# Subsetting and sorting

**Jeffrey Leek**
**Johns Hopkins Bloomberg School of Public Health**

# Subsetting - quick review

```
set.seed(13435)
X <- data.frame("var1"=sample(1:5),"var2"=sample(6:10),"var3"=sample(11:15))
X <- X[sample(1:5),]; X$var2[c(1,3)] = NA
X
```

```
  var1 var2 var3
1    2   NA   15
4    1   10   11
2    3   NA   12
3    5    6   14
5    4    9   13
```

# Subsetting - quick review

```
X[,1]
```

```
[1] 2 1 3 5 4
```

```
X[,"var1"]
```

```
[1] 2 1 3 5 4
```

```
X[1:2,"var2"]
```

```
[1] NA 10
```

# Logicals ands and ors

```
X[(X$var1 <= 3 & X$var3 > 11),]
```

```
  var1 var2 var3
1    2   NA   15
2    3   NA   12
```

```
X[(X$var1 <= 3 | X$var3 > 15),]
```

```
  var1 var2 var3
1    2   NA   15
4    1   10   11
2    3   NA   12
```

# Dealing with missing values

```
X[which(X$var2 > 8),]
```

```
  var1 var2 var3
4    1   10   11
5    4    9   13
```

# Sorting

```
sort(X$var1)
```

```
[1] 1 2 3 4 5
```

```
sort(X$var1,decreasing=TRUE)
```

```
[1] 5 4 3 2 1
```

```
sort(X$var2,na.last=TRUE)
```

```
[1]  6  9 10 NA NA
```

# Ordering

```
X[order(X$var1),]
```

```
  var1 var2 var3
4    1   10   11
1    2   NA   15
2    3   NA   12
5    4    9   13
3    5    6   14
```

# Ordering

```
X[order(X$var1,X$var3),]
```

```
  var1 var2 var3
4    1   10   11
1    2   NA   15
2    3   NA   12
5    4    9   13
3    5    6   14
```

# Ordering with plyr

```
library(plyr)
arrange(X,var1)
```

```
  var1 var2 var3
1    1   10   11
2    2   NA   15
3    3   NA   12
4    4    9   13
5    5    6   14
```

```
arrange(X,desc(var1))
```

```
  var1 var2 var3
1    5    6   14
2    4    9   13
3    3   NA   12
4    2   NA   15
```

# Adding rows and columns

```
X$var4 <- rnorm(5)
X
```

```
  var1 var2 var3     var4
1    2   NA   15  0.18760
4    1   10   11  1.78698
2    3   NA   12  0.49669
3    5    6   14  0.06318
5    4    9   13 -0.53613
```

# Adding rows and columns

```
Y <- cbind(X,rnorm(5))
Y
```

```
  var1 var2 var3     var4 rnorm(5)
1    2   NA   15  0.18760  0.62578
4    1   10   11  1.78698 -2.45084
2    3   NA   12  0.49669  0.08909
3    5    6   14  0.06318  0.47839
5    4    9   13 -0.53613  1.00053
```

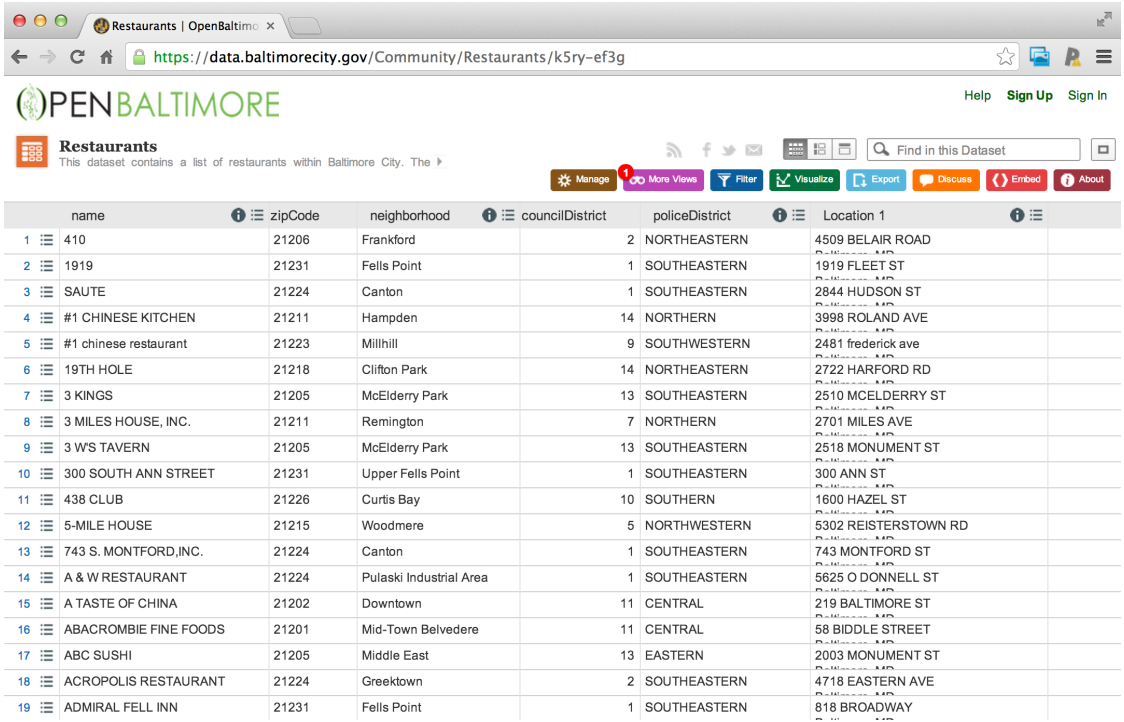
# Notes and further resources

- R programming in the Data Science Track
- Andrew Jaffe's lecture notes http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%202.pdf

