BASICS OF DATA MUNGING IN R

adam hogan

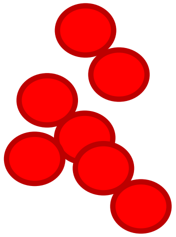
Design & Analytics

Presentation for the Chicago R-Users Group

Chicago, IL
November 14, 2012

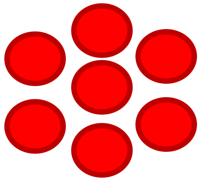
DATA MUNGING

Your data looks like this



	Adams	Allen	Anderson	Bailey
Adams		1	2	3
Allen	1			2
Anderson	2	3		
Bailey		1		2

Needs to look like this



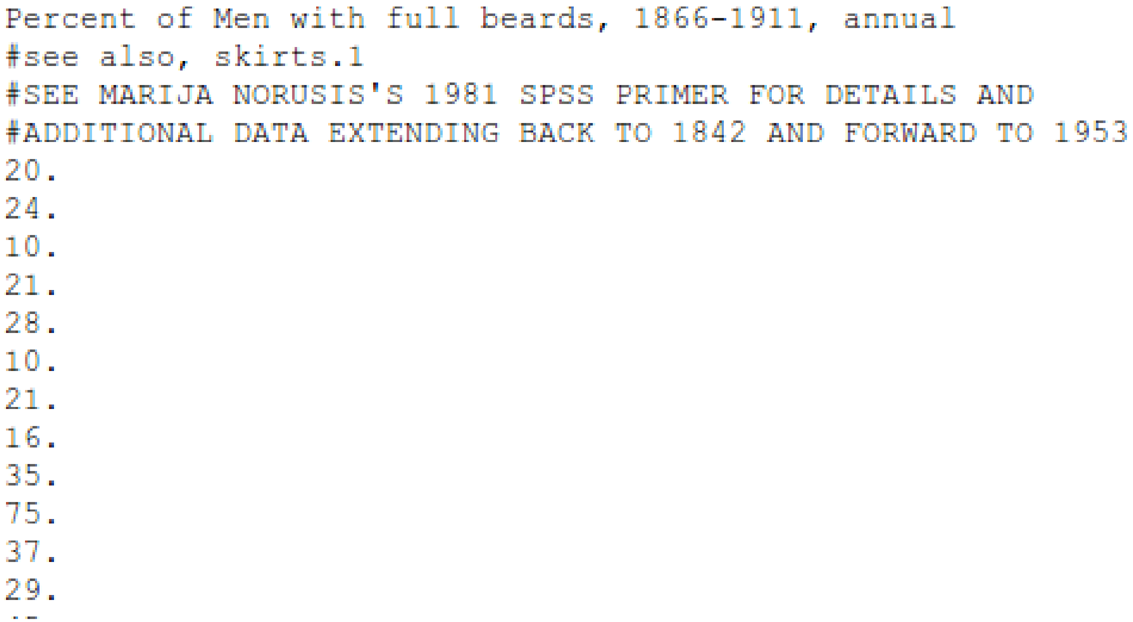
SOURCE	TARGET	WEIGHT
Adams	Allen	1
Adams	Anderson	2
Adams	Bailey	3
Allen	Adams	1
Allen	Bailey	2
Anderson	Adams	2
…	…	…

OUR GOAL

- You'll be ready to flip around R data to suit the demands of whatever package you need to work with.
- Really important stuff we won't talk about:
  - Date math
  - Probability distributions
  - Database access
  - String manipulation
  - Doing statistics

SOURCES:
LET’S GET DATA

DATA LOOKS LIKE THIS



```
Percent of Men with full beards, 1866-1911, annual
#see also, skirts.1
#SEE MARIJA NORUSIS'S 1981 SPSS PRIMER FOR DETAILS AND
#ADDITIONAL DATA EXTENDING BACK TO 1842 AND FORWARD TO 1953
20.
24.
10.
21.
28.
10.
21.
16.
35.
75.
37.
29.
```

