BEGINNER R

adam hogan
@mittenchops on github

Presentation for NYC Stats Programming Masters Class

New York, NY
November 06, 2013

OUR GOAL

- If you're not already familiar with stats/programming elsewhere, this should give you enough examples to dig in yourself.
- If you are already familiar with statistics or programming through other languages, this should get you up to speed in R.
- If you're familiar with both, you might understand some concepts of the language more formally.

INTRO AND DATA MANAGEMENT

Part I

FOLLOWING ALONG

- If you're having trouble seeing the screen, or just want to go at your own pace, check out the code here:
- https://github.com/mittenchops/NYCStatsMasters/
- The file is called `walkthrough.R`

INSTALLATION

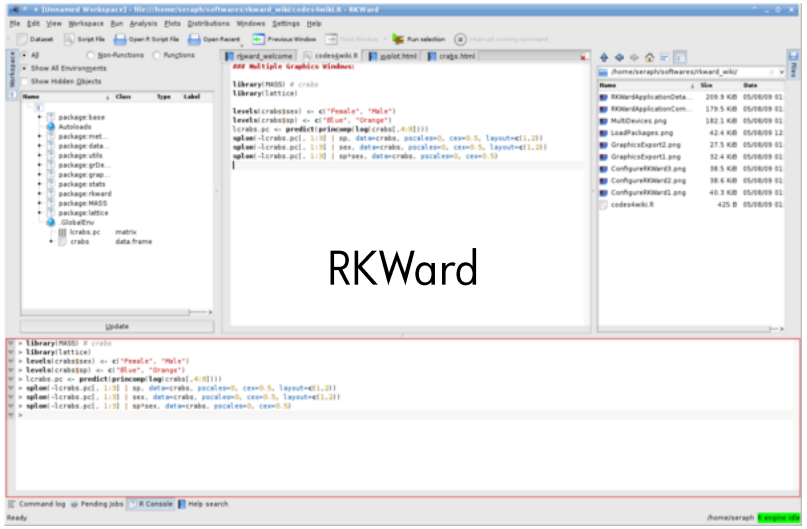
- Windows
- Ubuntu
- Mac

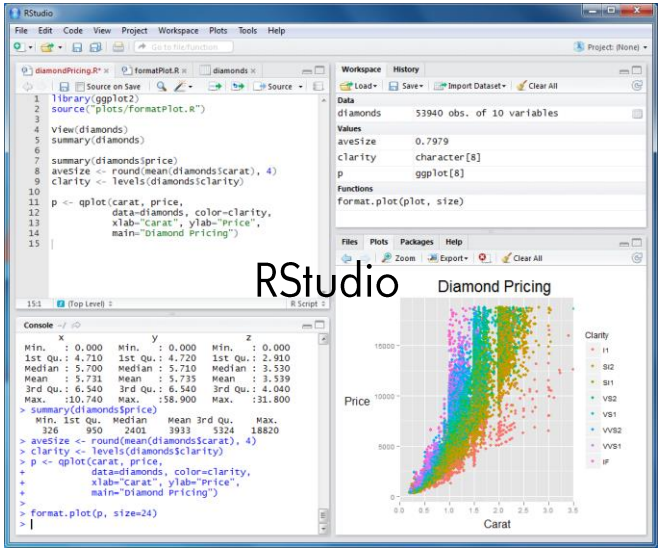
```
# Windows
C:\Users\name>R-3.0.2-win.exe

# LINUX
$ sudo apt-key adv --keyserver
keyserver.ubuntu.com --recv-keys
E084DAB9

$ deb
http://http://cran.revolutionanalytics.
com/bin/linux/ubuntu precise/

$ sudo apt-get update

$ sudo apt-get install r-base r-base-
dev
```

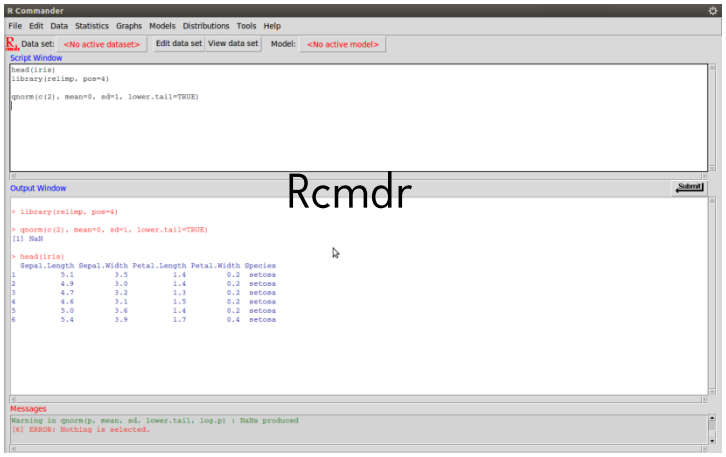
COMPOSING R

- R Studio <- most popular IDE
- Eclipse <- if you're coming from compiled
- R Commander <- popular on Windows, easy install
- RKWard <- KDE
- Revolution R <- Fancy
- Deducer
- ESS
- Or just your terminal and a text editor.
  - I like a persistent terminal like Guake.

AT A GLANCE








```
$ Rscript myscript.R
$ R
R version 2.14.1 (2011-12-22)
Copyright (C) 2011 The R
Foundation for Statistical
Computing
ISBN 3-900051-07-0
```

HOW TO GET HELP

```
> ?plot
> # this is a
> # shortcut for
> # help(plot)
> ??kolmogorov
> vignette()
> # this is a
> # comment.
```

- Stackoverflow for R
- CrossValidated for statistics
- As a beginner, I would avoid the Rmailing lists.

USING PACKAGES

- To download a package in the first place, use `install.packages()`
  - dep=T means install dependencies as well.
- To use a package you already have downloaded, use the `library()` function

```
# Install for the first time
>
install.packages(
"quantmod",
dep=T)

# To use, once downloaded:
>
library(quantmod)
```

Quotes

No quotes

GETTING DATA

- R has several data sets already built in.
  - Iris, measurements of flowers
  - AirPassengers, volume of air travel
  - Beaver1, body temperatures of aquatic rodents
- Type `data()`

```
> iris
 Sepal.Length Sepal.Width Petal.Length Petal.Width
Species
1          5.1         3.5          1.4         0.2
setosa
2          4.9         3.0          1.4         0.2
setosa
3          4.7         3.2          1.3         0.2
setosa
4          4.6         3.1          1.5         0.2
setosa
5          5.0         3.6          1.4         0.2
setosa
6          5.4         3.9          1.7         0.4
setosa

> AirPassengers
[1] 112 118 132 129 121 135

> beaver1
  day time  temp activ
1 346  840 36.33     0
2 346  850 36.34     0
3 346  900 36.35     0
4 346  910 36.42     0
5 346  920 36.55     0
6 346  930 36.69     0
```

OTHER LANGUAGES' DATA

```
> library(xlsx)
> library(foreign)

# SAS
> read.xport(file)

# Stata
> read.dta(file)

# SPSS
> read.spss(file)

# Matlab
> read.octave(file)

# minitab
> read.mtp(file)
```

- Idiom: read.method()
- xlsx lets you use modern excel files (read and write)
- Foreign lets you import from
  - SAS
  - Stata
  - SPSS
  - Matlab
  - Minitab
  - S
  - Systat

MARKUP INTO R

```
> library(XML)
> url2 <-
'http://www.faa.gov/data_research/passengers_cargo/unruly_passengers/'
> X <- readHTMLTable(url2, header=T,
stringsAsFactors=FALSE)[[1]]
> X
   Year                    Total
1  1995                      146
2  1996                      184
3  1997                      235
4  1998                      200
5  1999                      226
6  2000                      227
7  2001                      300
8  2002                      306
9  2003                      302
10 2004                      330
11 2005                      226
12 2006                      156
13 2007                      176
14 2008                      134
15 2009                      176
16 2010                      148
17 2011                      131
18 2012                      130
19 2013 68 as of September 30, 2013
```

- library(XML) is useful for scraping HTML tables
- readHTMLTable()
  - Add or remove headers
  - stringsAsFactors=F
  - skip.lines=n
  - [[1]] first element of list
  - …to taste.

OTHER DATA SOURCES

- Wrapper packages often help you connect more easily to external APIs.

```
> library(quantmod)
> library(twitteR)
> library(RNYTimes)
> library(RClimate)



> getSymbols("GOOG")
[1] "GOOG"


>
searchTwitter('#ilovestatistics'
,n=10)[[2]]


[1]"Statistics: the best kind of
homework #ilovestatistics #nerd
#shouldhavebeenastatistician
#gradschoolproblems"
```

BASICS

- Like all programming languages, R has a concept of different types of data.
  - character, logical, numeric, integer, complex
- These basic types are combined into structured groups.

```
# Assignment
> Person <- ”Fred”
> Person
[1] "Fred"

# Equality
> 0 == F
[1] TRUE
```

Single “=“ works, too.
Try to use “<-” instead.

BASICS

- Like all programming languages, R has a concept of different types of data.
  - character, logical, numeric, integer, complex
- These basic types are combined into structured groups.

```
# Assignment
> Person <- ”Fred”
> Person
[1] "Fred"

# Equality
> 0 == F
[1] TRUE
```

```
# Other operators work as you’d expect

> 1 + 1
[1] 2
> 2*4
[1] 8
> 2^5
[1] 32
> 300 %% 21
[1] 6
```

DATA TYPES

- Types:
  - vector
  - matrix
  - list
  - data frame
- Other types you'll run into, like objects, are *composites* of these simple types.
- Generally, you'll want a matrix or a data frame.
  - And you'll build it out of vectors and lists.

```
> a <- c(1,2,3,4)
> b <- matrix(c(1,2,3,4), nrow=2)
> c <- list("a"="fred", "b"="bill")
> d <- data.frame(b)

> a # VECTOR
[1] 1 2 3 4

> b # MATRIX
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> c # LIST (Note the key-value structure)
$a
[1] "fred"
$b
[1] "bill"

> d # DATA FRAME
  X1 X2
1  1  3
2  2  4
```

CONVERSION AND COERCION

- Conversion
  - Paradigm is as.X() or the name of the class you're casting to.
  - Sometimes the "as." is unnecessary.
    - matrix() rather than as.matrix()
    - data.frame() rather than as.data.frame()
  - If you're using a package where one doesn't work, try the other.
  - Use ?help

```
> as.data.frame(a)
  a
1 1
2 2
3 3
4 4
> data.frame(a)
  a
1 1
2 2
3 3
4 4

> as.matrix(a, nrow=2) # WATCH OUT
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
> matrix(a, nrow=2)    # THIS INSTEAD!
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

VARIABLE INTERROGATION

- Basics:
  - head(), tail(),
  - str(),
  - dim, ls.str(),
  - View, summary().
- Objects have attr():
  - names()
  - dim()
  - dimnames()
  - class()

```
> Y <- runif(200)
> str(Y)
num [1:200] 0.5053 0.3564 0.0359 0.7377
0.0302 ...

