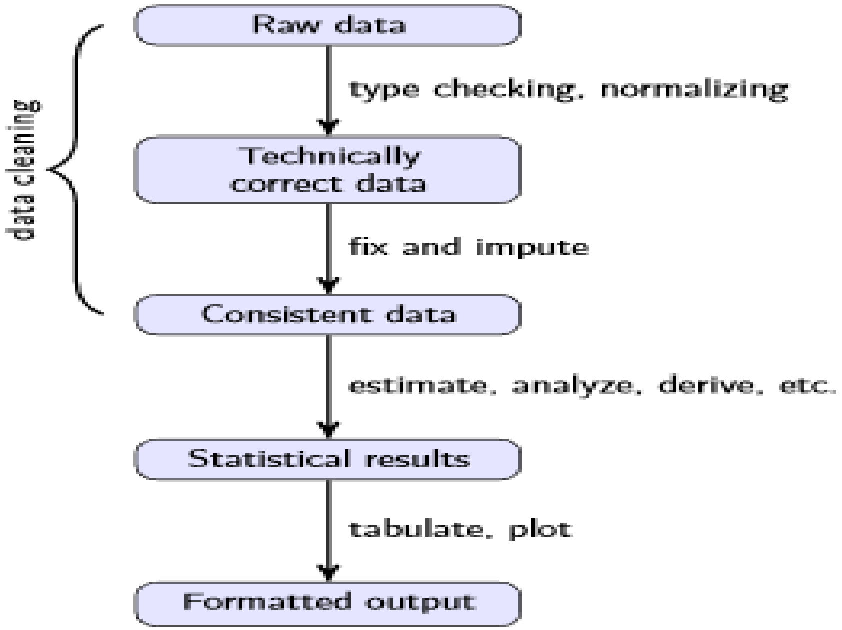
Getting & Cleaning Data

Statistical Analysis

- Summarizing, a statistical analysis can be separated in five stages, from raw data to formatted output
- The quality of the data improves in every step towards the final result.
- Data cleaning encompasses two of the five stages in a statistical analysis, which again emphasizes its importance in statistical practice



