Intro to R

http://jacobfenton.s3.amazonaws.com/R-handson.pdf

Jacob Fenton
CAR Director
Investigative Reporting Workshop,
American University

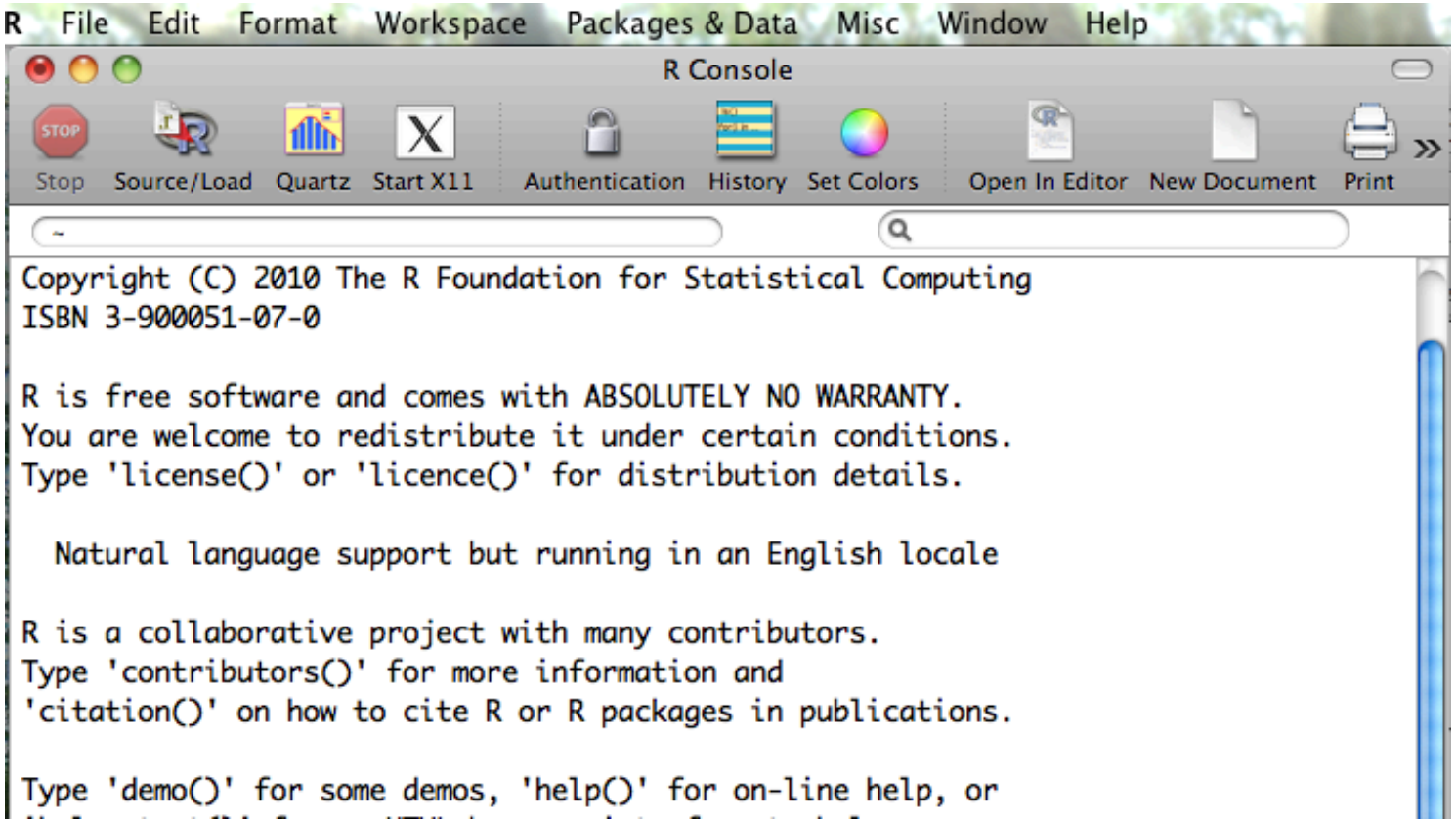
INVESTIGATIVE
Reporting Workshop
AMERICAN UNIVERSITY SCHOOL OF COMMUNICATION