> head(Y) # GIVE ME THE FIRST 5
[1] 0.50525553 0.35636648 0.03589792
0.73766891 0.03020607 0.50628327

> tail(Y) # GIVE ME THE LAST 5
[1] 0.6612501 0.9930194 0.8392855
0.5459498 0.2587155 0.3704778

> dim(Y) # NOPE! HE IS A VECTOR
NULL

> length(Y)
[1] 200
```

INDEXES

Rows Columns

```
A[m,n]
```

Rows Columns

```
A[ ,n]
```

Blank means "Give me EVERYTHING"

USING INDEXES

- Give me column 1
- Give me row 2
- Give me all rows EXCEPT row 1

- Give me all days where price > $768.00.
- Give me all states where unemployment > 10%

```
> head(unemp)
   rank     region Aug. 2012 Sept. 201 change
14   14    alabama       8.5       8.3   -0.2
15   14     alaska       7.7       7.5   -0.2
27   27    arizona       8.3       8.2   -0.1
16   14   arkansas       7.3       7.1   -0.2
2     2 california      10.6      10.2   -0.4
17   14   colorado       8.2       8.0   -0.2

> unemp[unemp[4]>10,]
   rank       region Aug. 2012 Sept. 201 change
2     2   california      10.6      10.2   -0.4
11    6       nevada      12.1      11.8   -0.3
24   14 rhode island      10.7      10.5   -0.2
```

```
> b
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> b[,1]  # ALL ROWS, FIRST COLUMN
[1] 1 2

> b[2,]  # SECOND ROW, ALL COLUMNS
[1] 2 4

> b[-1,] # ALL ROWS EXCEPT 1
[1] 2 4

> b>3
      [,1]  [,2]
[1,] FALSE FALSE
[2,] FALSE  TRUE

> GOOG[GOOG[,6]>1036.00,6]
           GOOG.Adjusted
2013-10-29       1036.24
```

USE NAMES INSTEAD: $

- Call out columns to assign values with the $ sign.
  - In a list or a df

```
> name <- c("Fred", "Bill")
> occupation <- c("Doctor", "Dancer")
> people <- data.frame(name, occupation)

> people
  name      occupation
1 Fred          Doctor
2 Bill          Dancer

> people$age <- 35
> people
  name      occupation age
1 Fred          Doctor  35
2 Bill          Dancer  35

people[people$name=="Fred",]$age=40
```

```
> XX <- 1
> XX$name <- "Fred"
Warning message:
In XX$name <- "Fred" : Coercing LHS
to a list
> XX$occupation <- "Doctor"
> XX$age <- 21

> XX
[[1]]
[1] 1

$name
[1] "Fred"

$occupation
[1] "Doctor"

$age
[1] 21

> XX$name == XX[2]
```

MORE STRUCTURE

Combine columns

```
cbind(a,b)
```

Combine rows

```
rbind(a,b)
```

MORE ADVANCED

- data.table
  - This is a data frame that has some very fast indexing properties.
  - Nothing to dive into right away, but you may see people using it on answers online.
  - We'll cover this in a future workshop.

MAKING A FUNCTION

- `function(arg1,arg2)` is the syntax, followed by your function block.
- The second version, shown here with implicit return, can work a lot like a lambda function.

```
> mult2 <-
function(arg1,arg2){
+ z = arg1 * arg2
+ return(z)
+ }
> mult2(2,10)
[1] 20



> g <- function(x){x^2}
> mylist <- c(1,2,3,4,5)
> g(mylist)
[1]  1  4  9 16 25
```

NOW YOU CAN...

- Install R
- Use a GUI
- Install packages
- Load built-in datasets
- Import data from other software
- Import data from HTML
- Use vectors, matrices, lists, data frames
- Interrogate variables with head(), tail()
- Index data structures
- Bind rows and columns
- Make functions

Next up...
* Statistical tests
* Regression
* And forecasting babies

FOR THE BREAK (If you want to)

- Take the data we downloaded for unruly airline passengers.
- Use your indexing knowledge to remove the last observation (2013)
- Convert the vector to numeric
- Use ? to look up how to take differences of a vector
- Write a function that…
  - computes the average yearly change of values
  - replaces the last value with that average + the 2012 value
  - A simple forecast. You're random walking, now.

Hint:

```
mean(diff(as.numeric(X[-19,2])))
```

MODELS AND TESTS

Part II

LET'S LOAD SOME DATA

- When I first moved here, New Yorkers would say, "Chicago is nice, but it's soooo cold!"
- Chicago is at 41.8°N, New York at 40.7°N; that's only 60 miles north on sphere earth. That can't be that much colder. Let's find out.

```
nyc <-
c(33.4,33.3,33.1,33,32.9,3
2.8,32.7,32.6,32.5,32.4,32
.4,32.3,32.3,32.3,32.3,32.
2,32.2,32.3,32.3,32.3,32.3
,32.4,32.4,32.5,32.5,32.6,
32.7,32.8,32.9,33,33.1)


chi <-
c(24.4,24.3,24.2,24.1,24,2
3.9,23.9,23.8,23.7,23.7,23
.6,23.6,23.5,23.5,23.5,23.
5,23.4,23.4,23.4,23.4,23.5
,23.5,23.5,23.6,23.6,23.7,
23.8,23.9,24,24.1,24.2)
```

STATISTICAL TESTING

- The t-test
  - t.test()
  - Used to test whether two sets of data are statistically different from each other.
- Looks pretty different.

```
> t.test(nyc,chi,paired=T)

        Paired t-test

data:  nyc and chi
t = 644.8545, df = 30,
p-value < 2.2e-16
alternative hypothesis: true
difference in means is not equal to
0
95 percent confidence interval:
 8.830011 8.886118
sample estimates:
mean of the differences
                8.858065
```

STATISTICAL TESTING

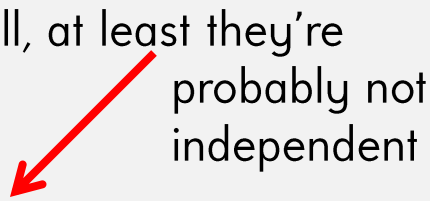
- The F-test
  - var.test
  - Used to test whether variances are the same between two populations.
  - Used in model fitting.
- Well, maybe Chicago is colder on average, but is more consistent in temperature?
- Can't reject that they're the same variance.

```
> var(nyc)
[1] 0.1152903
> var(chi)
[1] 0.08658065

> var.test(nyc,chi)

        F test to compare two variances

data:  nyc and chi
F = 1.3316, num df = 30, denom df = 30, p-value = 0.4375
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6420592 2.7616524
sample estimates:
ratio of variances
          1.331595
```

STATISTICAL TESTING

- Chi-squared
  - chisq.test()
  - Can do lots of stuff, like testing for independence of two samples, or goodness-of-fit for model.
  - You'll probably use the second one more.

```
> chisq.test(chi,nyc)

        Pearson's Chi-squared test

data:  chi and nyc
X-squared = 217.8611, df = 110, p-value = 4.216e-09

Warning message:
In chisq.test(chi, nyc) : Chi-squared approximation may be incorrect
```



FORMULA

```
Y ~ X1 + X2
```

is explained by

FORMULA

```
Y ~ X1 + X2
```

```
Y ~ .
```

FORMULA

```
Y ~ X1 + X2
```

```
Y ~ .
```

Interaction terms

```
Y ~ X1+X2+X1:X2
    +Z^2
```

Expressions are allowed too

LINEAR REGRESSION

- Welcome to `lm`
- Load up the dataset cars.
- Let's examine how many feet a Tin Lizzie takes to stop, depending on how fast it's going.
- Every additional MPH seems to add an additional 4 feet.

```
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10

> fit <- lm(dist ~ speed, data=cars)

> fit
Call:
lm(formula = dist ~ speed, data =
cars)

Coefficients:
(Intercept)         speed
    -17.579         3.932
```

LOGISTIC REGRESSION

- Welcome to glm
- First, make distance into a threshold variable with ifelse()
- Family=binomial() gives a logistic regression

```
> c2 <- cars
> c2[,2] <- ifelse(c2[,2]>20,1,0)
> head(c2)
  speed dist
1     4    0
2     4    0
3     7    0
4     7    1
5     8    0

> lfit <- glm(dist ~ speed, data=c2,
family=binomial)

> lfit
Call:  glm(formula = dist ~ speed, family
= binomial, data = c2)

Coefficients:
(Intercept)        speed
    -5.0800       0.5067

Degrees of Freedom: 49 Total (i.e. Null);
48 Residual
Null Deviance:         50.04
Residual Deviance: 27.9     AIC: 31.9
```

DIAGNOSTICS

- Summary()
- Compare models
  - anova(fit1,fit2)
- Cross-validation
  - DAAG
  - Write your own

```
> summary(fit)
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’
0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of
freedom
Multiple R-squared: 0.6511,     Adjusted R-squared:
0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

DIAGNOSTICS

- Summary()
- Compare models
  - anova(fit1,fit2)
- Cross-validation
  - DAAG
  - Write your own

```
> summary(fit)
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’
0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of
freedom
Multiple R-squared: 0.6511,     Adjusted R-squared:
0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Looks significant

TIME SERIES MODELING

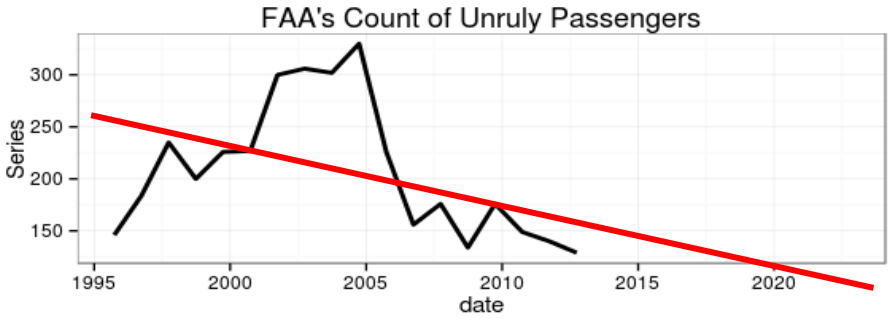
- There are several time series libraries in R
  - ts
  - zoo
  - xts
  - timeseries
- Some functions play nice across libraries…