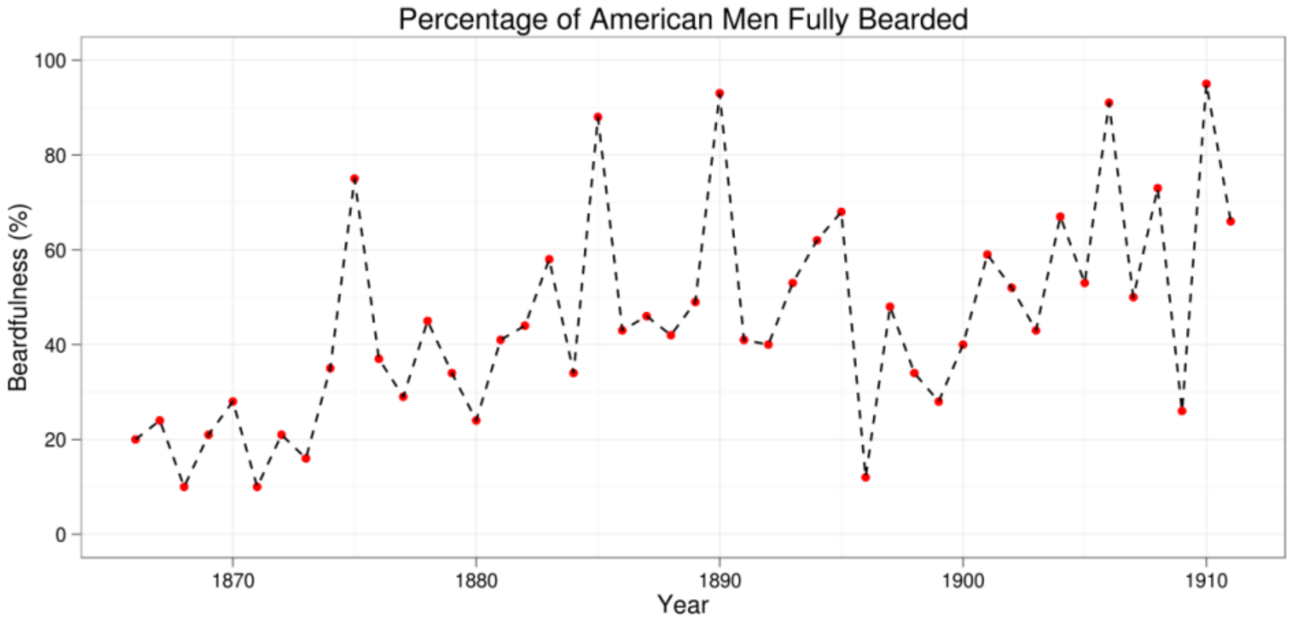
TEXT INTO R

- Base R uses the read() function for reading from CSV files and flat text files.
- read.table() and read.csv() are useful for interacting with local files.

```
> read.table()
> read.csv()

# Read a table
> url <-
'http://robjhyndman.com/tsdldata/robert
s/beards.dat'

> read.table(url, header=FALSE,skip=4)
   V1
1  20
2  24
3  10
4  21
5  28
…  …

# Read a CSV file
> Y <- read.csv(filename, header=F)
```