# Summarizing data

**Jeffrey Leek**
**Johns Hopkins Bloomberg School of Public Health**

# Example data set




| | name | zipCode | neighborhood | councilDistrict | policeDistrict | Location 1 |
|---|---|---|---|---|---|---|
| 1 | 410 | 21206 | Frankford | 2 | NORTHEASTERN | 4509 BELAIR ROAD |
| 2 | 1919 | 21231 | Fells Point | 1 | SOUTHEASTERN | 1919 FLEET ST |
| 3 | SAUTE | 21224 | Canton | 1 | SOUTHEASTERN | 2844 HUDSON ST |
| 4 | #1 CHINESE KITCHEN | 21211 | Hampden | 14 | NORTHERN | 3998 ROLAND AVE |
| 5 | #1 chinese restaurant | 21223 | Millhill | 9 | SOUTHWESTERN | 2481 frederick ave |
| 6 | 19TH HOLE | 21218 | Clifton Park | 14 | NORTHEASTERN | 2722 HARFORD RD |
| 7 | 3 KINGS | 21205 | McElderry Park | 13 | SOUTHEASTERN | 2510 MCELDERRY ST |
| 8 | 3 MILES HOUSE, INC. | 21211 | Remington | 7 | NORTHERN | 2701 MILES AVE |
| 9 | 3 W'S TAVERN | 21205 | McElderry Park | 13 | SOUTHEASTERN | 2518 MONUMENT ST |
| 10 | 300 SOUTH ANN STREET | 21231 | Upper Fells Point | 1 | SOUTHEASTERN | 300 ANN ST |
| 11 | 438 CLUB | 21226 | Curtis Bay | 10 | SOUTHERN | 1600 HAZEL ST |
| 12 | 5-MILE HOUSE | 21215 | Woodmere | 5 | NORTHWESTERN | 5302 REISTERSTOWN RD |
| 13 | 743 S. MONTFORD,INC. | 21224 | Canton | 1 | SOUTHEASTERN | 743 MONTFORD ST |
| 14 | A & W RESTAURANT | 21224 | Pulaski Industrial Area | 1 | SOUTHEASTERN | 5625 O DONNELL ST |
| 15 | A TASTE OF CHINA | 21202 | Downtown | 11 | CENTRAL | 219 BALTIMORE ST |
| 16 | ABACROMBIE FINE FOODS | 21201 | Mid-Town Belvedere | 11 | CENTRAL | 58 BIDDLE STREET |
| 17 | ABC SUSHI | 21205 | Middle East | 13 | EASTERN | 2003 MONUMENT ST |
| 18 | ACROPOLIS RESTAURANT | 21224 | Greektown | 2 | SOUTHEASTERN | 4718 EASTERN AVE |
| 19 | ADMIRAL FELL INN | 21231 | Fells Point | 1 | SOUTHEASTERN | 818 BROADWAY |

https://data.baltimorecity.gov/Community/Restaurants/k5ry-ef3g

# Getting the data from the web

```
if(!file.exists("./data")){dir.create("./data")}
fileUrl <- "https://data.baltimorecity.gov/api/views/k5ry-ef3g/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl,destfile="./data/restaurants.csv",method="curl")
restData <- read.csv("./data/restaurants.csv")
```

# Look at a bit of the data

```
head(restData,n=3)
```

```
   name zipCode neighborhood councilDistrict policeDistrict                        Location.1
1   410   21206    Frankford               2   NORTHEASTERN 4509 BELAIR ROAD\nBaltimore, MD\n
2  1919   21231  Fells Point               1   SOUTHEASTERN    1919 FLEET ST\nBaltimore, MD\n
3 SAUTE   21224       Canton               1   SOUTHEASTERN   2844 HUDSON ST\nBaltimore, MD\n
```

```
tail(restData,n=3)
```

```
                name zipCode  neighborhood councilDistrict policeDistrict
1325 ZINK'S CAF\u0090   21213 Belair-Edison              13   NORTHEASTERN
1326     ZISSIMOS BAR   21211       Hampden               7       NORTHERN
1327           ZORBAS   21224     Greektown               2   SOUTHEASTERN
                           Location.1
1325 3300 LAWNVIEW AVE\nBaltimore, MD\n
1326      1023 36TH ST\nBaltimore, MD\n
1327  4710 EASTERN Ave\nBaltimore, MD\n
```

# Make summary

```
summary(restData)
```

```
                          name         zipCode              neighborhood councilDistrict
MCDONALD'S                  :   8   Min.   :-21226   Downtown    :128   Min.   : 1.00
POPEYES FAMOUS FRIED CHICKEN:   7   1st Qu.: 21202   Fells Point : 91   1st Qu.: 2.00
SUBWAY                      :   6   Median : 21218   Inner Harbor: 89   Median : 9.00
KENTUCKY FRIED CHICKEN      :   5   Mean   : 21185   Canton      : 81   Mean   : 7.19
BURGER KING                 :   4   3rd Qu.: 21226   Federal Hill: 42   3rd Qu.:11.00
DUNKIN DONUTS               :   4   Max.   : 21287   Mount Vernon: 33   Max.   :14.00
(Other)                     :1293                    (Other)     :863
     policeDistrict                          Location.1
SOUTHEASTERN:385    1101 RUSSELL ST\nBaltimore, MD\n:   9
CENTRAL     :288    201 PRATT ST\nBaltimore, MD\n   :   8
SOUTHERN    :213    2400 BOSTON ST\nBaltimore, MD\n :   8
NORTHERN    :157    300 LIGHT ST\nBaltimore, MD\n   :   5
NORTHEASTERN: 72    300 CHARLES ST\nBaltimore, MD\n :   4
EASTERN     : 67    301 LIGHT ST\nBaltimore, MD\n   :   4
(Other)     :145    (Other)                         :1289
```

# Mpre in depth information

```
str(restData)
```

```
'data.frame':   1327 obs. of  6 variables:
 $ name           : Factor w/ 1277 levels "#1 CHINESE KITCHEN",..: 9 3 992 1 2 4 5 6 7 8 ...
 $ zipCode        : int  21206 21231 21224 21211 21223 21218 21205 21211 21205 21231 ...
 $ neighborhood   : Factor w/ 173 levels "Abell","Arlington",..: 53 52 18 66 104 33 98 133 98 157 ...
 $ councilDistrict: int  2 1 1 14 9 14 13 7 13 1 ...
 $ policeDistrict : Factor w/ 9 levels "CENTRAL","EASTERN",..: 3 6 6 4 8 3 6 4 6 6 ...
 $ Location.1     : Factor w/ 1210 levels "1 BIDDLE ST\nBaltimore, MD\n",..: 835 334 554 755 492 537 5
```

# Quantiles of quantitative variables

```
quantile(restData$councilDistrict,na.rm=TRUE)
```

```
  0%  25%  50%  75% 100% 
   1    2    9   11   14 
```

```
quantile(restData$councilDistrict,probs=c(0.5,0.75,0.9))
```

```
50% 75% 90% 
  9  11  12 
```

# Make table

```
table(restData$zipCode,useNA="ifany")
```

```
-21226  21201  21202  21205  21206  21207  21208  21209  21210  21211  21212  21213  21214  21215
     1    136    201     27     30      4      1      8     23     41     28     31     17     54
 21216  21217  21218  21220  21222  21223  21224  21225  21226  21227  21229  21230  21231  21234
    10     32     69      1      7     56    199     19     18      4     13    156    127      7
 21237  21239  21251  21287
     1      3      2      1
```

# Make table

```
table(restData$councilDistrict,restData$zipCode)
```

```
   -21226 21201 21202 21205 21206 21207 21208 21209 21210 21211 21212 21213 21214 21215 21216
1       0     0    37     0     0     0     0     0     0     0     0     2     0     0     0
2       0     0     0     3    27     0     0     0     0     0     0     0     0     0     0
3       0     0     0     0     0     0     0     0     0     0     0     2    17     0     0
4       0     0     0     0     0     0     0     0     0     0    27     0     0     0     0
5       0     0     0     0     0     3     0     6     0     0     0     0     0    31     0
6       0     0     0     0     0     0     0     1    19     0     0     0     0    15     1
7       0     0     0     0     0     0     0     1     0    27     0     0     0     6     7
8       0     0     0     0     0     1     0     0     0     0     0     0     0     0     0
9       0     1     0     0     0     0     0     0     0     0     0     0     0     0     2
10      1     0     1     0     0     0     0     0     0     0     0     0     0     0     0
11      0   115   139     0     0     0     1     0     0     0     1     0     0     0     0
12      0    20    24     4     0     0     0     0     0     0     0    13     0     0     0
13      0     0     0    20     3     0     0     0     0     0     0    13     0     1     0
14      0     0     0     0     0     0     0     0     4    14     0     1     0     1     0

   21217 21218 21220 21222 21223 21224 21225 21226 21227 21229 21230 21231 21234 21237 21239
```