```
xts(x = NULL,
        order.by = index(x),
        frequency = NULL,
        unique = TRUE,
        tzone = Sys.getenv("TZ"),
        ...)
```

```
> url2 <-
'http://www.faa.gov/data_r
esearch/passengers_cargo/u
nruly_passengers/'

> X <- readHTMLTable(url2,
header=T,
stringsAsFactors=FALSE)[[1
]]

> head(X)
  Year Total
1 1995   146
2 1996   184
3 1997   235
4 1998   200
5 1999   226
6 2000   227
```

TIME SERIES, QUICKLY...

- First thought is maybe just draw a "trend line"---or a regression.
- This neglects something very powerful though---you have the ORDER of items

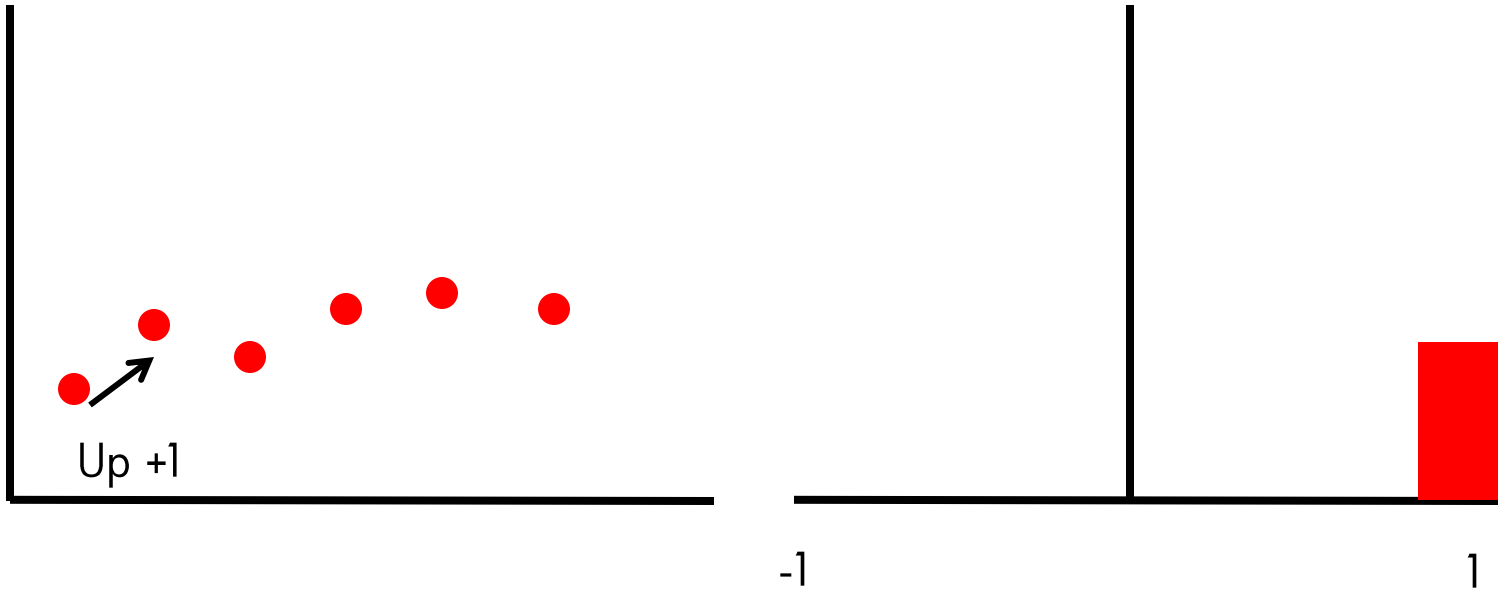


- Y1, Y2, Y3 may depend on the change in the value before them, not the value itself.

not $Y \sim X$ but $Y_n \sim Y_{n-1}$

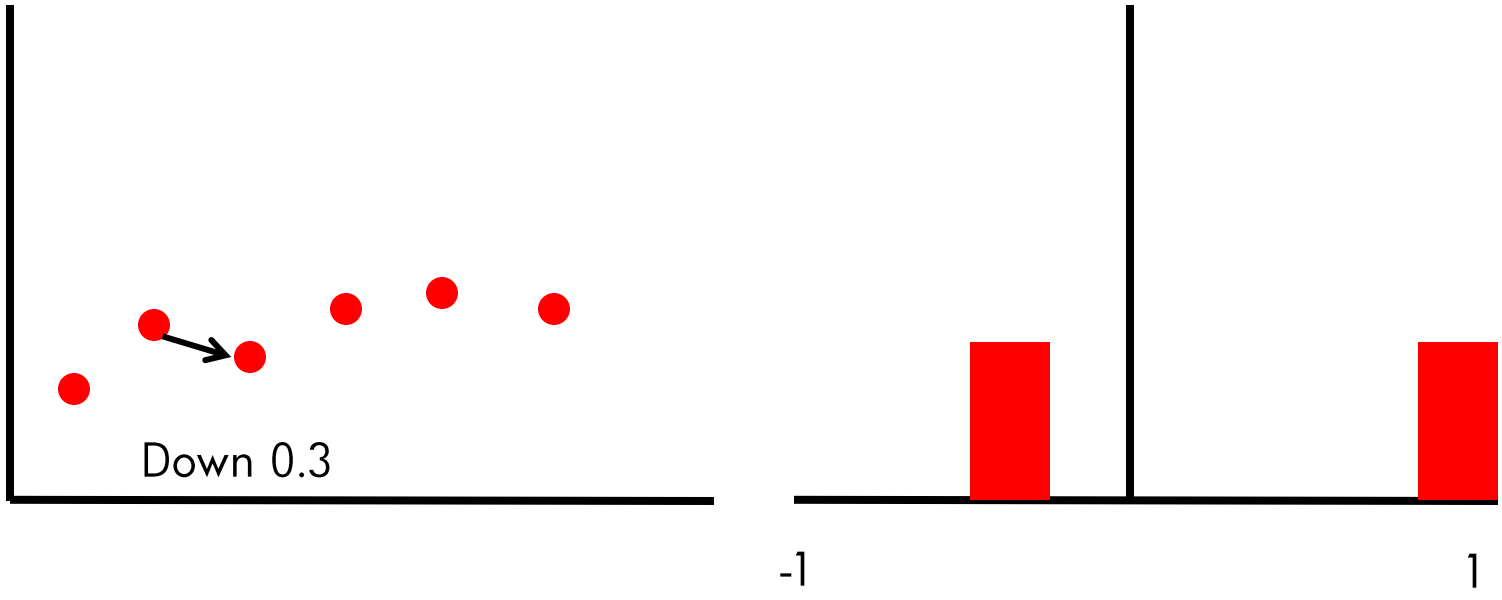
DIFFERENCING

- Often, you're not regressing on the levels, but on the relationship between Y_n and Y_{n-1}



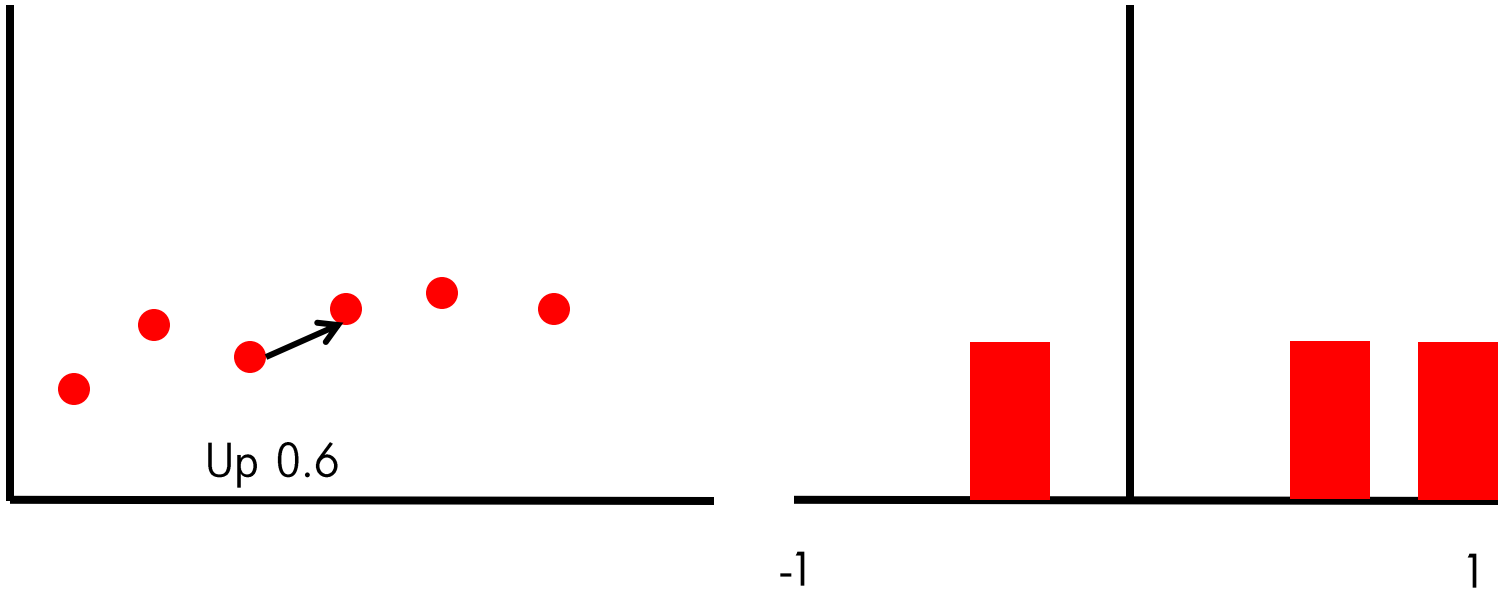
DIFFERENCING

- Often, you're not regressing on the levels, but on the relationship between Y_n and Y_{n-1}



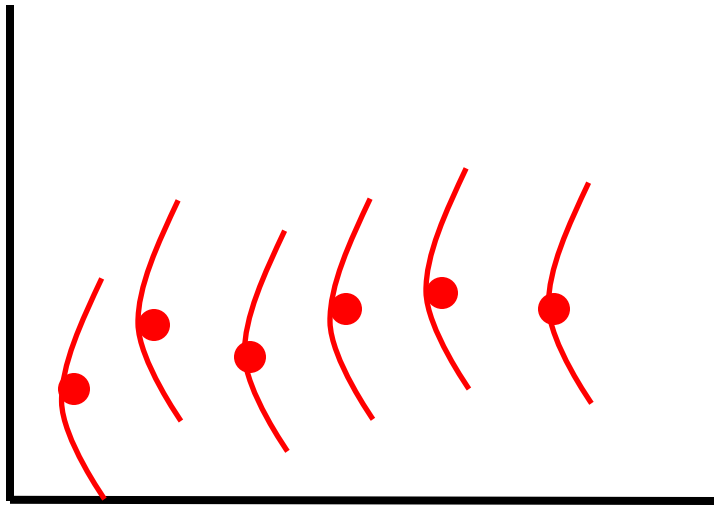
DIFFERENCING

- Often, you're not regressing on the levels, but on the relationship between Y_n and Y_{n-1}



DIFFERENCING

- You want the relationship between entries to be consistent over time.



- Each point is telling you something about the distribution of the relationship between points
- Take differences until that's stable.

Model at the level where the randomness lives, not a function of the randomness

TIME SERIES WORKHORSE

AR Integration MA