Overview

- Import data
- Move around the file system, save an image
- Do some pretty basic commands
- A few simple graphics

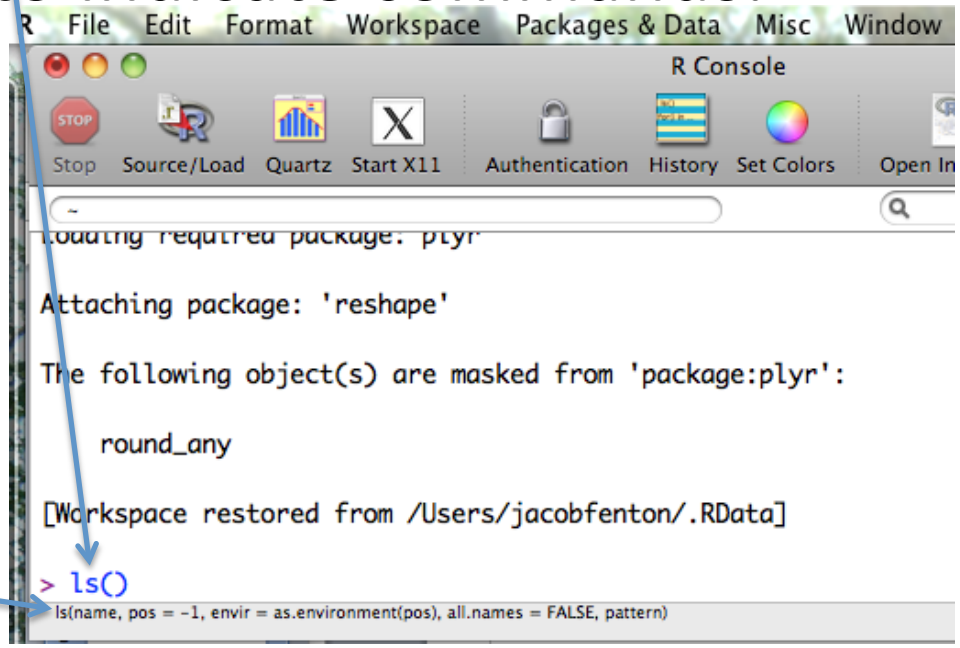
1. Start up R

- Not totally sure of version installed in labs.
- The stuff that follows is on Mac OS 10.6, so…



Where to enter commands

- Enter commands in the console, where there's an arrow On my screen text appears blue. I'll use the '>' prompt to indicate commands.



Depending on version, R may suggest argument completions—but keep in mind that many arguments can be null.

Packages

- You should figure out how to import packages on the system that you're using. I'm not attempting to do so in the lab (and I'm really hoping the stuff I want to use here works!)
- There are often many ways to do stuff—I'm trying to do stuff here in a way that's relatively simple and makes sense (to me), but some tasks are handled much more easily with external libraries.

Documentation – finding help

- The official docs are complete, but sometimes have incredible amounts of detail
- Often I find it easier to google for stuff and poke around until it works
- Really helpful to have someone to ask.

Getting around the filesystem

- R knows where you are in the filesystem and has a 'working directory'

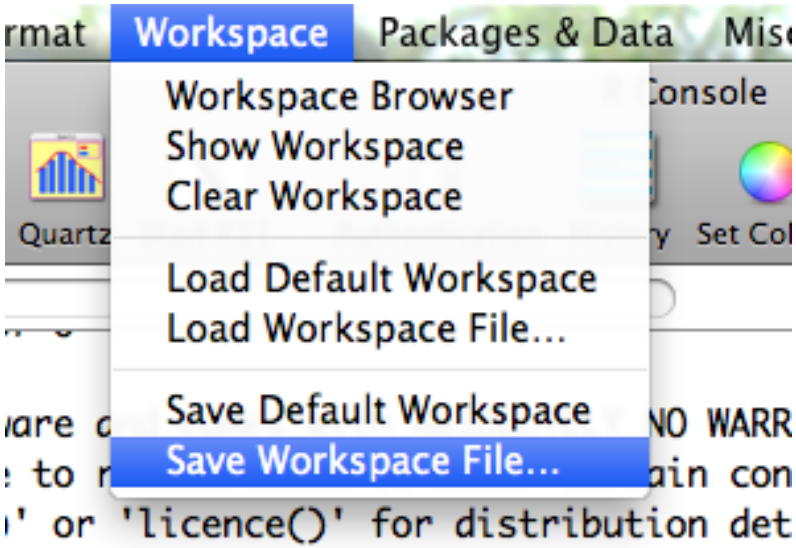
> getwd()