# Check for missing values

```
sum(is.na(restData$councilDistrict))
```

```
[1] 0
```

```
any(is.na(restData$councilDistrict))
```

```
[1] FALSE
```

```
all(restData$zipCode > 0)
```

```
[1] FALSE
```

# Row and column sums

```
colSums(is.na(restData))
```

| name | zipCode | neighborhood | councilDistrict | policeDistrict | Location.1 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |

```
all(colSums(is.na(restData))==0)
```

```
[1] TRUE
```

# Values with specific characteristics

```
table(restData$zipCode %in% c("21212"))
```

```
FALSE  TRUE
 1299    28
```

```
table(restData$zipCode %in% c("21212","21213"))
```

```
FALSE  TRUE
 1268    59
```

# Values with specific characteristics

```
restData[restData$zipCode %in% c("21212","21213"),]
```

| | name | zipCode | neighborhood | councilDistrict |
|---|---|---|---|---|
| 29 | BAY ATLANTIC CLUB | 21212 | Downtown | 11 |
| 39 | BERMUDA BAR | 21213 | Broadway East | 12 |
| 92 | ATWATER'S | 21212 | Chinquapin Park-Belvedere | 4 |
| 111 | BALTIMORE ESTONIAN SOCIETY | 21213 | South Clifton Park | 12 |
| 187 | CAFE ZEN | 21212 | Rosebank | 4 |
| 220 | CERIELLO FINE FOODS | 21212 | Chinquapin Park-Belvedere | 4 |
| 266 | CLIFTON PARK GOLF COURSE SNACK BAR | 21213 | Darley Park | 14 |
| 276 | CLUB HOUSE BAR & GRILL | 21213 | Orangeville Industrial Area | 13 |
| 289 | CLUBHOUSE BAR & GRILL | 21213 | Orangeville Industrial Area | 13 |
| 291 | COCKY LOU'S | 21213 | Broadway East | 12 |
| 362 | DREAM TAVERN, CARRIBEAN U.S.A. | 21213 | Broadway East | 13 |
| 373 | DUNKIN DONUTS | 21212 | Homeland | 4 |
| 383 | EASTSIDE SPORTS SOCIAL CLUB | 21213 | Broadway East | 13 |
| 417 | FIELDS OLD TRAIL | 21212 | Mid-Govans | 4 |
| 475 | GRAND CRU | 21212 | Chinquapin Park-Belvedere | 4 |
| 545 | RANDY'S BAR | 21213 | Broadway East | 12 |
| 604 | MURPHY'S NEIGHBORHOOD BAR & GRILL | 21212 | Mid-Govans | 4 |

# Cross tabs

```
data(UCBAdmissions)
DF = as.data.frame(UCBAdmissions)
summary(DF)
```

```
     Admit       Gender   Dept       Freq
 Admitted:12   Male  :12   A:4   Min.   :  8
 Rejected:12   Female:12   B:4   1st Qu.: 80
                           C:4   Median :170
                           D:4   Mean   :189
                           E:4   3rd Qu.:302
                           F:4   Max.   :512
```

# Cross tabs

```
xt <- xtabs(Freq ~ Gender + Admit,data=DF)
xt
```

```
        Admit
Gender   Admitted Rejected
  Male       1198     1493
  Female      557     1278
```

# Flat tables

```
warpbreaks$replicate <- rep(1:9, len = 54)
xt = xtabs(breaks ~.,data=warpbreaks)
xt
```

```
, , replicate = 1

    tension
wool  L  M  H
   A 26 18 36
   B 27 42 20

, , replicate = 2

    tension
wool  L  M  H
   A 30 21 21
   B 14 26 21

, , replicate = 3
```

# Flat tables

```
ftable(xt)
```

```
            replicate  1  2  3  4  5  6  7  8  9
wool tension
A    L                26 30 54 25 70 52 51 26 67
     M                18 21 29 17 12 18 35 30 36
     H                36 21 24 18 10 43 28 15 26
B    L                27 14 29 19 29 31 41 20 44
     M                42 26 19 16 39 28 21 39 29
     H                20 21 24 17 13 15 15 16 28
```

# Size of a data set

```
fakeData = rnorm(1e5)
object.size(fakeData)
```

```
800040 bytes
```

```
print(object.size(fakeData),units="Mb")
```

```
0.8 Mb
```

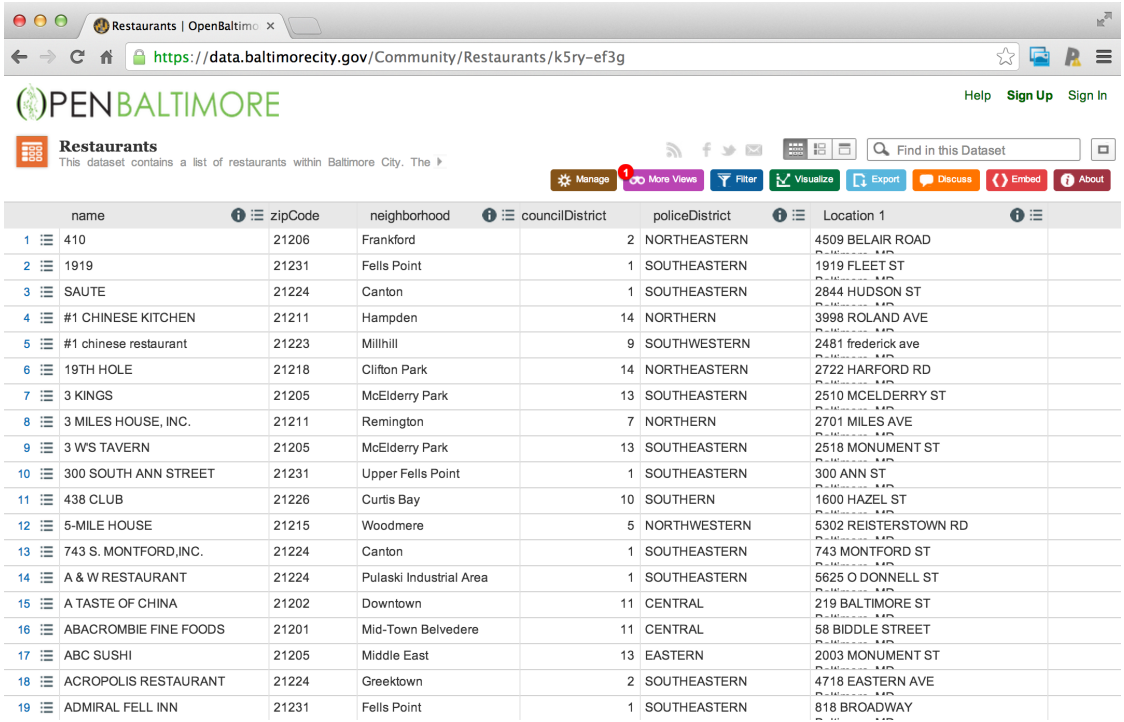
# Creating new variables

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

# Why create new variables?

- Often the raw data won't have a value you are looking for
- You will need to transform the data to get the values you would like
- Usually you will add those values to the data frames you are working with
- Common variables to create
    - Missingness indicators
    - "Cutting up" quantitative variables
    - Applying transforms