```
ARIMA[P,I,Q]
```

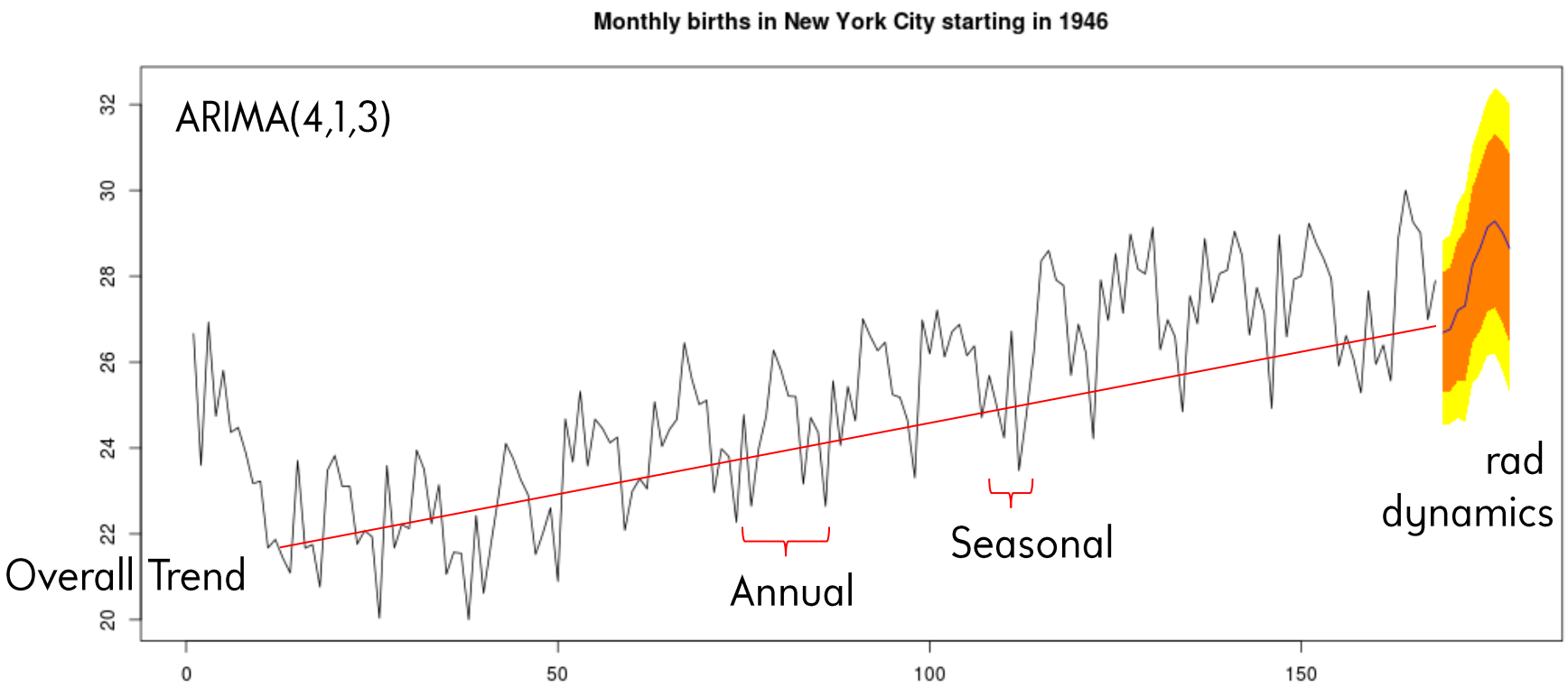
- Autoregressive Integrated Moving Average Models
- AR = the trend component
- MA = the mean-reversion component
- I = the differencing part

- The forecast is the dynamics between following the trend and going back to normal---like a spring.

INTERPRETING P AND Q

- AR(p) and MA(q) deal with how many observations back continue to impact the present---this is how you deal with seasonality.
- The last 4 periods might be a financial quarter, the last 12 a year, maybe your series depends on a rhythm of time of day.
- A high order model, like ARIMA(12,2,5) can incorporate some really rad dynamics.

BIRTHS IN NYC OVER TIME

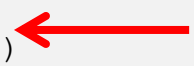


HERE'S HOW

```
> library(XML)
> library(forecast)
> births <- read.csv(file='monthly-new-york-city-births.csv', header=T,
col.names=c('month','births'))
> fit <- auto.arima(as.vector(births[,2]))
> fit
Series: as.vector(births[, 2])
ARIMA(4,1,3)

Coefficients:
         ar1     ar2      ar3      ar4      ma1     ma2     ma3
      0.4918  0.6102  -0.2806  -0.5472  -1.2276  0.0361  0.5153
s.e.  0.1142  0.1140   0.0560   0.0877   0.1165  0.2000  0.1115

sigma^2 estimated as 1.187:  log likelihood=-251.26
AIC=518.39   AICc=519.3   BIC=543.33

> forecast(fit,10)
```

HERE'S HOW

```
> library(XML)
> library(forecast)
> births <- read.csv(file='monthly-new-york-city-births.csv', header=T,
col.names=c('month','births'))
> fit <- auto.arima(as.vector(births[,2]))
> fit
Series: as.vector(births[, 2])
ARIMA(4,1,3)

Coefficients:
         ar1     ar2      ar3      ar4      ma1     ma2     ma3
      0.4918  0.6102  -0.2806  -0.5472  -1.2276  0.0361  0.5153
s.e.  0.1142  0.1140   0.0560   0.0877   0.1165  0.2000  0.1115

sigma^2 estimated as 1.187:  log likelihood=-251.26
AIC=518.39   AICc=519.3   BIC=543.33

> forecast(fit,10)
> plot(forecast(fit,10))
```



The goods

NOW YOU CAN...

- Apply the t-, F-, and χ^2-tests.
- Compute linear, logistic regressions
- Interpret regression output
- Think about time series analysis
- Forecast with ARIMA models

Next up...
* Plots!
* Plots!
* Plots!

FOR THE BREAK (If you want to)

Time Series

- Take the data we downloaded for unruly airline passengers.
- Convert it into xts
  - Strip out the passengers as.numeric()
  - Strip out the dates as.Date()
  - Set the format to yearly.
- Use an ARIMA model to forecast the next 10 years

Modeling

- Explore the iris dataset.
- Predict sepal.width based on other variables
- Compare models that use different combinations of variables.
- Which one fits best?
- Are its coefficients most significant?
- Is it also the most parsimonious?

Hint:

```
as.xts(as.numeric(X[-19,2]), order.by = as.Date(X[-19,1], format="%Y"))
```

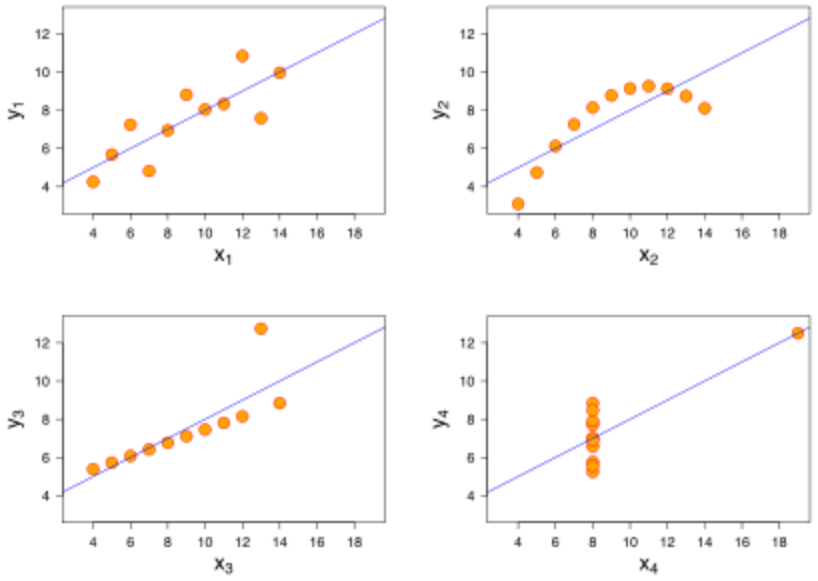
PLOTS AND GRAPHICS

Part III

FIRST OFF

- Always plot everything.