> setwd("/some/file/path")

It's not incredibly useful to navigate like this—but it's helpful if you save a file and then can't remember where.

Saving a file

- You can save a text file with your input—which is helpful—but you can also save the workspace, which is often more helpful. It includes command history, etc.



Adding data

Download the test data from:

http://jacobfenton.s3.amazonaws.com/presentation_files.zip

```
> a <- read.delim("nicar_demo.txt", header=FALSE, sep="\t",
    quote="")
> names(a) <- c('geo_name', 'state', 'county', 'total',
    'less_than_hs_grad_rate', 'less_than_hs_grad_rate_f',
    'less_than_hs_grad_rate_m', 'ba_plus_rate',
    'ba_plus_f_rate', 'ba_plus_m_rate', 'fraction_male', 'mi',
    'mi_lthsg', 'mi_male', 'mi_female', 'mi_fmratio')
```

* Sometimes R is picky about quotes—if these commands don't work perfectly try them with only double quotes, etc.

Aside: about the data or, not suitable for analysis

- 2009 5-year census estimates for all US Census tracts (sum level 140), with rates I added. The uncertainties aren't included.
- B15002: SEX BY EDUCATIONAL ATTAINMENT FOR THE POPULATION 25 YEARS AND OVER --- Universe: Population 25 years and over
- B20004: MEDIAN EARNINGS IN THE PAST 12 MONTHS (IN 2009 INFLATION-ADJUSTED DOLLARS) BY SEX BY EDUCATIONAL ATTAINMENT FOR THE POPULATION 25 YEARS AND OVER --- Universe: Population 25 years and over with earnings

Nicar_demo.txt header key

"geo_name" - the census' name for the tract
"state" - state number
"county" - county number
"total" - total 25 and older
"less_than_hs_grad_rate" - rate of 25+ with less than a hs diploma
"less_than_hs_grad_rate_f" - rate of 25+ women with less than a hs diploma
"less_than_hs_grad_rate_m" - rate of 25+ men with less than a hs diploma
"ba_plus_rate" - rate of 25+ with a BA or higher
"ba_plus_f_rate" - rate of 25+ women with a BA or higher
"ba_plus_m_rate" - rate of 25+ men with a BA or higher
"fraction_male" - fraction of residents 25+ that are male
"mi" - median income (of Population 25 years and over with earnings)
"mi_lthsg" - median income of residents with less than a hs diploma
"mi_male" - median income of men
"mi_female" - median income of women
"mi_fmratio" - ratio of median income of women to median income of men.

About adding data

- Read.delim works the way you think it will. It doesn't require a 'quote' argument, but it's sometimes helpful to set it to empty (if there isn't one) so it doesn't get confused by actual quotes.
- You can use header=True too if there are headers, of course.
- There are libraries for importing excel files, but we're keeping it simple here (not sure about hardware here)
- Always check that the right number of rows were imported. You can use:

```
> nrow(a)
```

Rename variable, column names

- We imported the file as ‘a’. Use the assignment operator ‘<-’ to save it to a new dataframe (that’s r’s word for named columns).

