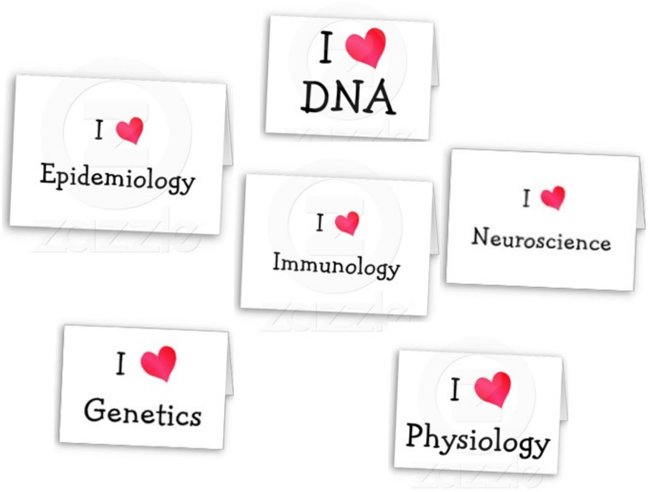
Publishing Newspapers

New York Times


The 1% Next Door
A household must earn more than $380,000 to rank in the top 1 percent of all American households. But to be in the top 1 percent in Laredo, Tex., a household needs to earn much less — $228,000.
Amount a household must earn annually to be in the top 1% of a given metropolitan area
$176,000
$266,000
$380,000
$458,000
$908,000
LOWEST INCOME REQUIRED TO BE IN THE LOCAL 1%
Jamestown, N.Y.
$176,000
HIGHEST INCOME REQUIRED TO BE IN THE LOCAL 1%
Stamford, Conn.
$908,000
Bold indicates the 25 metropolitan areas with the highest and lowest incomes required to be in the local 1%

Doing Science

by scientists for scientists



Doing Public Health
Chicago DOH Food Inspections

"An open source approach helps build a foundation for other models attempting to forecast violations at food establishments. The analytic code is written in R, an open source, widely-known programming language for statisticians. There is no need for expensive software licenses to view and run this code."

what R is *not*

- initially easy and intuitive or a GUI experience
- warrantied in any way (if it runs and doesn't break R, it can be on CRAN)
- (traditionally) well suited to large data sets
  - R stores everything in RAM
  - historical 32-bit limit ($2^{31} - 1$) on size of a vector (objects like matrices are actually vectors)
  - old rules of thumb: 100,000 rows, 20 variables (*very* conservative), single object 10% of RAM, etc...
- but things have changed
  - new approaches and tools
    - optimize reading in your csv files
    - connect to external data bases like SQLite or MonetDB ("dplyr" makes this even easier)
    - packages like "data.table" (fread), parallel computing

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?

how to install R

1. go to http://www.r-project.org/
2. select RANComprehensive R Archive Network) from left menu
3. link to nearby geographic site (e.g. Carnegie Mellon University, Pittsburgh, PA)
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

- start by working in the console
- convert 68 degrees Fahrenheit to Celsius ($C^0 = \frac{5}{9}(F^0 - 32)$)

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

math operators and functions

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

assignment operator

'memory' key

$$<-$$

```
y <- 5^3 # assign, do not return result
y # return result
(y <- 5^3) # assign and return result
```

$\#$

comment character

concatenation *function*

combine or "vectorize"

c()

,

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

indexing

select out results

[ ]

```
x[1]
x[2]
x[c(3,4)]
y[1:3]
```

functions
R "apps"

fx()

have already seen a function: c()

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

math operators and functions

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

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"?

moving from the console to the editor
better

- base R comes with an editor
  - mac: file - new document, command / enter to execute
  - pc: file - new script, control / R to execute
- RStudio is popular
  - integrated development environment (IDE)
  - command (or control) / enter to execute
  - separate instance of R running concomitantly can interfere
- other editors
  - Tinn-R
  - ESS (emacs)
  - textmate macs, R-bundle
  - Vim, Eclipse, Sublime Text

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
```

variables

$

```
myDat<-data.frame(x=c(4,5,6,7), y=c("e","f","g","h"))
x
myDat$x
mydat$x     # capitalization counts!
summary(myDat$x)
summary(x)
```

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

using statistical functions in 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)
```

same function, different output!?

and your first regression in R!