```
> anscombe
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
4   9  9  9  8  8.81 8.77  7.11  8.84
5  11 11 11  8  8.33 9.26  7.81  8.47
6  14 14 14  8  9.96 8.10  8.84  7.04
7   6  6  6  8  7.24 6.13  6.08  5.25
8   4  4  4 19  4.26 3.10  5.39 12.50
9  12 12 12  8 10.84 9.13  8.15  5.56
10  7  7  7  8  4.82 7.26  6.42  7.91
11  5  5  5  8  5.68 4.74  5.73  6.89
```



- 4 datasets with the same mean, variance, correlation

PLOTTING OVERVIEW

base-R

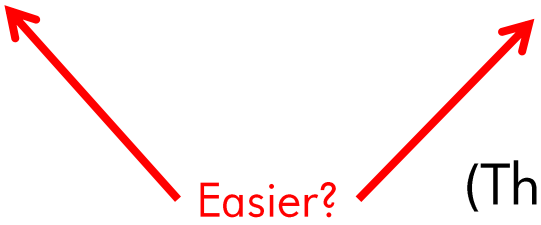
- You construct your graphic sequentially, layer upon layer.
- You construct your plot by telling it exactly what to do.

Faster

ggplot2

- You reshape and restructure your data into layers before plotting.
- You create a function of the relationships between items, and let the plot figure out the layering.

Prettier



(There's another called **lattice**)

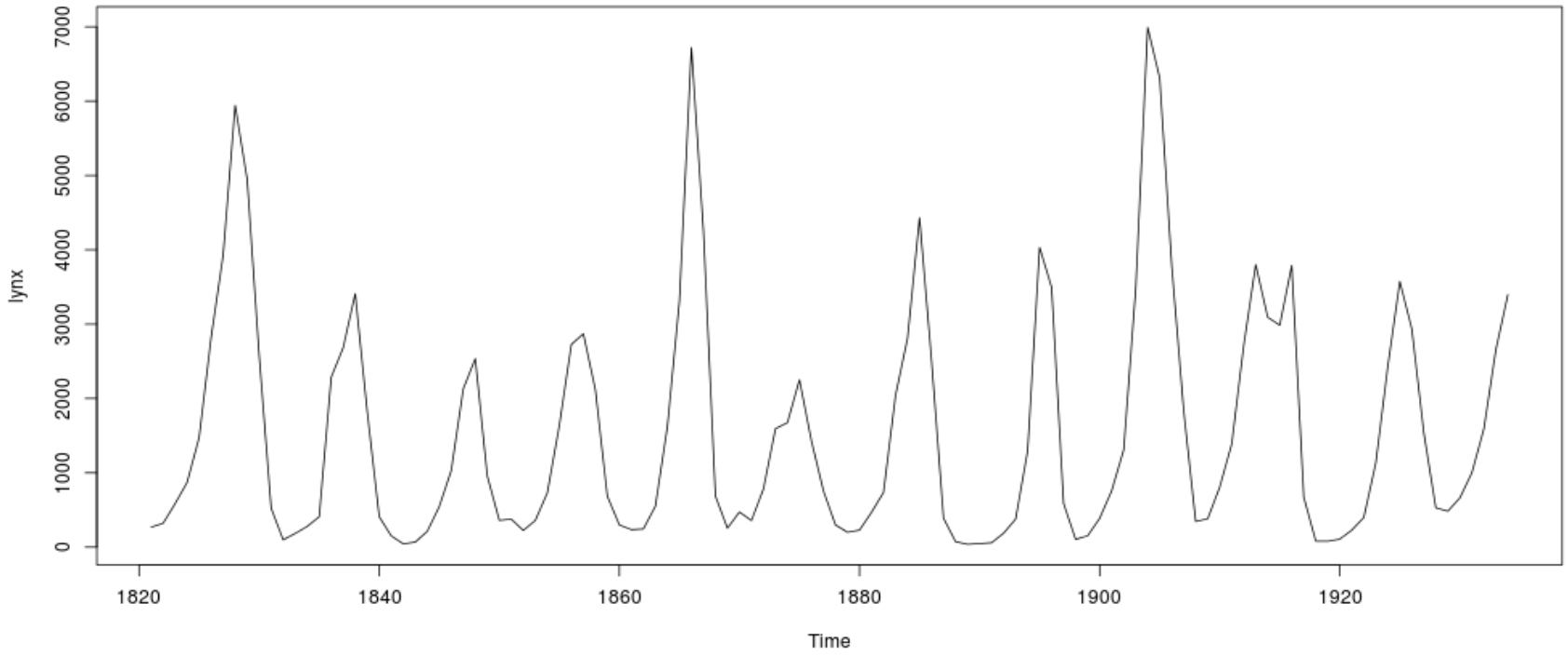
FIDDLING WITH OPTIONS

- Title
- Subtitle
- Axes
- Limits
- Marker/line types
- Colors
- Grid lines

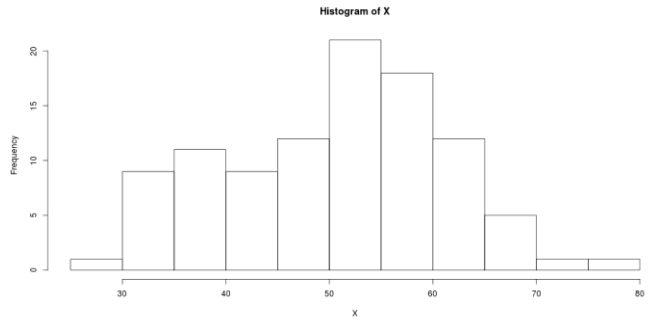
LINE GRAPH (BASE)

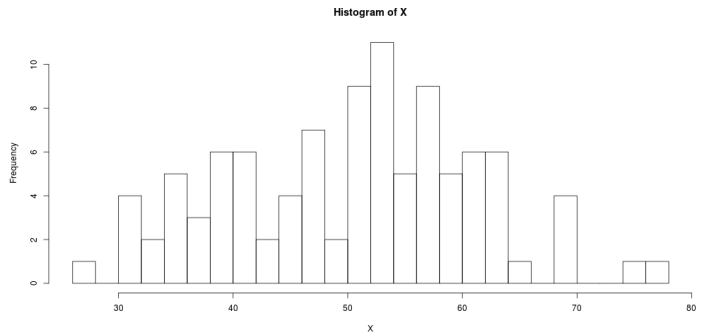
```
# all at once
> plot(lynx, type="l")
```

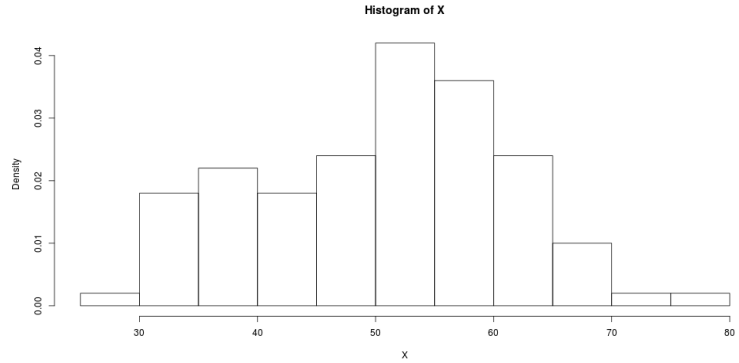
```
# add it as you go
> layout(1)
> plot(lynx)
> lines(lowess(lynx))
```



HISTOGRAM (BASE)








```
> X <- rnorm(100,50,10)
> hist(X)
```

```
# more bins
> hist(X, breaks=20)
```

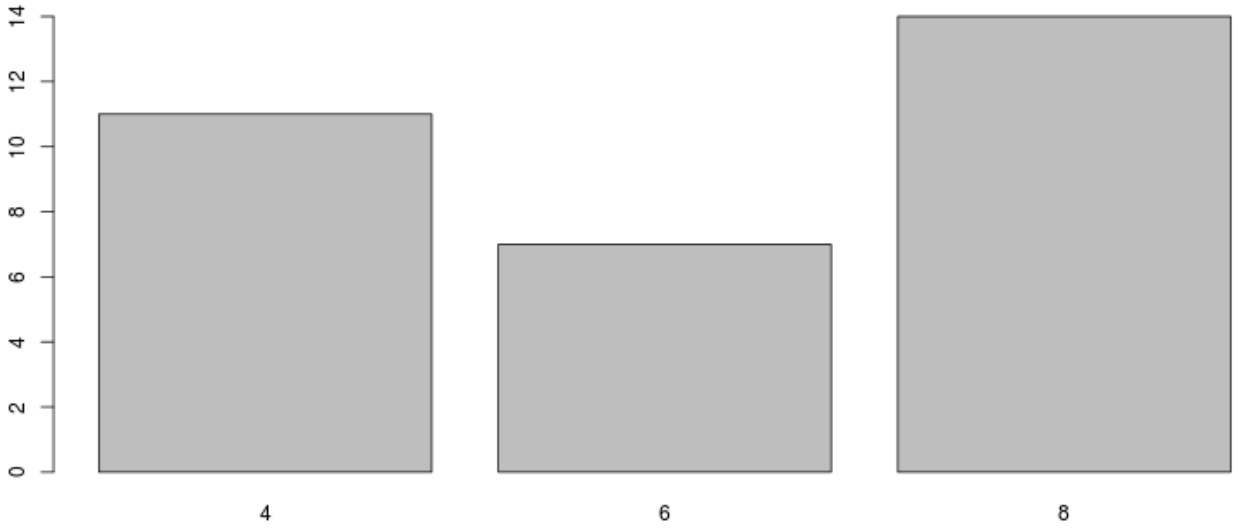
```
# density instead of count
> hist(X,freq=F)
```

GENERIC BAR GRAPH (BASE)