> mydata <- a

If you just type in mydata and hit return it’ll try to output all the data. Not very useful. Instead use:

> colnames(a)

Simplify data, summarize

You can access just a single column of a dataframe with the '$' operator, i.e.

mydata$mi

Try:

> summary(mydata$mi)

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2499   26250   31990   34790   40710  198600     522
```

Shows minimum, quartiles, mean and NA.

You can also summarize the whole dataframe at once with summary(mydata) – its less readable

What does summary show?

- These are census tracts, so the results are for a single tract
- Quartiles—like percentiles, but ¼. Also, the mean is included.

Slicing and dicing

- Simplest way—create a subset of a dataframe:

```
> alabama <- subset(mydata, mydata$state=='1')
> nrow(alabama)
[1] 1082
```

Standard deviation

Standard deviation is sd. But this doesn't work:

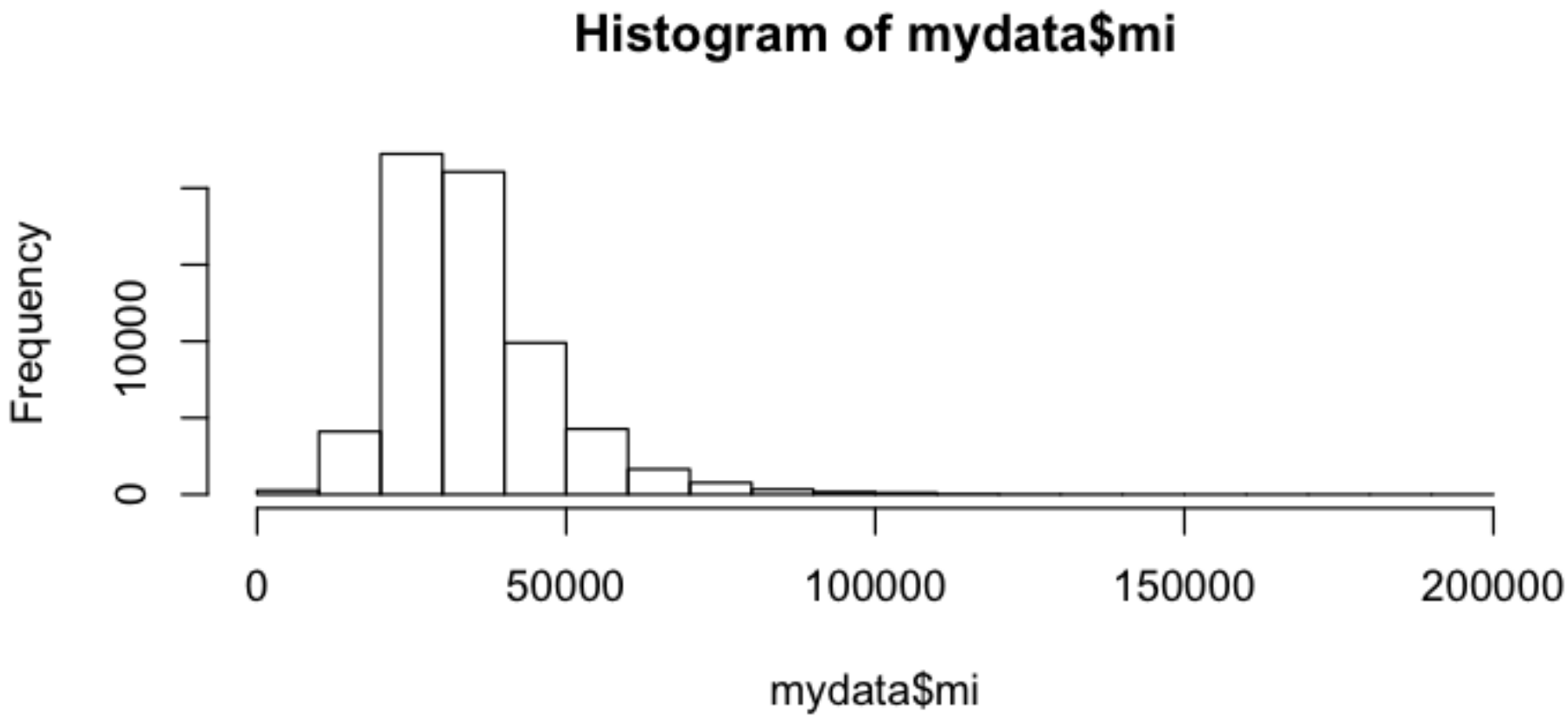
```
> sd(mydate$mi)
[1] NA
```

Instead, remove the nulls with na.rm = True

```
> sd(mydate$mi, na.rm=TRUE)
[1] 13278.38
```

Visualize median income of tracts

> hist(mydata$mi)



Pretty up the histogram slightly

- Lets add axes labels, titles, and save it to a file with the png command.