# Example data set




| | name | zipCode | neighborhood | councilDistrict | policeDistrict | Location 1 |
|---|---|---|---|---|---|---|
| 1 | 410 | 21206 | Frankford | 2 | NORTHEASTERN | 4509 BELAIR ROAD |
| 2 | 1919 | 21231 | Fells Point | 1 | SOUTHEASTERN | 1919 FLEET ST |
| 3 | SAUTE | 21224 | Canton | 1 | SOUTHEASTERN | 2844 HUDSON ST |
| 4 | #1 CHINESE KITCHEN | 21211 | Hampden | 14 | NORTHERN | 3998 ROLAND AVE |
| 5 | #1 chinese restaurant | 21223 | Millhill | 9 | SOUTHWESTERN | 2481 frederick ave |
| 6 | 19TH HOLE | 21218 | Clifton Park | 14 | NORTHEASTERN | 2722 HARFORD RD |
| 7 | 3 KINGS | 21205 | McElderry Park | 13 | SOUTHEASTERN | 2510 MCELDERRY ST |
| 8 | 3 MILES HOUSE, INC. | 21211 | Remington | 7 | NORTHERN | 2701 MILES AVE |
| 9 | 3 W'S TAVERN | 21205 | McElderry Park | 13 | SOUTHEASTERN | 2518 MONUMENT ST |
| 10 | 300 SOUTH ANN STREET | 21231 | Upper Fells Point | 1 | SOUTHEASTERN | 300 ANN ST |
| 11 | 438 CLUB | 21226 | Curtis Bay | 10 | SOUTHERN | 1600 HAZEL ST |
| 12 | 5-MILE HOUSE | 21215 | Woodmere | 5 | NORTHWESTERN | 5302 REISTERSTOWN RD |
| 13 | 743 S. MONTFORD,INC. | 21224 | Canton | 1 | SOUTHEASTERN | 743 MONTFORD ST |
| 14 | A & W RESTAURANT | 21224 | Pulaski Industrial Area | 1 | SOUTHEASTERN | 5625 O DONNELL ST |
| 15 | A TASTE OF CHINA | 21202 | Downtown | 11 | CENTRAL | 219 BALTIMORE ST |
| 16 | ABACROMBIE FINE FOODS | 21201 | Mid-Town Belvedere | 11 | CENTRAL | 58 BIDDLE STREET |
| 17 | ABC SUSHI | 21205 | Middle East | 13 | EASTERN | 2003 MONUMENT ST |
| 18 | ACROPOLIS RESTAURANT | 21224 | Greektown | 2 | SOUTHEASTERN | 4718 EASTERN AVE |
| 19 | ADMIRAL FELL INN | 21231 | Fells Point | 1 | SOUTHEASTERN | 818 BROADWAY |

https://data.baltimorecity.gov/Community/Restaurants/k5ry-ef3g

# Getting the data from the web

```
if(!file.exists("./data")){dir.create("./data")}
fileUrl <- "https://data.baltimorecity.gov/api/views/k5ry-ef3g/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl,destfile="./data/restaurants.csv",method="curl")
restData <- read.csv("./data/restaurants.csv")
```

# Creating sequences

*Sometimes you need an index for your data set*

```
s1 <- seq(1,10,by=2) ; s1
```

```
[1] 1 3 5 7 9
```

```
s2 <- seq(1,10,length=3); s2
```

```
[1]  1.0  5.5 10.0
```

```
x <- c(1,3,8,25,100); seq(along = x)
```

```
[1] 1 2 3 4 5
```

# Subsetting variables

```
restData$nearMe = restData$neighborhood %in% c("Roland Park", "Homeland")
table(restData$nearMe)
```

```

FALSE  TRUE 
 1314    13 
```

# Creating binary variables

```
restData$zipWrong = ifelse(restData$zipCode < 0, TRUE, FALSE)
table(restData$zipWrong,restData$zipCode < 0)
```

```
       FALSE TRUE
  FALSE  1326    0
  TRUE      0    1
```

# Creating categorical variables

```
restData$zipGroups = cut(restData$zipCode,breaks=quantile(restData$zipCode))
table(restData$zipGroups)
```

```
(-2.123e+04,2.12e+04]  (2.12e+04,2.122e+04] (2.122e+04,2.123e+04] (2.123e+04,2.129e+04]
                  337                   375                   282                   332
```

```
table(restData$zipGroups,restData$zipCode)
```

```
                     -21226 21201 21202 21205 21206 21207 21208 21209 21210 21211 21212 21213
  (-2.123e+04,2.12e+04]      0   136   201     0     0     0     0     0     0     0     0     0
  (2.12e+04,2.122e+04]       0     0     0    27    30     4     1     8    23    41    28    31
  (2.122e+04,2.123e+04]      0     0     0     0     0     0     0     0     0     0     0     0
  (2.123e+04,2.129e+04]      0     0     0     0     0     0     0     0     0     0     0     0

                     21214 21215 21216 21217 21218 21220 21222 21223 21224 21225 21226 21227
```

# Easier cutting

```
library(Hmisc)
restData$zipGroups = cut2(restData$zipCode,g=4)
table(restData$zipGroups)
```

```
[-21226,21205) [ 21205,21220) [ 21220,21227) [ 21227,21287]
           338            375            300            314
```

# Creating factor variables

```
restData$zcf <- factor(restData$zipCode)
restData$zcf[1:10]
```

```
 [1] 21206 21231 21224 21211 21223 21218 21205 21211 21205 21231
32 Levels: -21226 21201 21202 21205 21206 21207 21208 21209 21210 21211 21212 21213 21214 ... 21287
```

```
class(restData$zcf)
```

```
[1] "factor"
```

# Levels of factor variables

```
yesno <- sample(c("yes","no"),size=10,replace=TRUE)
yesnofac = factor(yesno,levels=c("yes","no"))
relevel(yesnofac,ref="yes")
```

```
 [1] yes yes yes yes no  yes yes yes no  no
Levels: yes no
```

```
as.numeric(yesnofac)
```

```
 [1] 1 1 1 1 2 1 1 1 2 2
```

# Cutting produces factor variables

```
library(Hmisc)
restData$zipGroups = cut2(restData$zipCode,g=4)
table(restData$zipGroups)
```

```
[-21226,21205) [ 21205,21220) [ 21220,21227) [ 21227,21287]
           338            375            300            314
```

# Using the mutate function

```
library(Hmisc); library(plyr)
restData2 = mutate(restData,zipGroups=cut2(zipCode,g=4))
table(restData2$zipGroups)
```

```
[-21226,21205) [ 21205,21220) [ 21220,21227) [ 21227,21287]
           338            375            300            314
```

