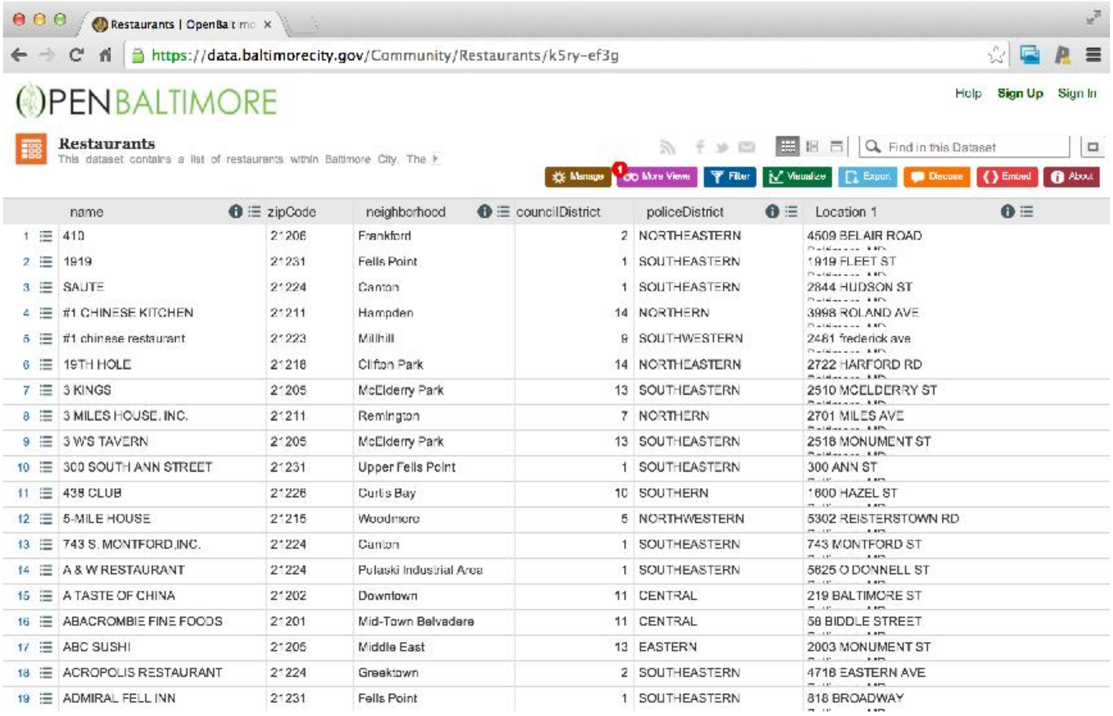
Why create new variables?

- 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 WS 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 Belvadere	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
  (-2.123e+04,2.12e+04]     0     0     0     0     0     0     0     0     0     0     0     0
  (2.12e+04,2.122e+04]     17    54    10    32    69     0     0     0     0     0     0     0
  (2.122e+04,2.123e+04]     0     0     0     0     0     1     7    56   199    19     0     0
  (2.123e+04,2.129e+04]     0     0     0     0     0     0     0     0     0     0    18     4

                        21229 21230 21231 21234 21237 21239 21251 21287
  (-2.123e+04,2.12e+04]     0     0     0     0     0     0     0     0
  (2.12e+04,2.122e+04]      0     0     0     0     0     0     0     0
  (2.122e+04,2.123e+04]     0     0     0     0     0     0     0     0
  (2.123e+04,2.129e+04]    13   156   127     7     1     3     2     1
```

/

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
-

/