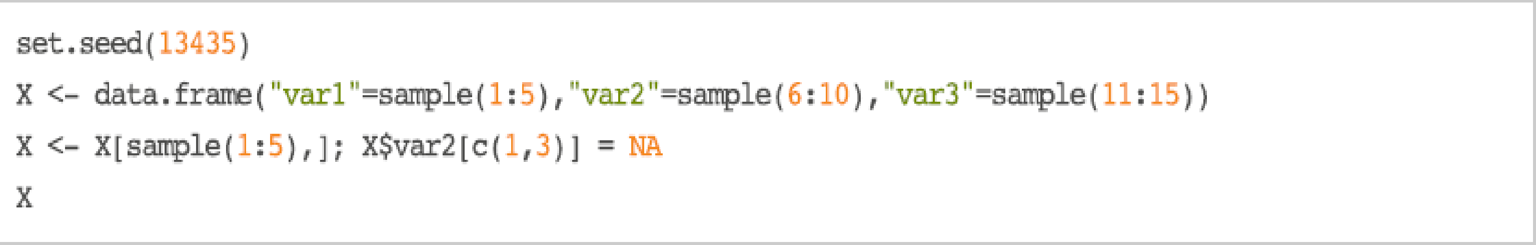
Source : cran.r-project.org

Subsetting

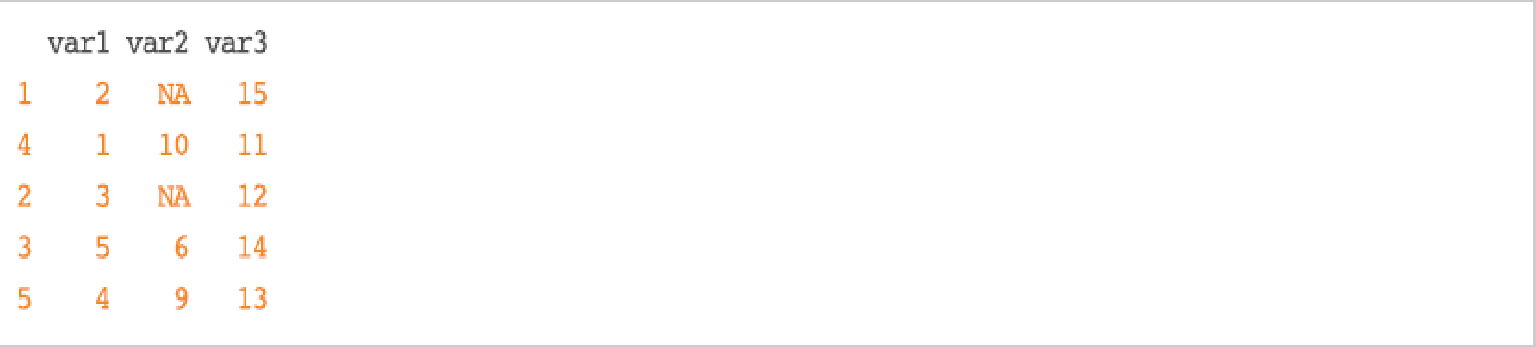
- ❑ Subsetting data.
- ❑ Use a logical operator to do this.
  - ==, >, <, <=, >=, <> are all logical operators.
  - Note that the "equals" logical operator is two = signs.
- ❑ Example:
  - `D[D$Gender == "M",]`
  - This will return the rows of D where Gender is "M".
  - Remember R is case sensitive!
  - This code does nothing to the original dataset.
  - `D.M <- D[D$Gender == "M",]` gives a dataset with the appropriate rows.

Source : *http://www.statmethods.net*

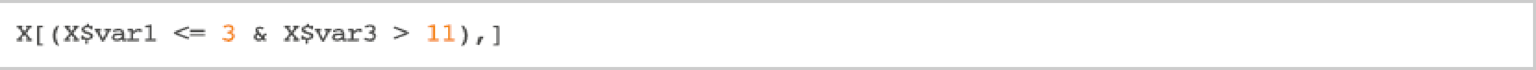
Subsetting Example



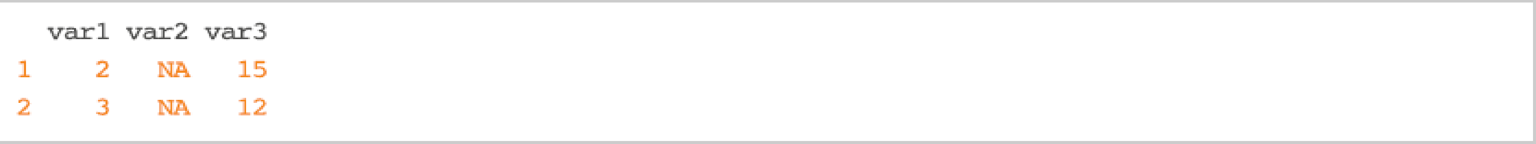
```
set.seed(13435)
X <- data.frame("var1"=sample(1:5),"var2"=sample(6:10),"var3"=sample(11:15))
X <- X[sample(1:5),]; X$var2[c(1,3)] = NA
X
```



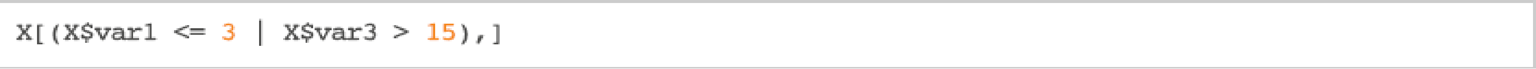
```
  var1 var2 var3
1    2   NA   15
4    1   10   11
2    3   NA   12
3    5    6   14
5    4    9   13
```



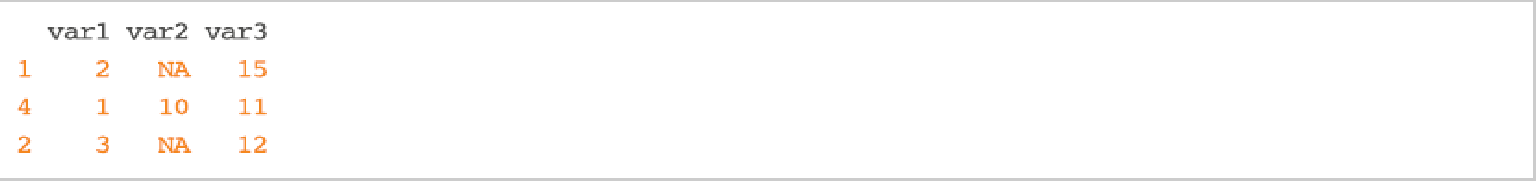
```
X[(X$var1 <= 3 & X$var3 > 11),]
```



```
  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
```