# Common transforms

- `abs(x)` absolute value
- `sqrt(x)` square root
- `ceiling(x)` ceiling(3.475) is 4
- `floor(x)` floor(3.475) is 3
- `round(x,digits=n)` roun(3.475,digits=2) is 3.48
- `signif(x,digits=n)` signif(3.475,digits=2) is 3.5
- `cos(x), sin(x)` etc.
- `log(x)` natural logarithm
- `log2(x)`, `log10(x)` other common logs
- `exp(x)` exponentiating x

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%202.pdf
http://statmethods.net/management/functions.html

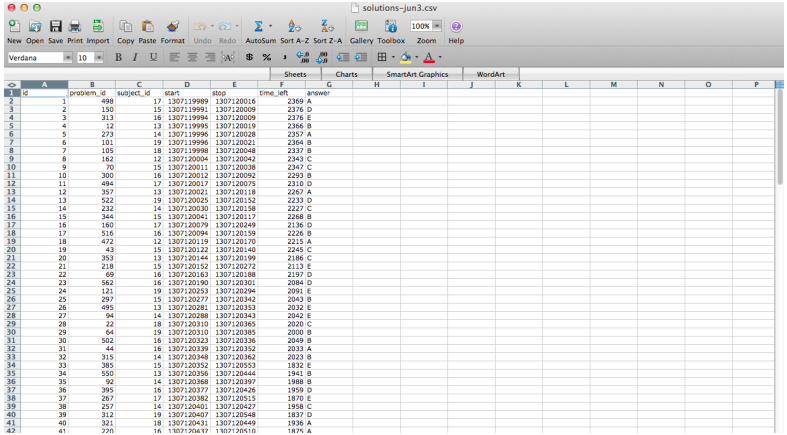
# Notes and further reading

- A tutorial from the developer of plyr - http://plyr.had.co.nz/09-user/
- Andrew Jaffe's R notes http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%202.pdf
-

# Reshaping data

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

# The goal is tidy data




| id | problem_id | subject_id | start | stop | time_left | answer |
|---|---|---|---|---|---|---|
| 1 | 498 | 17 | 1307119989 | 1307120016 | 2369 | A |
| 2 | 150 | 15 | 1307119991 | 1307120009 | 2376 | D |
| 3 | 313 | 16 | 1307119994 | 1307120009 | 2376 | E |
| 4 | 12 | 13 | 1307119995 | 1307120019 | 2366 | B |
| 5 | 273 | 14 | 1307119996 | 1307120028 | 2357 | A |
| 6 | 181 | 19 | 1307119996 | 1307120021 | 2364 | B |
| 7 | 105 | 18 | 1307119998 | 1307120048 | 2337 | B |
| 8 | 162 | 12 | 1307120004 | 1307120042 | 2343 | C |
| 9 | 70 | 15 | 1307120011 | 1307120038 | 2347 | C |
| 10 | 300 | 16 | 1307120012 | 1307120092 | 2293 | B |
| 11 | 494 | 17 | 1307120017 | 1307120075 | 2310 | D |
| 12 | 357 | 13 | 1307120021 | 1307120118 | 2267 | A |
| 13 | 522 | 19 | 1307120025 | 1307120152 | 2233 | D |
| 14 | 232 | 14 | 1307120030 | 1307120158 | 2227 | C |
| 15 | 344 | 15 | 1307120041 | 1307120117 | 2268 | B |
| 16 | 160 | 17 | 1307120079 | 1307120249 | 2136 | D |
| 17 | 516 | 16 | 1307120094 | 1307120159 | 2226 | B |
| 18 | 472 | 12 | 1307120119 | 1307120170 | 2215 | A |
| 19 | 43 | 15 | 1307120122 | 1307120140 | 2245 | C |
| 20 | 353 | 13 | 1307120144 | 1307120199 | 2186 | C |
| 21 | 218 | 15 | 1307120152 | 1307120272 | 2113 | E |
| 22 | 69 | 16 | 1307120163 | 1307120188 | 2197 | D |
| 23 | 562 | 16 | 1307120190 | 1307120301 | 2084 | D |
| 24 | 121 | 19 | 1307120253 | 1307120294 | 2091 | E |
| 25 | 297 | 15 | 1307120277 | 1307120342 | 2043 | B |
| 26 | 495 | 13 | 1307120281 | 1307120353 | 2032 | E |
| 27 | 94 | 14 | 1307120288 | 1307120343 | 2042 | E |
| 28 | 22 | 18 | 1307120310 | 1307120365 | 2020 | C |
| 29 | 64 | 19 | 1307120310 | 1307120385 | 2000 | B |
| 30 | 502 | 16 | 1307120323 | 1307120336 | 2049 | B |
| 31 | 44 | 16 | 1307120339 | 1307120352 | 2033 | A |
| 32 | 315 | 14 | 1307120348 | 1307120362 | 2023 | B |
| 33 | 385 | 15 | 1307120352 | 1307120553 | 1832 | E |
| 34 | 550 | 13 | 1307120356 | 1307120444 | 1941 | B |
| 35 | 92 | 14 | 1307120368 | 1307120397 | 1988 | B |
| 36 | 395 | 16 | 1307120377 | 1307120426 | 1959 | D |
| 37 | 267 | 17 | 1307120382 | 1307120515 | 1870 | E |
| 38 | 257 | 14 | 1307120401 | 1307120427 | 1958 | C |
| 39 | 312 | 19 | 1307120407 | 1307120548 | 1837 | D |
| 40 | 321 | 18 | 1307120431 | 1307120449 | 1936 | A |

1. Each variable forms a column
2. Each observation forms a row
3. Each table/file stores data about one kind of observation (e.g. people/hospitals).

http://vita.had.co.nz/papers/tidy-data.pdf

Leek, Taub, and Pineda 2011 PLoS One

# Start with reshaping

```
library(reshape2)
head(mtcars)
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

# Melting data frames

```
mtcars$carname <- rownames(mtcars)
carMelt <- melt(mtcars,id=c("carname","gear","cyl"),measure.vars=c("mpg","hp"))
head(carMelt,n=3)
```

```
        carname gear cyl variable value
1     Mazda RX4    4   6      mpg  21.0
2 Mazda RX4 Wag    4   6      mpg  21.0
3    Datsun 710    4   4      mpg  22.8
```

```
tail(carMelt,n=3)
```

```
         carname gear cyl variable value
62  Ferrari Dino    5   6       hp   175
63 Maserati Bora    5   8       hp   335
64    Volvo 142E    4   4       hp   109
```



http://www.statmethods.net/management/reshape.html

# Casting data frames

```
cylData <- dcast(carMelt, cyl ~ variable)
cylData
```

```
  cyl mpg hp
1   4  11 11
2   6   7  7
3   8  14 14
```

```
cylData <- dcast(carMelt, cyl ~ variable,mean)
cylData
```

```
  cyl   mpg     hp
1   4 26.66  82.64
2   6 19.74 122.29
3   8 15.10 209.21
```



http://www.statmethods.net/management/reshape.html

# Averaging values

```
head(InsectSprays)
```

```
  count spray
1    10     A
2     7     A
3    20     A
4    14     A
5    14     A
6    12     A
```