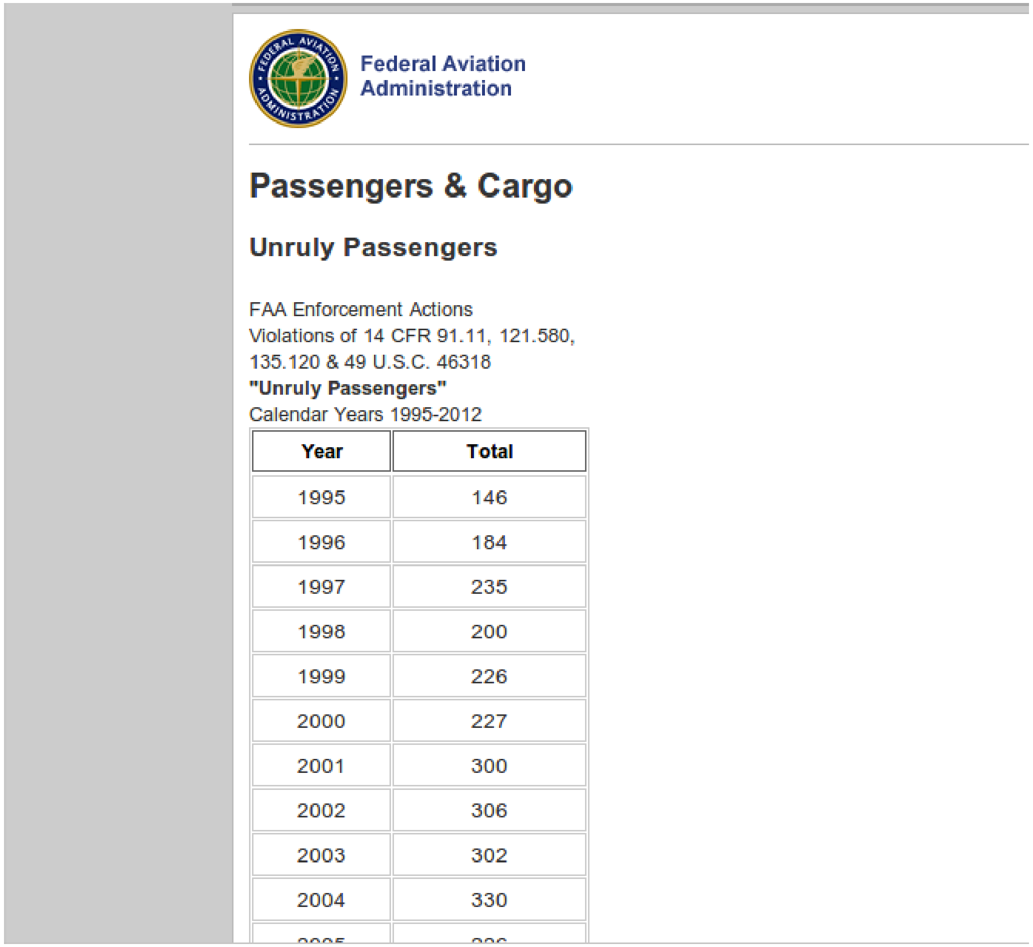
PLAIN TEXT

```
Percent of Men with full beards, 1866-1911, annual
#see also, skirts.1
#SEE MARIJA NORUSIS'S 1981 SPSS PRIMER FOR DETAILS AND
#ADDITIONAL DATA EXTENDING BACK TO 1842 AND FORWARD TO 1953
20.
24.
10.
21.
28.
10.
21.
16.
35.
75.
37.
29.
```



```
beard <- read.zoo(URL,
    header=FALSE,
    index.column=0,
    skip=4,
    FUN=function(x) as.Date(as.yearmon(x) + 1865))
```

DATA LOOKS LIKE THIS



Federal Aviation
Administration

Passengers & Cargo

Unruly Passengers

FAA Enforcement Actions
Violations of 14 CFR 91.11, 121.580,
135.120 & 49 U.S.C. 46318
"Unruly Passengers"
Calendar Years 1995-2012

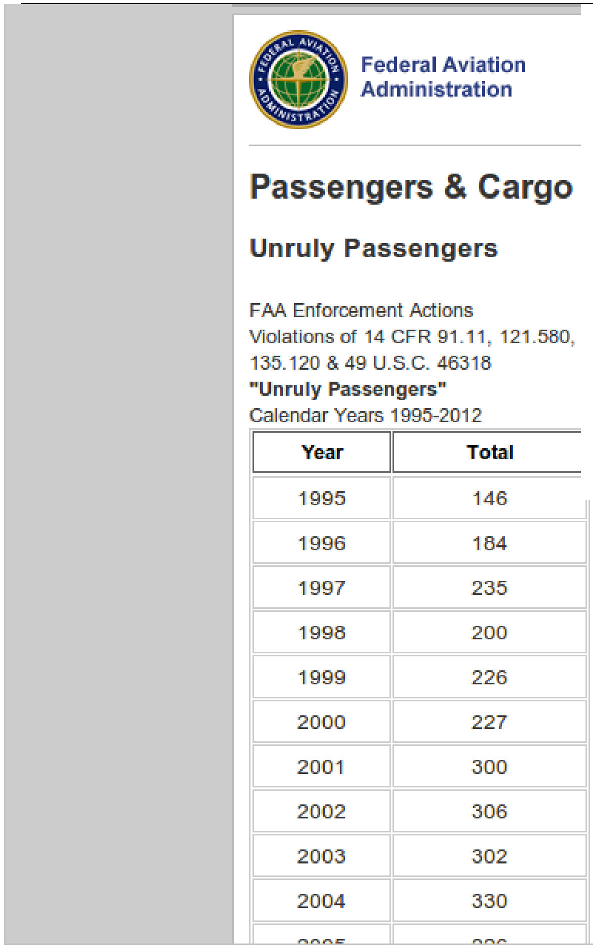
Year	Total
1995	146
1996	184
1997	235
1998	200
1999	226
2000	227
2001	300
2002	306
2003	302
2004	330
2005	226