```
> png("median_income.png", width=500,
  heigh=300, units="px")
> hist(mydata$mi/1000,  xlab="Median income,
  $000", main="Median income in Census tracts")
> dev.off()
```

Can't find the file ? Run

```
> getwd()
```

What the heck is “c”?

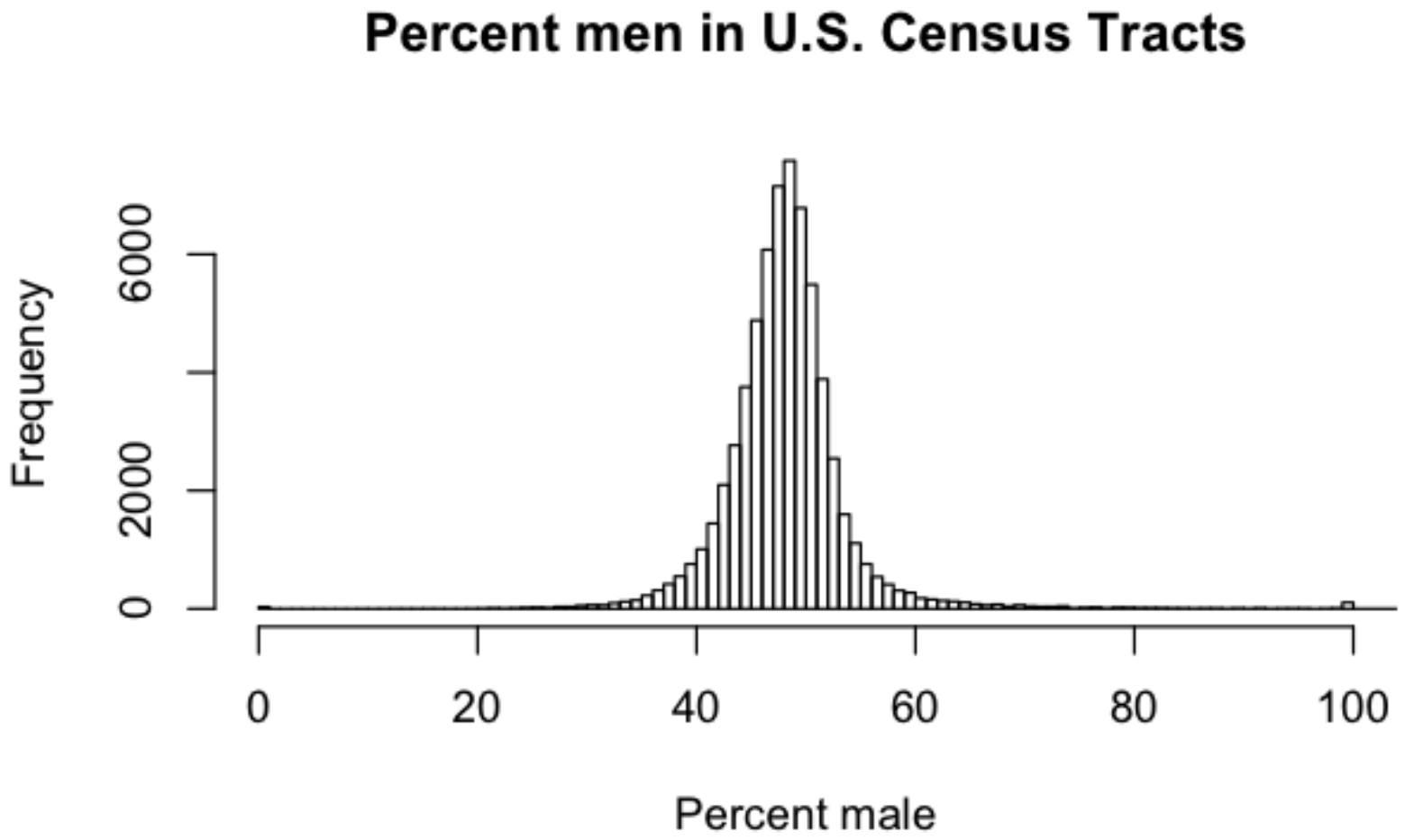
```
> c(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10
> c(10*1:10)
 [1]  10  20  30  40  50  60  70  80  90 100
> c("blah", "blah2")
[1] "blah"  "blah2"
>
```

Getting more specific with graphing

- Using columns to set ‘breaks’ in the histogram
- You often have to create a column of values, or a list of things as an argument—graphing is no exception