```
tapply(InsectSprays$count,InsectSprays$spray,sum)
```

```
  A   B   C   D   E   F
174 184  25  59  42 200
```

http://www.r-bloggers.com/a-quick-primer-on-split-apply-combine-problems/

# Another way - split

```
spIns =  split(InsectSprays$count,InsectSprays$spray)
spIns
```

```
$A
 [1] 10  7 20 14 14 12 10 23 17 20 14 13

$B
 [1] 11 17 21 11 16 14 17 17 19 21  7 13

$C
 [1] 0 1 7 2 3 1 2 1 3 0 1 4

$D
 [1]  3  5 12  6  4  3  5  5  5  5  2  4

$E
 [1] 3 5 3 5 3 6 1 1 3 2 6 4

$F
 [1] 11  9 15 22 15 16 13 10 26 26 24 13
```

# Another way - apply

```
sprCount = lapply(spIns,sum)
sprCount
```

```
$A
[1] 174

$B
[1] 184

$C
[1] 25

$D
[1] 59

$E
[1] 42

$F
[1] 200
```

# Another way - combine

```
unlist(sprCount)
```

```
  A   B   C   D   E   F
174 184  25  59  42 200
```

```
sapply(spIns,sum)
```

```
  A   B   C   D   E   F
174 184  25  59  42 200
```

# Another way - plyr package

```
ddply(InsectSprays,.(spray),summarize,sum=sum(count))
```

```
  spray sum
1     A 174
2     B 184
3     C  25
4     D  59
5     E  42
6     F 200
```

# Creating a new variable

```
spraySums <- ddply(InsectSprays,.(spray),summarize,sum=ave(count,FUN=sum))
dim(spraySums)
```

```
[1] 72  2
```

```
head(spraySums)
```

```
  spray sum
1     A 174
2     A 174
3     A 174
4     A 174
5     A 174
6     A 174
```

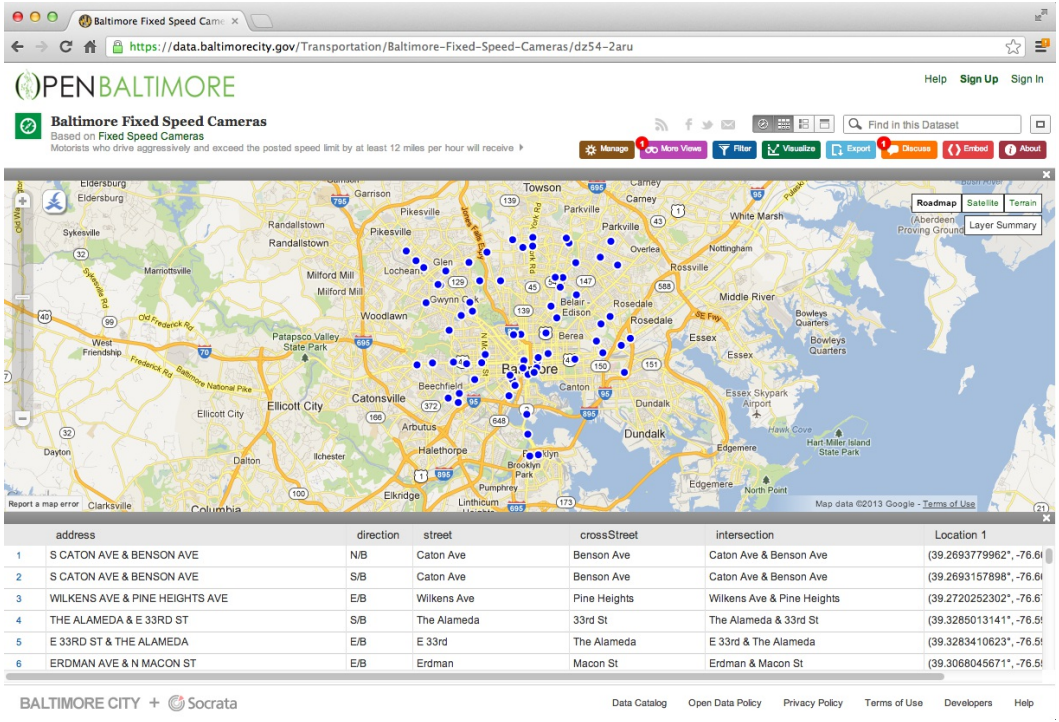
# More information

- A tutorial from the developer of plyr - http://plyr.had.co.nz/09-user/
- A nice reshape tutorial http://www.slideshare.net/jeffreybreen/reshaping-data-in-r
- A good plyr primer - http://www.r-bloggers.com/a-quick-primer-on-split-apply-combine-problems/
- See also the functions
  - acast - for casting as multi-dimensional arrays
  - arrange - for faster reordering without using order() commands
  - mutate - adding new variables

# Editing text variables

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

# Example - Baltimore camera data




https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru

# Fixing character vectors - tolower(), toupper()

```
if(!file.exists("./data")){dir.create("./data")}
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl,destfile="./data/cameras.csv",method="curl")
cameraData <- read.csv("./data/cameras.csv")
names(cameraData)
```

```
[1] "address"      "direction"    "street"       "crossStreet"  "intersection" "Location.1"
```

```
tolower(names(cameraData))
```

```
[1] "address"      "direction"    "street"       "crossstreet"  "intersection" "location.1"
```

# Fixing character vectors - strsplit()

- Good for automatically splitting variable names
- Important parameters: *x*, *split*

```
splitNames = strsplit(names(cameraData),"\\.")
splitNames[[5]]
```

```
[1] "intersection"
```

```
splitNames[[6]]
```

```
[1] "Location" "1"
```

# Quick aside - lists

```
mylist <- list(letters = c("A", "b", "c"), numbers = 1:3, matrix(1:25, ncol = 5))
head(mylist)
```

```
$letters
[1] "A" "b" "c"

$numbers
[1] 1 2 3

[[3]]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
```

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%203.pdf

# Quick aside - lists