MARKUP INTO R

```
> library(XML)
> url2 <-
‘http://www.faa.gov/data_research/passengers_cargo/un
ruly_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 12 as of April 10, 2012
```

- 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.

HTML



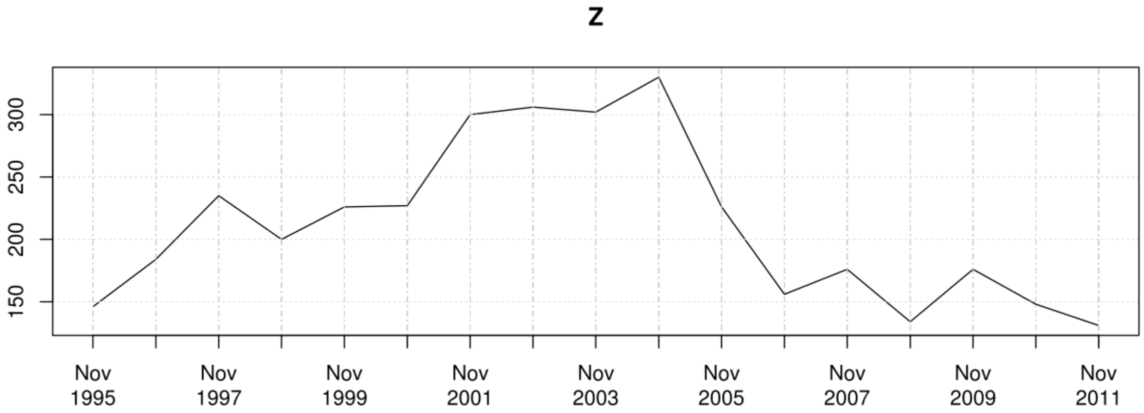
Federal Aviation Administration

Passengers & Cargo

Unruly Passengers

FAA Enforcement Actions
Violations of 14 CFR 91.11, 121.580, 135.120 & 49 U.S.C. 46318
"Unruly Passengers"
Calendar Years 1995-2012

Year	Total
1995	146
1996	184
1997	235
1998	200
1999	226
2000	227
2001	300
2002	306
2003	302
2004	330



```
Z <- as.xts(as.numeric(X[-18,2]),
                    order.by=as.Date(X[-18,1],
                                         format="%Y"))
```

FROM OTHER LANGUAGES

```
> 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

CUSTOM PACKAGES

- Specialized packages help you connect more easily to external APIs.
- More useful:
  - do.call("rbind", lapply(stats_tweets, as.data.frame))
  - We'll get to this…

```
> 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"
```

BUILDING AND EXPLORING DATA

TYPES OF DATA

- Types:
  - vector
  - matrix
  - list
  - data frame
- Other types you’ll run into are just *composites* of these simple types.
- Generally, you’ll want a matrix or a data frame.
  - 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.

```
> 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

- 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]>768.00,6]
           GOOG.Adjusted
2012-10-04        768.05
```

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
```

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

> X
[[1]]
[1] 1

$name
[1] "Fred"

$occupation
[1] "Doctor"

$age
[1] 21

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

MORE STRUCTURE

Combine columns

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

Combine rows

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

TRANSFORMING DATA

SPEAK LIKE A NATIVE

- Functional programming
  - apply
    - Apply a function to rows or columns of a matrix.
  - lapply
    - Apply a function to a list, return a list
  - sapply
    - As above, but returns vector
  - tapply
    - As above, but subset.

```
> mymatrix <-
matrix(rep(seq(2,6,by=2), 3),
ncol = 3)


> mymatrix
     [,1] [,2] [,3]
[1,]    2    2    2
[2,]    4    4    4
[3,]    6    6    6


> apply(mymatrix, 1, sum)
[1]  6 12 18


> apply(mymatrix, 2, sum)
[1] 12 12 12
```