```
> hist(a$fraction_male*100,breaks=c
  (1*0:100,1000), xlim=c(0,100), freq=TRUE,
  xlab="Percent male", main="Percent men in
  U.S. Census Tracts")
```

Result—more ‘bins’

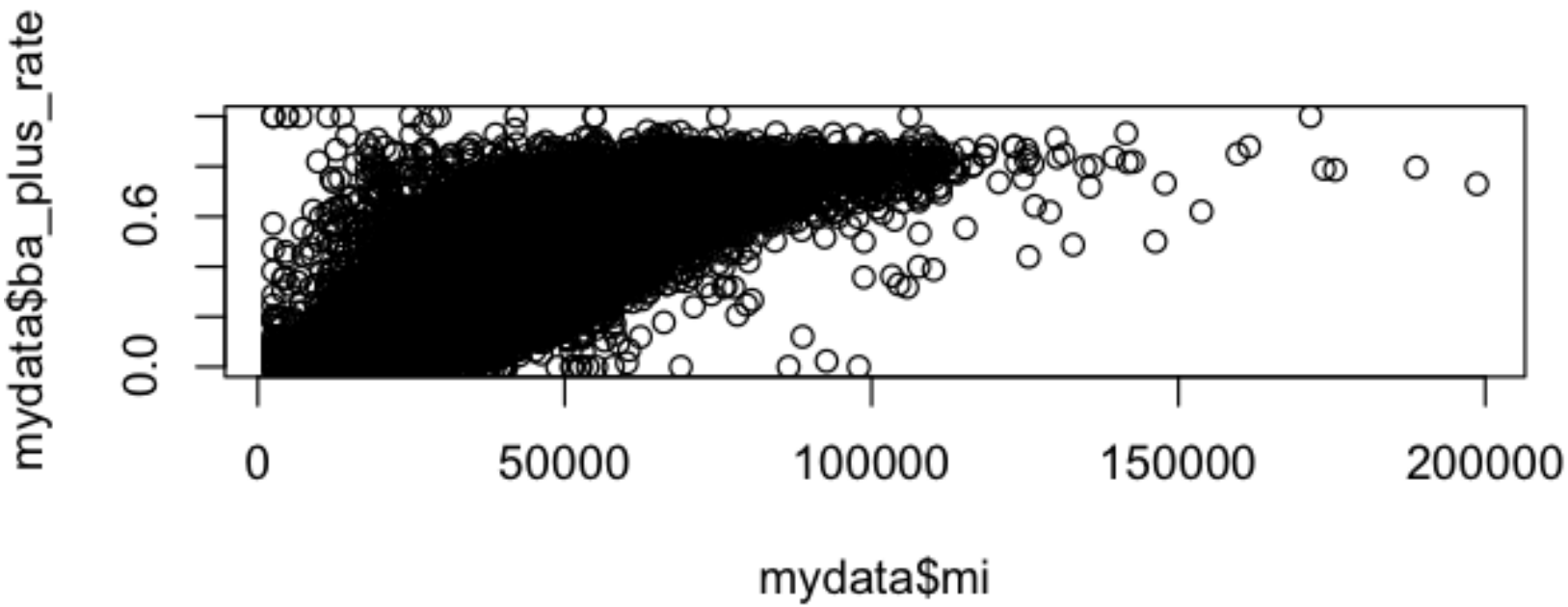


Scatter plot

- Simple to throw up a scatter plot.

> plot(mydata$mi, mydata$ba_plus_rate)

There's a lot of points here though..



Quantifying relationships

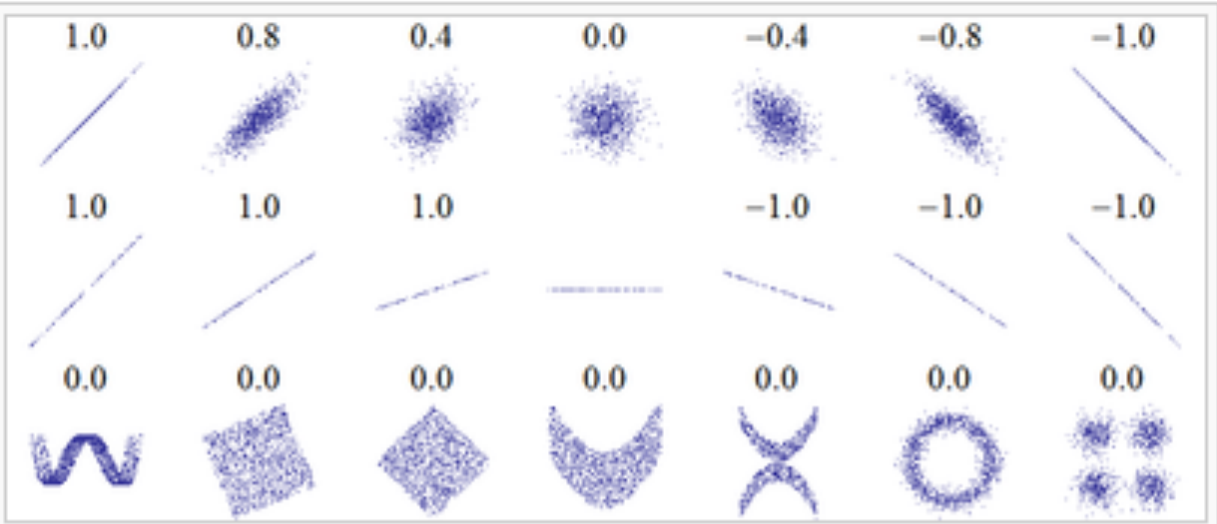
- Plotting the data helps visualize what's going on, but it's often helpful to quantify it.