```
mylist[1]
```

```
$letters
[1] "A" "b" "c"
```

```
mylist$letters
```

```
[1] "A" "b" "c"
```

```
mylist[[1]]
```

```
[1] "A" "b" "c"
```

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%203.pdf

# Fixing character vectors - sapply()

- Applies a function to each element in a vector or list
- Important parameters: *X*,*FUN*

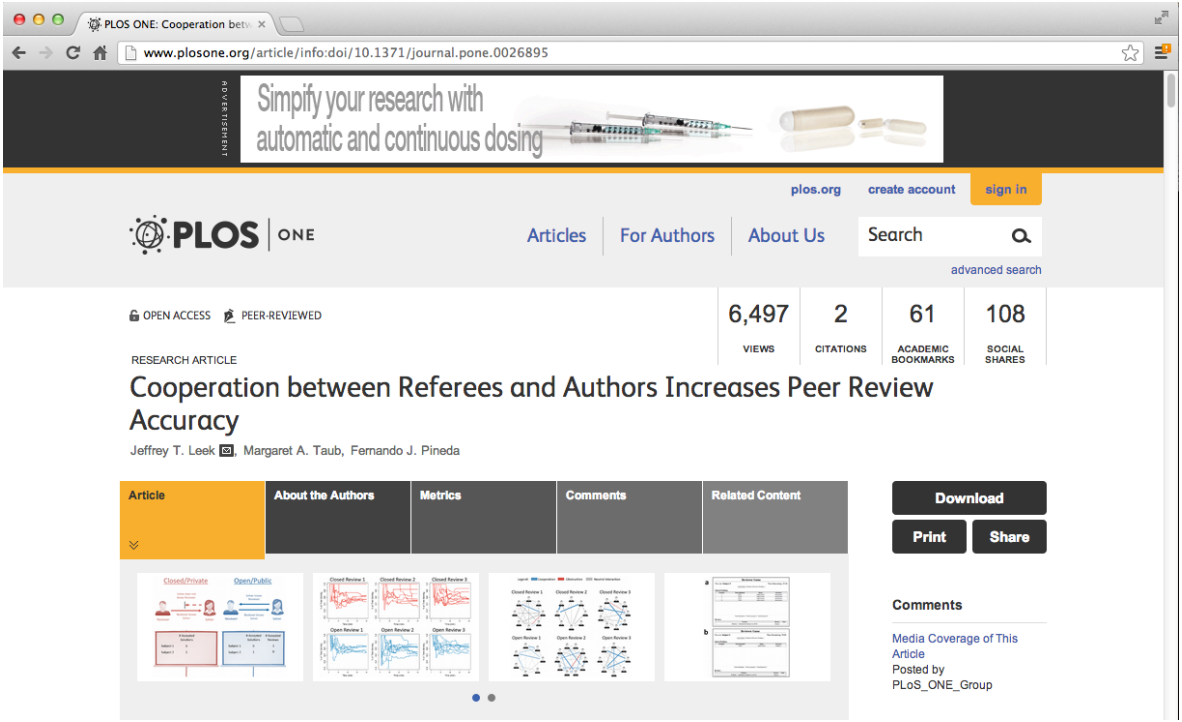
```
splitNames[[6]][1]
```

```
[1] "Location"
```

```
firstElement <- function(x){x[1]}
sapply(splitNames,firstElement)
```

```
[1] "address"      "direction"    "street"       "crossStreet"  "intersection" "Location"
```

# Peer review experiment data




http://www.plosone.org/article/info:doi/10.1371/journal.pone.0026895

# Peer review data

```
fileUrl1 <- "https://dl.dropboxusercontent.com/u/7710864/data/reviews-apr29.csv"
fileUrl2 <- "https://dl.dropboxusercontent.com/u/7710864/data/solutions-apr29.csv"
download.file(fileUrl1,destfile="./data/reviews.csv",method="curl")
download.file(fileUrl2,destfile="./data/solutions.csv",method="curl")
reviews <- read.csv("./data/reviews.csv"); solutions <- read.csv("./data/solutions.csv")
head(reviews,2)
```

```
  id solution_id reviewer_id      start       stop time_left accept
1  1           3          27 1304095698 1304095758      1754      1
2  2           4          22 1304095188 1304095206      2306      1
```

```
head(solutions,2)
```

```
  id problem_id subject_id      start       stop time_left answer
1  1        156         29 1304095119 1304095169      2343      B
2  2        269         25 1304095119 1304095183      2329      C
```

# Fixing character vectors - sub()

- Important parameters: *pattern*, *replacement*, *x*

```
names(reviews)
```

```
[1] "id"          "solution_id" "reviewer_id" "start"       "stop"        "time_left"
[7] "accept"
```

```
sub("_","",names(reviews),)
```

```
[1] "id"         "solutionid" "reviewerid" "start"      "stop"       "timeleft"   "accept"
```

# Fixing character vectors - gsub()

```
testName <- "this_is_a_test"
sub("_","",testName)
```

```
[1] "thisis_a_test"
```

```
gsub("_","",testName)
```

```
[1] "thisisatest"
```

# Finding values - grep(),grepl()

```
grep("Alameda",cameraData$intersection)
```

```
[1]  4  5 36
```

```
table(grepl("Alameda",cameraData$intersection))
```

```
FALSE  TRUE 
   77     3 
```

```
cameraData2 <- cameraData[!grepl("Alameda",cameraData$intersection),]
```

# More on grep()

```
grep("Alameda",cameraData$intersection,value=TRUE)
```

```
[1] "The Alameda  & 33rd St"   "E 33rd  & The Alameda"    "Harford \n & The Alameda"
```

```
grep("JeffStreet",cameraData$intersection)
```

```
integer(0)
```

```
length(grep("JeffStreet",cameraData$intersection))
```

```
[1] 0
```

http://www.biostat.jhsph.edu/~ajaffe/lec_winterR/Lecture%203.pdf

# More useful string functions

```
library(stringr)
nchar("Jeffrey Leek")
```

```
[1] 12
```

```
substr("Jeffrey Leek",1,7)
```

```
[1] "Jeffrey"
```

```
paste("Jeffrey","Leek")
```

```
[1] "Jeffrey Leek"
```

# More useful string functions

```
paste0("Jeffrey","Leek")
```

```
[1] "JeffreyLeek"
```

```
str_trim("Jeff      ")
```

```
[1] "Jeff"
```

# Important points about text in data sets

- Names of variables should be
  - All lower case when possible
  - Descriptive (Diagnosis versus Dx)
  - Not duplicated
  - Not have underscores or dots or white spaces
- Variables with character values
  - Should usually be made into factor variables (depends on application)
  - Should be descriptive (use TRUE/FALSE instead of 0/1 and Male/Female versus 0/1 or M/F)

# Regular Expressions

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

# Regular expressions

- Regular expressions can be thought of as a combination of literals and *metacharacters*
- To draw an analogy with natural language, think of literal text forming the words of this language, and the metacharacters defining its grammar
- Regular expressions have a rich set of metacharacters

# Literals

Simplest pattern consists only of literals. The literal “nuclear” would match to the following lines:

```
Ooh. I just learned that to keep myself alive after a
nuclear blast! All I have to do is milk some rats
then drink the milk. Aweosme. :}

Laozi says nuclear weapons are mas macho

Chaos in a country that has nuclear weapons -- not good.

my nephew is trying to teach me nuclear physics, or
possibly just trying to show me how smart he is
so I'll be proud of him [which I am].

lol if you ever say "nuclear" people immediately think
DEATH by radiation LOL
```

# Literals

The literal “Obama” would match to the following lines

```
Politics r dum. Not 2 long ago Clinton was sayin Obama
was crap n now she sez vote 4 him n unite? WTF?
Screw em both + Mcain. Go Ron Paul!

Clinton conceeds to Obama but will her followers listen??