```
> png("carcyl.png",width=800,height=400)
> barplot(table(mtcars$cyl))
```

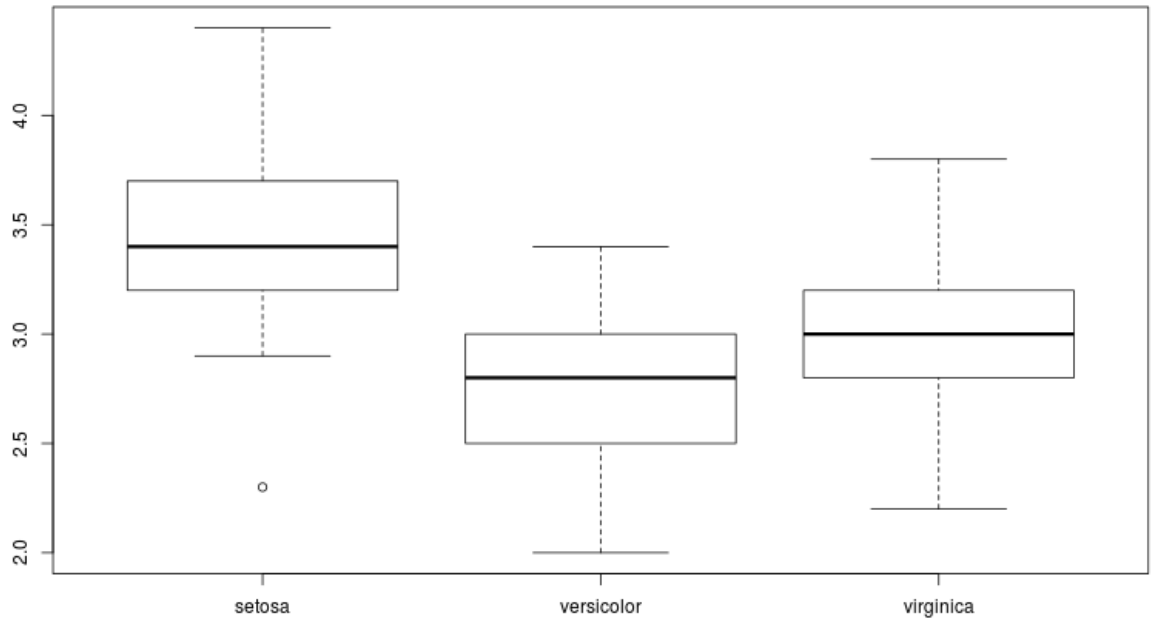
For this data, also a histogram

Function is called barplot()



BOX-AND-WHISKER (BASE)

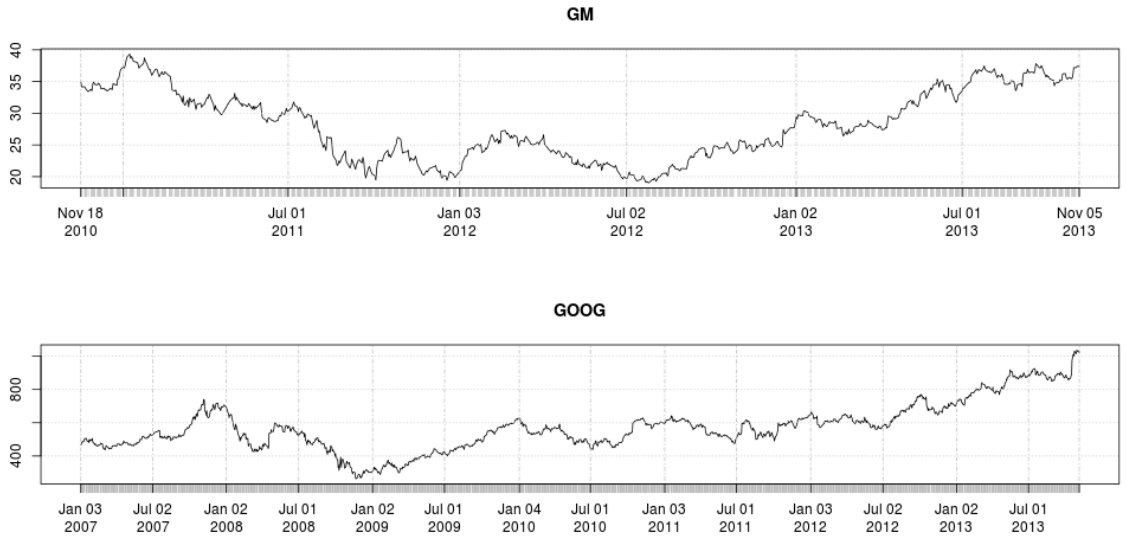
```
> boxplot(Sepal.Width ~ Species, data=iris)
```



PANELS (BASE)

- Lets get stock data from quantmod and stack GM on top of Google.

```
> library(quantmod)
> getSymbols("GM")
> getSymbols("GOOG")
> layout(matrix(c(1,2),2,1, byrow=T)
> plot(GM)
> plot(GOOG)
```

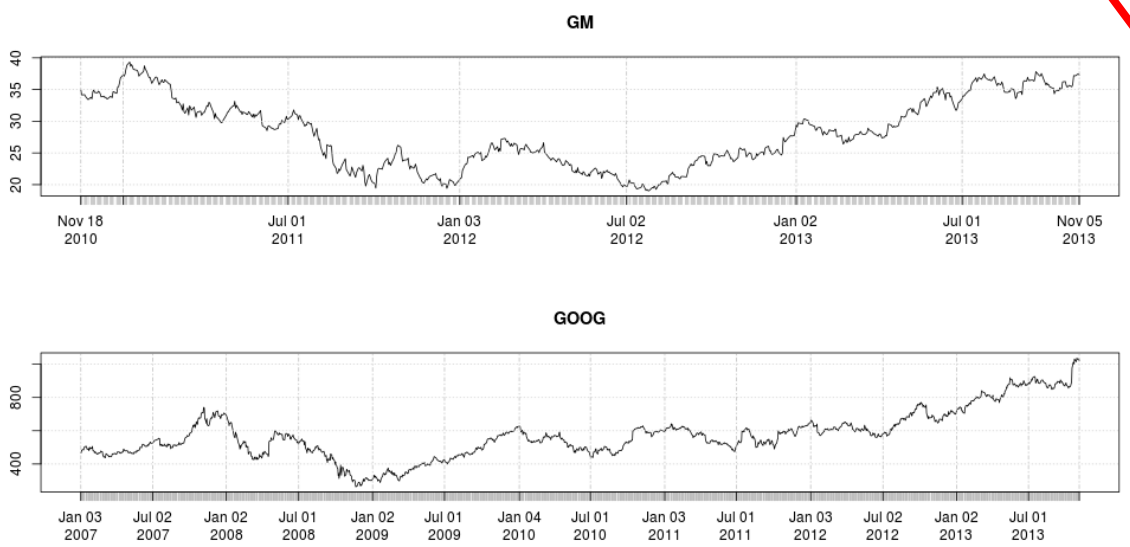


PANELS (BASE)

- Lets get stock data from quantmod and stack GM on top of Google.

```
> library(quantmod)
> getSymbols("GM")
> getSymbols("GOOG")
> layout(matrix(c(1,2),2,1, byrow=T)
> plot(GM)
> plot(GOOG)
```

Space 1 is occupied by chart 1

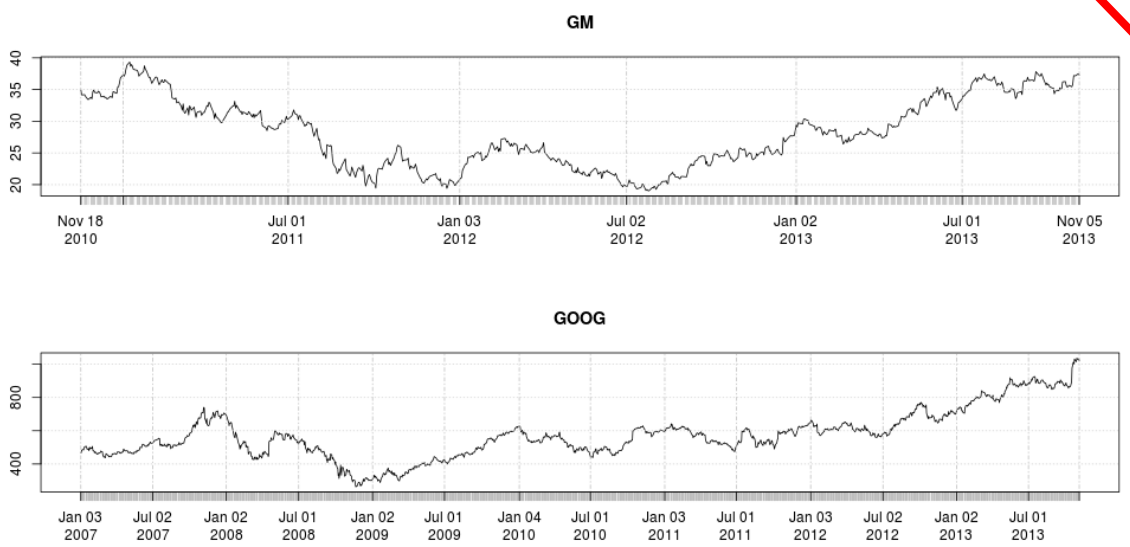


PANELS (BASE)

- Lets get stock data from quantmod and stack GM on top of Google.

```
> library(quantmod)
> getSymbols("GM")
> getSymbols("GOOG")
> layout(matrix(c(1,2),2,1, byrow=T)
> plot(GM)
> plot(GOOG)
```

Space 2 is occupied by chart 2



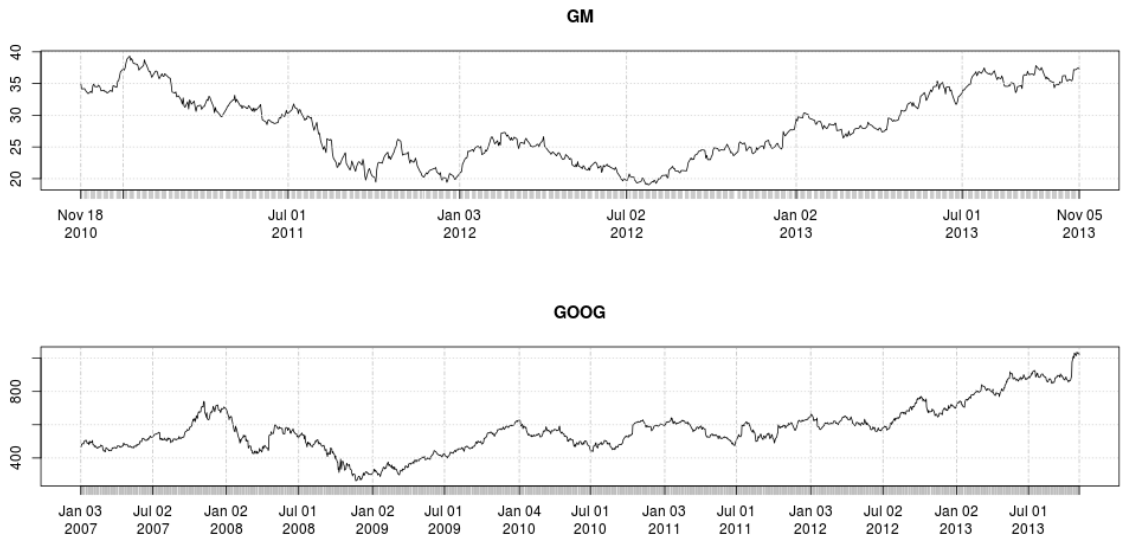
PANELS (BASE)

- Lets get stock data from quantmod and stack GM on top of Google.