LAPPLY

- If you find yourself writing unlist(lapply) statements, then use sapply.
- All of these functional things can be very confusing.
  - Don't worry.
  - Cheat:
  - http://vis.stanford.edu/wrangler/

```
> lapply(mymatrix[,1],sum)
[[1]]
[1] 2

[[2]]
[1] 4

[[3]]
[1] 6

> sapply(mymatrix[,1],sum)
[1] 2 4 6
```

LONG TO WIDE

```
Language Skill Users
1        R  High    10
2        R   Med    10
3        R   Low    10
4      SAS  High     1
5      SAS   Med    25
6      SAS   Low    20
```

“Long” format
is tall

“wide” format

```
Language Users.High Users.Med Users.Low
1        R       10        10        10
4      SAS        1        25        20
```

RESHAPE GYMNASTICS

- Reshape
  - Convert between long and wide
  - …and back.
- order() function

```
> df <-
data.frame(c("R","R","R","SAS","SAS","SAS"),
c("High","Med","Low","High","Med","Low"),
c(10,5,10,1,25,20)); colnames(df) <-
c("Language","Skill","Users")

> df
  Language Skill Users
1        R  High    10
2        R   Med    10
3        R   Low    10
4      SAS  High     1
5      SAS   Med    25
6      SAS   Low    20

> reshape(df, idvar="Language",
timevar="Skill", direction="wide")
  Language Users.High Users.Med Users.Low
1        R         10        10        10
4      SAS          1        25        20
```

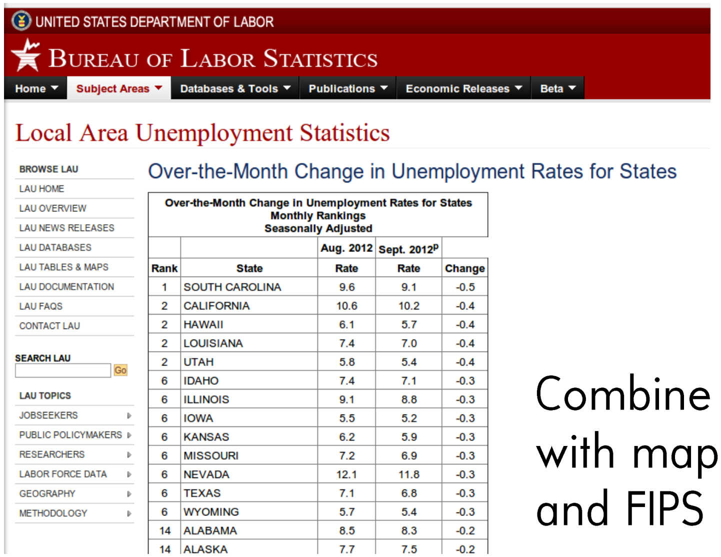
```
> reshape(df2, direction="long")
         Language Skill Users.High
R.High          R  High         10
SAS.High      SAS  High          1
R.Med           R   Med         10
SAS.Med       SAS   Med         25
R.Low           R   Low         10
SAS.Low       SAS   Low         20

> df3[order(df3$Language),]
```

NEW VARIABLES IN-PLACE

- R likes to avoid loops. Instead:
  - with()
  - Transform()
- Advanced
  - aggregate()
  - split()
  - do.call()

```
> head(mtcars)[1]
                   mpg
Mazda RX4         21.0
Mazda RX4 Wag     21.0
Datsun 710        22.8
Hornet 4 Drive    21.4
Hornet Sportabout 18.7
Valiant           18.1

> head(with(mtcars, mpg*10)) # NEW VECTOR
[1] 210 210 228 214 187 181

> head(transform(mtcars,
electricdreams=mpg*10))[c(1,12)]
                   mpg electricdreams
Mazda RX4         21.0            210
Mazda RX4 Wag     21.0            210
Datsun 710        22.8            228
Hornet 4 Drive    21.4            214
Hornet Sportabout 18.7            187
Valiant           18.1            181
```