Sorting & Ordering

- ❑ Using sort function
- ❑ Descending order is achieved by option decreasing = TRUE
- ❑ Ordering is done using order function
- ❑ To sort a data frame on one or more columns, you can use the arrange function from plyr package

```
# Make up a randomly ordered vector v <- sample(101:110) #
102 107 104 106 105 103 101 108 109 110

# Sort the vector sort(v) # 101 102 103 104 105 106 107 108
109 110

# Reverse sort sort(v, decreasing=TRUE) # 110 109 108 107
106 105 104 103 102 101
```

Source :
http://www.statmethods.net

Excercises

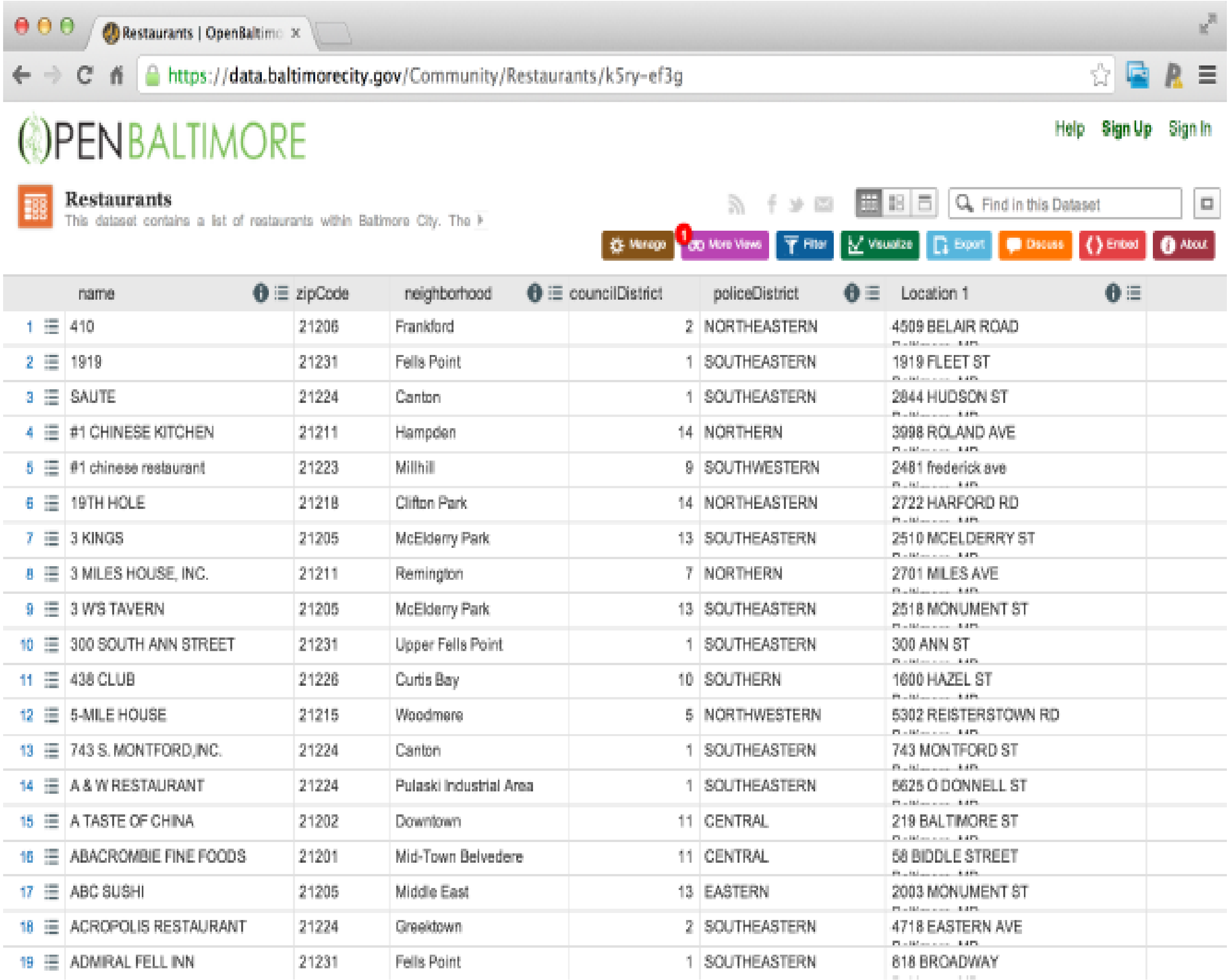
- ❑ sort(X$var1)
- ❑ sort(X$var1,descreasing=TRUE)
- ❑ X[order(X$var1),]