```
> cor(mydata$mi, mydata$ba_plus_rate,
    use="complete.obs")
[1] 0.7771208
```

That's a really high number—as you might have expected.

What does correlation look like?

Correlation finds linear relationships—but not slope.
Image shamelessly ripped off from Wikipedia



Several sets of (*x*, *y*) points, with the Pearson correlation coefficient of *x* and *y* for each set. Note that the correlation reflects the noisiness and direction of a linear relationship (top row), but not the slope of that relationship (middle), nor many aspects of nonlinear relationships (bottom). N.B.: the figure in the center has a slope of 0 but in that case the correlation coefficient is undefined because the variance of *Y* is zero.

Correlation uncertainty

```
> cor.test(mydata$mi, mydata$ba_plus_rate, use="complete.obs")

        Pearson's product-moment correlation

data:  mydata$mi and mydata$ba_plus_rate
t = 314.6599, df = 64937, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7740561 0.7801491
sample estimates:
      cor
0.7771208
```

The interval is really small because the sample size is so big. BUT: this uncertainty doesn't include the uncertainty of the variables. Also, uncertainty for correlation is, well, not something that easily translates into story-ese.

Correlation matrices

- R can do a whole boatload of correlations at once. We need to convert a dataframe to a matrix first though.

> mydatamatrix <- data.matrix(mydata)

> cor(mydatamatrix, use="complete.obs")

This will spit out a pretty big matrix. We can also dump it to a text file for analysis:

> write.table(cor(mydatamatrix, use="complete.obs"), file="correlations.txt", sep="|", eol="\n", row.names=TRUE)

Can import this to excel, etc.

Full file locations

- http://jacobfenton.s3.amazonaws.com/R-handson.pdf
- http://jacobfenton.s3.amazonaws.com/nicar-raleigh/nicar_demo.txt
- http://jacobfenton.s3.amazonaws.com/presentation_files.zip