MASH UP



UNITED STATES DEPARTMENT OF LABOR

BUREAU OF LABOR STATISTICS

Home ▾ | Subject Areas ▾ | Databases & Tools ▾ | Publications ▾ | Economic Releases ▾ | Beta ▾

Local Area Unemployment Statistics

BROWSE LAU
- LAU HOME
- LAU OVERVIEW
- LAU NEWS RELEASES
- LAU DATABASES
- LAU TABLES & MAPS
- LAU DOCUMENTATION
- LAU FAQS
- CONTACT LAU

SEARCH LAU

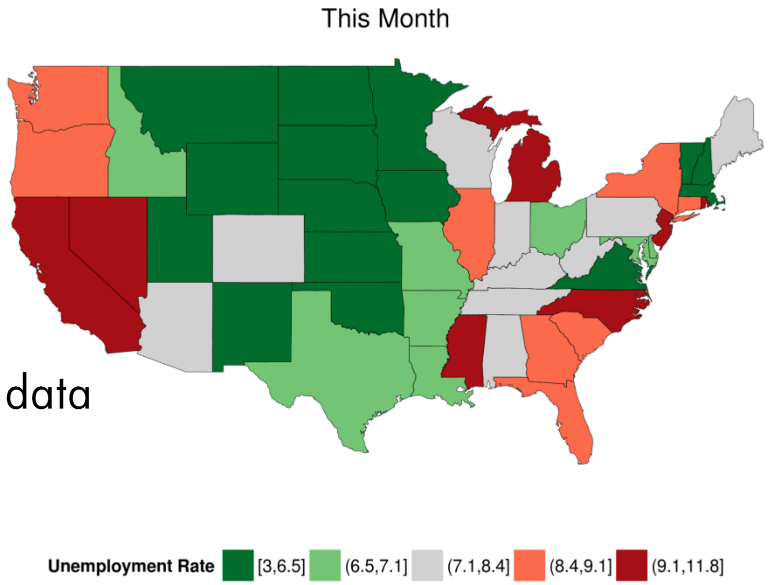
Go

LAU TOPICS
- JOBSEEKERS
- PUBLIC POLICYMAKERS
- RESEARCHERS
- LABOR FORCE DATA
- GEOGRAPHY
- METHODOLOGY

Over-the-Month Change in Unemployment Rates for States

Over-the-Month Change in Unemployment Rates for States
Monthly Rankings
Seasonally Adjusted

		Aug. 2012	Sept. 2012^{P}	
Rank	State	Rate	Rate	Change
1	SOUTH CAROLINA	9.6	9.1	-0.5
2	CALIFORNIA	10.6	10.2	-0.4
2	HAWAII	6.1	5.7	-0.4
2	LOUISIANA	7.4	7.0	-0.4
2	UTAH	5.8	5.4	-0.4
6	IDAHO	7.4	7.1	-0.3
6	ILLINOIS	9.1	8.8	-0.3
6	IOWA	5.5	5.2	-0.3
6	KANSAS	6.2	5.9	-0.3
6	MISSOURI	7.2	6.9	-0.3
6	NEVADA	12.1	11.8	-0.3
6	TEXAS	7.1	6.8	-0.3
6	WYOMING	5.7	5.4	-0.3
14	ALABAMA	8.5	8.3	-0.2
14	ALASKA	7.7	7.5	-0.2



Combine live BLS data
with map data
and FIPS codes

```
> head(unemp)
      region rank Aug. 2012 Sept. 201 change  DEV state_code State Abbreviation
1    alabama   14       8.5       8.3   -0.2  0.4         01                   AL
2     alaska   14       7.7       7.5   -0.2 -0.4         02                   AK
3    arizona   27       8.3       8.2   -0.1  0.3         04                   AZ
4   arkansas   14       7.3       7.1   -0.2 -0.8         05                   AR
5 california    2      10.6      10.2   -0.4  2.3         06                   CA
6   colorado   14       8.2       8.0   -0.2  0.1         08                   CO
```

THANKS

- Data from anonymized and public sources.
- Code snippets from lots of people, sorry if I missed crediting anyone

Want to talk more about R? Contact me.

adam hogan

adam.hogan@designandanalytics.com