```
> library(quantmod)
> getSymbols("GM")
> getSymbols("GOOG")
> layout(matrix(c(1,2),2,1, byrow=T)
> plot(GM)
> plot(GOOG)
```

There are 2 rows

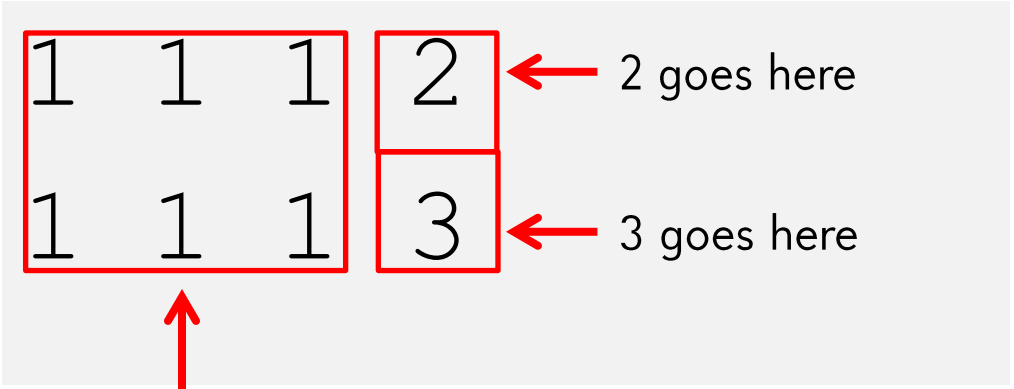
And 1 column



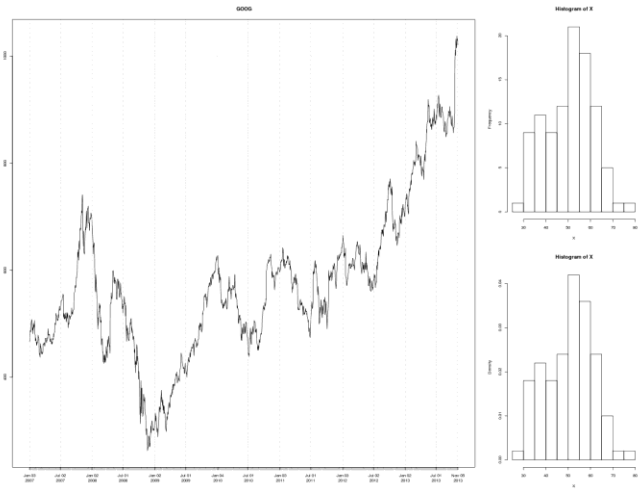
- layout() function

PANELS (BASE)

- This plot has 3 items, where plot 1 takes up spots marked 1
- You "draw" the layout this way.

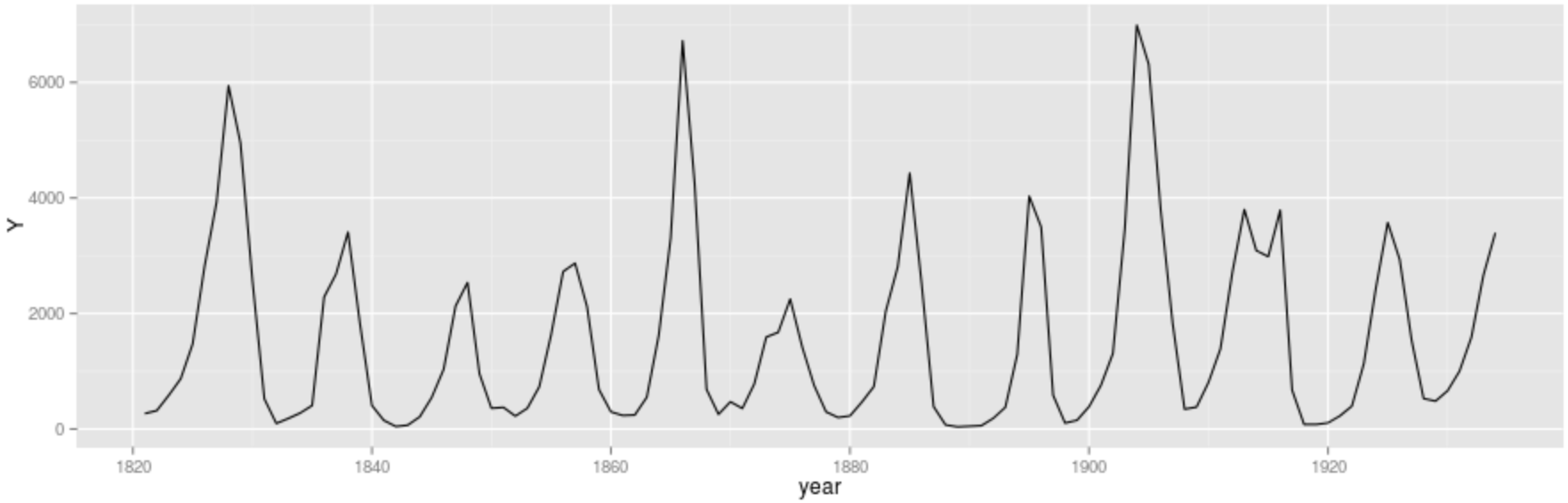



```
>
layout(matrix(c(1,1,1,2,1,1,1,3
),2,4,byrow=T))
> plot(GOOG)        # 1
> hist(X)           # 2
> hist(X, freq=F)   # 3
```



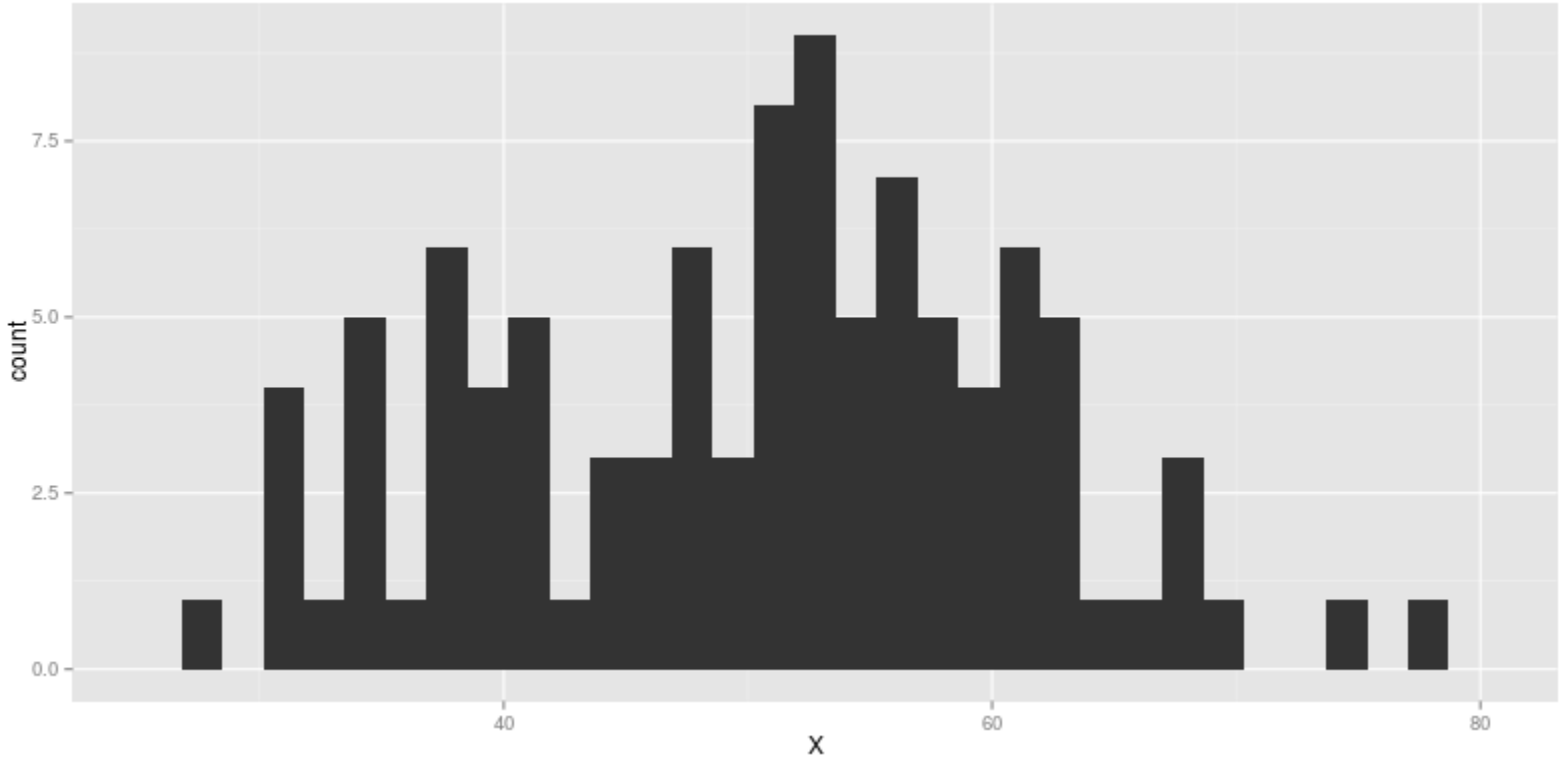
LINE GRAPH (GGPLOT)