Are we sure Chelsea didn’t vote for Obama?

thinking ... Michelle Obama is terrific!

jetlag..no sleep...early mornig to starbux..Ms. Obama
was moving
```

# Regular Expressions

- Simplest pattern consists only of literals; a match occurs if the sequence of literals occurs anywhere in the text being tested
- What if we only want the word “Obama”? or sentences that end in the word “Clinton”, or “clinton” or “clinto”?

# Regular Expressions

We need a way to express

- whitespace word boundaries
- sets of literals
- the beginning and end of a line
- alternatives (“war” or “peace”) Metacharacters to the rescue!

# Metacharacters

Some metacharacters represent the start of a line

```
^i think
```

will match the lines

```
i think we all rule for participating
i think i have been outed
i think this will be quite fun actually
i think i need to go to work
i think i first saw zombo in 1999.
```

# Metacharacters

$ represents the end of a line

```
morning$
```

will match the lines

```
well they had something this morning
then had to catch a tram home in the morning
dog obedience school in the morning
and yes happy birthday i forgot to say it earlier this morning
I walked in the rain this morning
good morning
```

# Character Classes with []

We can list a set of characters we will accept at a given point in the match

```
[Bb][Uu][Ss][Hh]
```

will match the lines

```
The democrats are playing, "Name the worst thing about Bush!"
I smelled the desert creosote bush, brownies, BBQ chicken
BBQ and bushwalking at Molonglo Gorge
Bush TOLD you that North Korea is part of the Axis of Evil
I'm listening to Bush - Hurricane (Album Version)
```

# Character Classes with []

```
^[Ii] am
```

will match

```
i am so angry at my boyfriend i can't even bear to
look at him

i am boycotting the apple store

I am twittering from iPhone

I am a very vengeful person when you ruin my sweetheart.

I am so over this. I need food. Mmmm bacon...
```

# Character Classes with []

Similarly, you can specify a range of letters [a-z] or [a-zA-Z]; notice that the order doesn't matter

```
^[0-9][a-zA-Z]
```

will match the lines

```
7th inning stretch
2nd half soon to begin. OSU did just win something
3am - cant sleep - too hot still.. :(
5ft 7 sent from heaven
1st sign of starvagtion
```

# Character Classes with []

When used at the beginning of a character class, the “” is also a metacharacter and indicates matching characters NOT in the indicated class

```
[^?.]$
```

will match the lines

```
i like basketballs
6 and 9
dont worry... we all die anyway!
Not in Baghdad
helicopter under water? hmmm
```

# Regular Expressions II

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

# More Metacharacters

“.” is used to refer to any character. So

```
9.11
```

will match the lines

```
its stupid the post 9-11 rules
if any 1 of us did 9/11 we would have been caught in days.
NetBios: scanning ip 203.169.114.66
Front Door 9:11:46 AM
Sings: 0118999881999119725...3 !
```

# More Metacharacters: |

This does not mean “pipe” in the context of regular expressions; instead it translates to “or”; we can use it to combine two expressions, the subexpressions being called alternatives

```
flood|fire
```

will match the lines

```
is firewire like usb on none macs?
the global flood makes sense within the context of the bible
yeah ive had the fire on tonight
... and the floods, hurricanes, killer heatwaves, rednecks, gun nuts, etc.
```

# More Metacharacters: |

We can include any number of alternatives...

```
flood|earthquake|hurricane|coldfire
```

will match the lines

```
Not a whole lot of hurricanes in the Arctic.
We do have earthquakes nearly every day somewhere in our State
hurricanes swirl in the other direction
coldfire is STRAIGHT!
'cause we keep getting earthquakes
```

# More Metacharacters: |

The alternatives can be real expressions and not just literals

```
^[Gg]ood|[Bb]ad
```

will match the lines

```
good to hear some good knews from someone here
Good afternoon fellow american infidels!
good on you-what do you drive?
Katie... guess they had bad experiences...
my middle name is trouble, Miss Bad News
```

# More Metacharacters: ( and )

Subexpressions are often contained in parentheses to constrain the alternatives

```
^([Gg]ood|[Bb]ad)
```

will match the lines

```
bad habbit
bad coordination today
good, becuase there is nothing worse than a man in kinky underwear
Badcop, its because people want to use drugs
Good Monday Holiday
Good riddance to Limey
```

# More Metacharacters: ?

The question mark indicates that the indicated expression is optional

```
[Gg]eorge( [Ww]\.)? [Bb]ush
```

will match the lines

```
i bet i can spell better than you and george bush combined
BBC reported that President George W. Bush claimed God told him to invade I
a bird in the hand is worth two george bushes
```

# One thing to note...

In the following

```
[Gg]eorge( [Ww]\.)? [Bb]ush
```

we wanted to match a “.” as a literal period; to do that, we had to “escape” the metacharacter, preceding it with a backslash In general, we have to do this for any metacharacter we want to include in our match

# More metacharacters: * and +

The * and + signs are metacharacters used to indicate repetition; * means “any number, including none, of the item” and + means “at least one of the item”

```
(.*)
```

will match the lines

```
anyone wanna chat? (24, m, germany)
hello, 20.m here... ( east area + drives + webcam )
(he means older men)
()
```

# More metacharacters: * and +

The * and + signs are metacharacters used to indicate repetition; * means “any number, including none, of the item” and + means “at least one of the item”

```
[0-9]+ (.*)[0-9]+
```

will match the lines

```
working as MP here 720 MP battallion, 42nd birgade
so say 2 or 3 years at colleage and 4 at uni makes us 23 when and if we fin
it went down on several occasions for like, 3 or 4 *days*
Mmmm its time 4 me 2 go 2 bed
```

# More metacharacters: { and }

{ and } are referred to as interval quantifiers; the let us specify the minimum and maximum number of matches of an expression

```
[Bb]ush( +[^ ]+ +){1,5} debate
```

will match the lines

```
Bush has historically won all major debates he's done.
in my view, Bush doesn't need these debates..
bush doesn't need the debates? maybe you are right
That's what Bush supporters are doing about the debate.
Felix, I don't disagree that Bush was poorly prepared for the debate.
indeed, but still, Bush should have taken the debate more seriously.
Keep repeating that Bush smirked and scowled during the debate
```

# More metacharacters: and

- m,n means at least m but not more than n matches
- m means exactly m matches
- m, means at least m matches

# More metacharacters: ( and ) revisited

- In most implementations of regular expressions, the parentheses not only limit the scope of alternatives divided by a “|”, but also can be used to “remember” text matched by the subexpression enclosed
- We refer to the matched text with \1, \2, etc.

# More metacharacters: ( and ) revisited

So the expression

```
+([a-zA-Z]+) +\1 +
```

will match the lines

```
time for bed, night night twitter!
blah blah blah blah
my tattoo is so so itchy today
i was standing all all alone against the world outside...
hi anybody anybody at home
estudiando css css css css.... que desastritooooo
```

# More metacharacters: ( and ) revisited

The * is “greedy” so it always matches the *longest* possible string that satisfies the regular expression. So

```
^s(.*)s
```

matches

```
sitting at starbucks
setting up mysql and rails
studying stuff for the exams
spaghetti with marshmallows
stop fighting with crackers
sore shoulders, stupid ergonomics
```

# More metacharacters: ( and ) revisited

The greediness of * can be turned off with the ?, as in

```
^s(.*?)s$
```

# Summary

- Regular expressions are used in many different languages; not unique to R.
- Regular expressions are composed of literals and metacharacters that represent sets or classes of characters/words
- Text processing via regular expressions is a very powerful way to extract data from “unfriendly” sources (not all data comes as a CSV file)
- Used with the functions `grep`,`grepl`,`sub`,`gsub` and others that involve searching for text strings (Thanks to Mark Hansen for some material in this lecture.)