- R functions return different results based on the "object"
    - we will be talking a lot more about "objects" soon

```
summary(myDat$outcome1)
my.reg<-lm(outcome1 ~ grp, data=myDat)
summary(my.reg)
```

- NB: (almost) always save results of procedure to a named "object"

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
- what is the p-value for the regression of age on crash status?

Outline

R packages

- 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

quick example epitools package

did jello cause diarrhea in oswego?

```
install.packages("epitools") # install the package
library(epitools) # load  package
data(oswego) # load Oswego dataset

str(oswego) # get some info
epitab(oswego$jello,oswego$ill) #use epitab
myOR<-epitab(oswego$jello,oswego$ill) # assign to an object
round(myOR,3) # functions work on object
str(myOR)
round(myOR$tab, 3)

round(epitab(oswego$jello, oswego$ill,
method = "riskratio")$tab, 3)
```

YOUR TURN

- load epitools and the oswego data set
- what was the mean age of folks who attended the church picnic?
- compare the odds ratios for the association of illness with having eaten mashed potatoes vs. having eaten cabbage salad
- which, in your estimation, most likely to have caused illness?
- does your opinion change if you calculate risk ratios instead?

Outline

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(
"https://injuryepi.org/resources/Data/cars.txt",
header=T,
stringsAsFactors=F
)
```

using read.csv

```
dig<-read.csv(
"https://injuryepi.org/resources/Data/dig.csv",
header=T,
stringsAsFactors=F) #digitalis data
str(dig)
table(dig$TRTMT,dig$DEATH)
```

YOUR
TURN

- go to https://injuryepi.org/styled-21/styled-6/
    - scroll down to data sets on left margin
- click to download the data file called "sparcs" to your machine
- read the file into R using read.csv()
    - HINT: use the options to set the header to true and strings as factors to false
    - **remember to save the dataframe to a named object**
- what is the mean age?
- how many males are in county "59" (Brooklyn)?
    - HINT: use the table() function with the variable names for gender and county separated by a comma

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

- an R "workspace" is a collection of all the objects, including R dataframes, that you created or imported
- *ls()* to see what objects are in your workspace
  - *rm()* to remove objects no longer need, cluttering or taking up too much space
  - e.g. rm(obj1, obj2, obj3)
  - rm(list = ls()) will remove *everything* CAUTION!
- *getwd()* to find the physical location of your workspace on your computer
  - *setwd()* if you want to use a different location, e.g. can put on cloud service like Box to collaborate

saving objects and data

do you really want to save that workspace?

- at end of session prompted to "save your workspace"
- this automatically creates an image of everything currently in your workspace in the current physical location of your workspace
- next time you start up R, everything is there
- *could* be efficient if you are working in an explicitly defined working directory, or collaborating and ls() and rm() to keep track of things.
- otherwise large objects tend to accumulate and eat up R's memory, similar or identically-named objects sow confusion
- (my) preferred approach is to save editor files, and save and reload only large data files

so, how should I save files?

- always save your editor or script file
    - It's just a text file, really. Use the file commands of whatever program you are working in.
- in R use *save()* to specify only those large data frames or matrices you want to keep, and write them to a .Rdata (or .Rda) file
- then use *load()* to bring everything into you nice clean workspace
    - save(myDat1, myDat2, file="charlie/desktop/myproject.Rdata")
    - load("charlie/desktop/myproject.Rdata")
- *saveRDS()* quicker, more efficient if you are working with a single, very large data set use
- *readRDS* to read it back in
    - saveRDS(myDat, file="charlie/desktop/myDat.RDS")
    - myDat¡-readRDS(file="charlie/desktop/myDat.RDS")

Outline

getting help

online community (https://stackoverflow.com/questions/tagged/r), tutorials (https://stats.oarc.ucla.edu/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)
```