```
# Convert to data frame.
> lynxdf <- data.frame(year=as.numeric(time(lynx)),
Y=coredata(lynx))
> qplot(x=year, y=Y, data=lynxdf, geom="line")
```



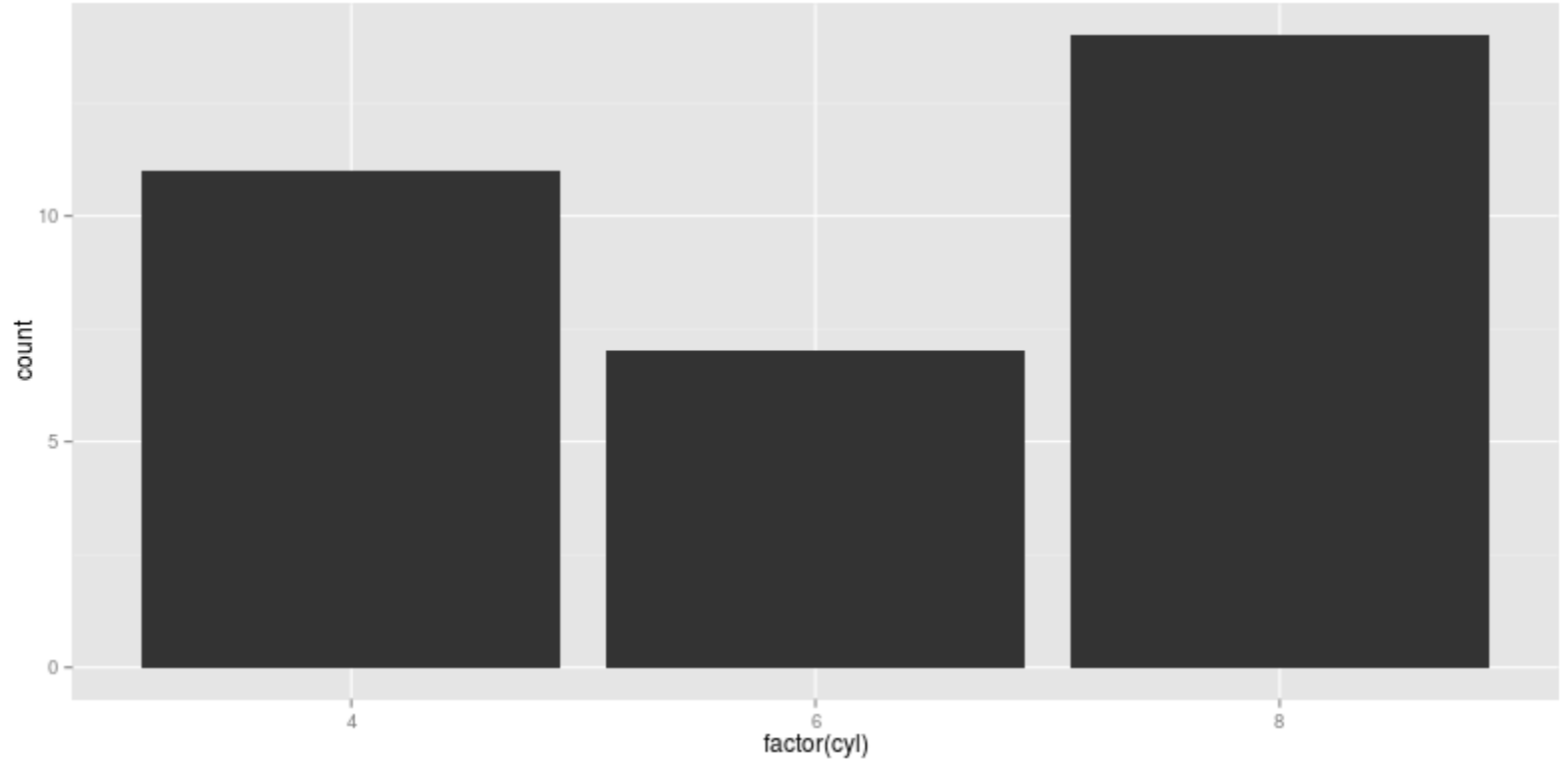
HISTOGRAM (GGPLOT)

```
# Convert to data frame.
> DF <- data.frame(X)
> ggplot(DF, aes(x=X)) + geom_histogram()
```



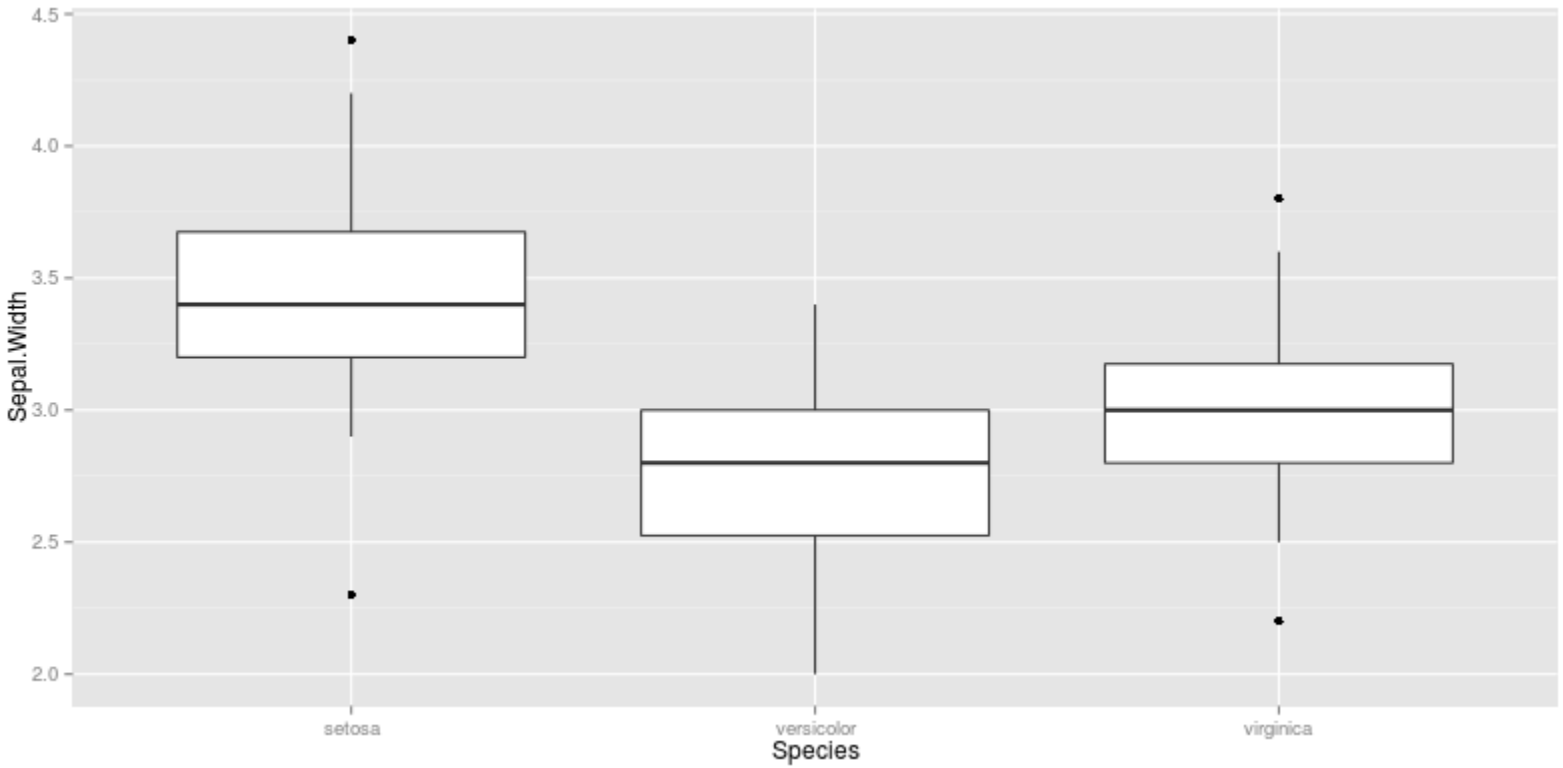
GENERIC BAR GRAPH (GGPLOT)

```
# mtcars is already a data frame
> ggplot(mtcars, aes(factor(cyl))) + geom_bar()
```



BOX-AND-WHISKER (GGPLOT)

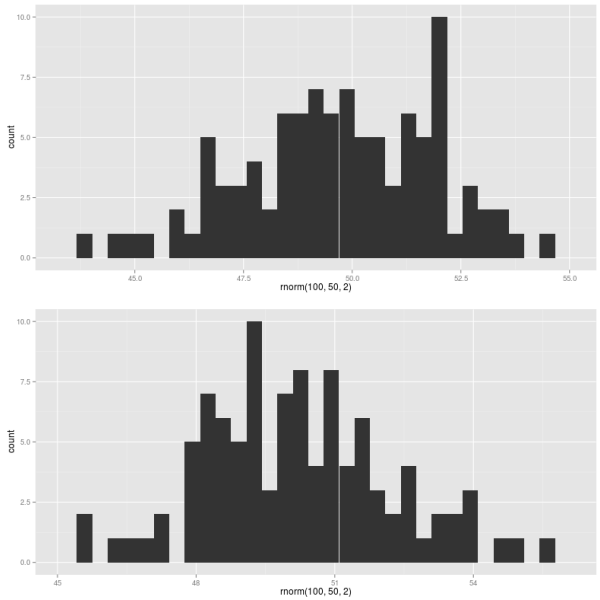
```
> ggplot(iris, aes(Species, Sepal.Width)) +
geom_boxplot()
```



PANELS (GGPLOT)

```
# Try...
> library(grid)
> library(gridExtra)
> p1 <-
qplot(rnorm(100,50,2)
)
> p2 <-
qplot(rnorm(100,50,2)
)
>
grid.arrange(p1,p2,nc
ol=1)
```

```
# If that doesn't work…
> pushViewport(viewport(layout = grid.layout(2,
1)))
> print(p1, vp=viewport(layout.pos.row =
1,layout.pos.col=1))
> print(p2, vp=viewport(layout.pos.row =
2,layout.pos.col=1))
```



NOW YOU CAN...

- Fiddle with axes, titles, and labels
- Plot in baseR or ggplot
  - line graphs
  - bar graphs
  - histograms
  - box and whiskers
  - panels

FOR THE BREAK

- Questions?
- If you want to work on something challenging, try going through the arima.R file in https://github.com/mittenchops/NYCStatsMasters/
- It incorporates:
  - indexing
  - xts
  - ARIMA time series analysis
  - ggplot plotting

Hint: yes

THANKS

- Data from public sources.
  - Built-in R
  - Time Series Data Library (Births in NYC)
  - NOAA (temperatures)
  - FAA (unruly passengers)
- Code snippets from lots of people, sorry if I missed crediting anyone
- Images from wikipedia and licensed under cc.

Want to talk more about R or stats?

adam hogan

github: @mittenchops

OTHER RESOURCES

- Websites
  - ggplot2:
    - http://docs.ggplot2.org/current/
  - baseR plots
    - http://www.statmethods.net/graphs/creating.html
  - Time series:
    - http://cran.r-project.org/web/packages/xts/vignettes/xts.pdf

OTHER RESOURCES

- Books
  - On General R, Paul Teetor:
    - http://www.amazon.com/Cookbook-OReilly-Cookbooks-Paul-Teetor/dp/0596809158/
  - On Time series, Ruey Tsay:
    - http://www.amazon.com/Analysis-Financial-Series-Probability-Statistics/dp/0471690740/
  - On Ggplot2, Hadley Wickham:
    - http://www.amazon.com/ggplot2-Elegant-Graphics-Data-Analysis/dp/0387981403/