Summarizing Data

- The summary() function works best if you just use R interactively at the command line for scanning your dataset quickly. You shouldn’t try to use it within a custom function you wrote yourself.
- The output of the summary() function shows you for every variable a set of descriptive statistics, depending on the type of the variable:
- **Numerical variables:** summary() gives you the range, quartiles, median, and mean.
- **Factor variables:** summary() gives you a table with frequencies.
- **Numerical and factor variables:** summary() gives you the number of missing values, if there are any.
- **Character variables:** summary() doesn’t give you any information at all apart from the length and the class (which is 'character').

Source: *www.dummies.com*

Sample Data



Restaurants | OpenBaltimo

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

OPENBALTIMORE

Help Sign Up Sign In

Restaurants

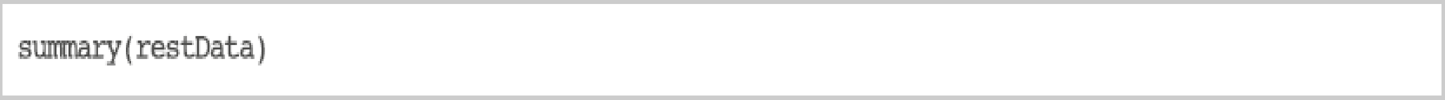
This dataset contains a list of restaurants within Baltimore City. The

Find in this Dataset

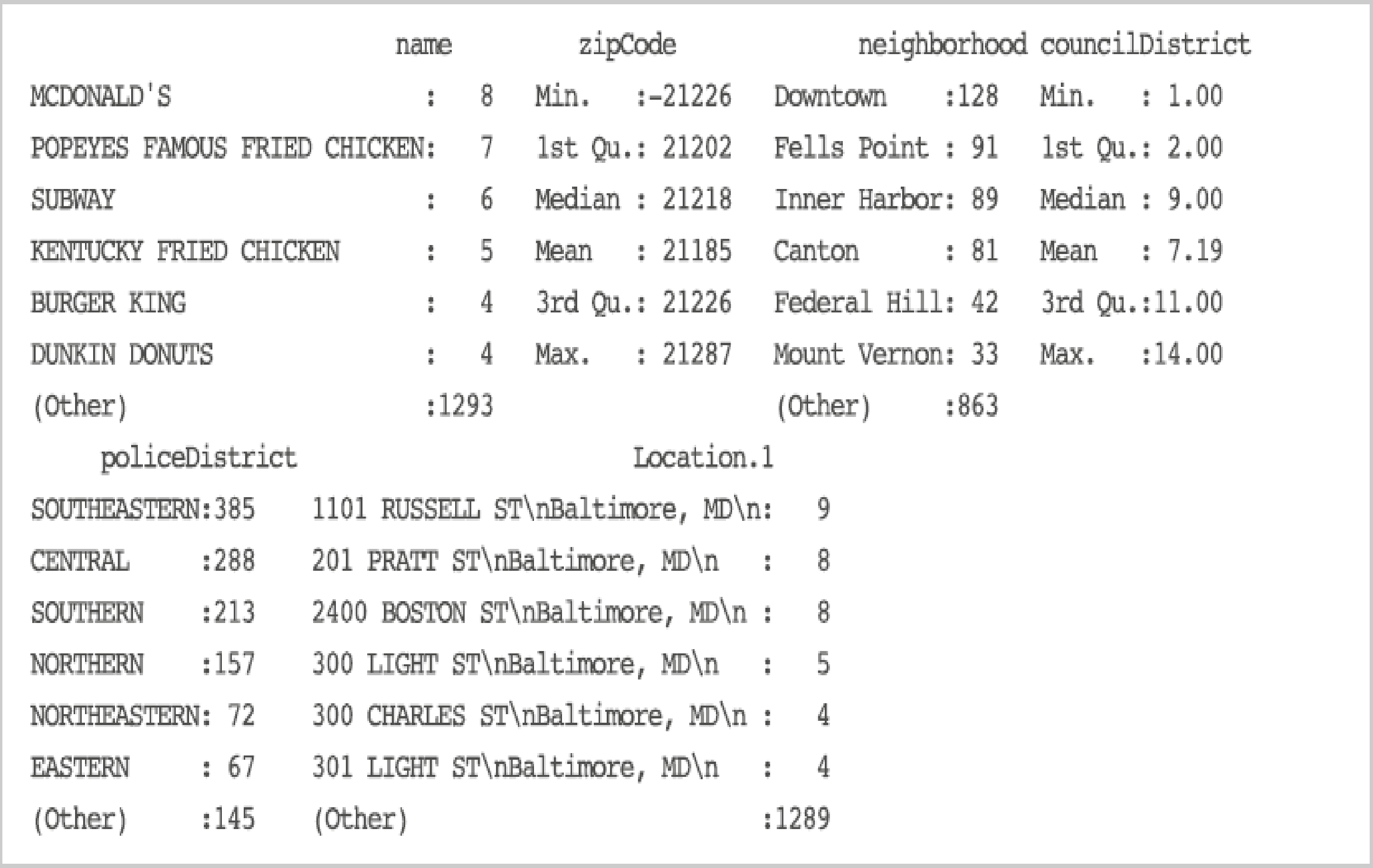
Manage | More Views | Filter | Visualize | Export | Discuss | Embed | About

	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

Summarize Example



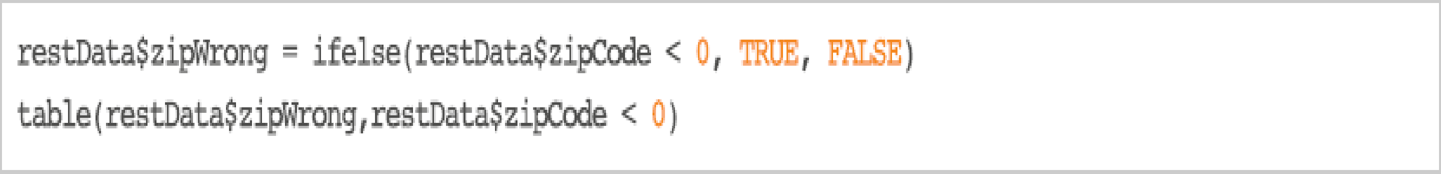
```
summary(restData)
```



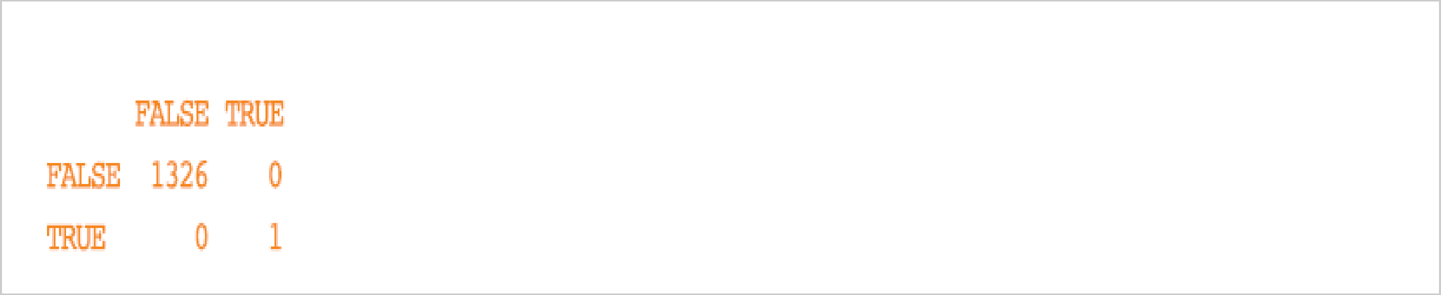
```
                          name         zipCode             neighborhood councilDistrict
MCDONALD'S                  :   8   Min.   :-21226   Downtown    :128   Min.   : 1.00  
POPEYES FAMOUS FRIED CHICKEN:   7   1st Qu.: 21202   Fells Point : 91   1st Qu.: 2.00  
SUBWAY                      :   6   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  
```

Creating New Variables

- ❑ Transformation of raw data to get the required values
- ❑ Creating Sequences s1 <- seq(1,10,by=2;s1 will return [1] 1 3 5 7 9
- ❑ Creating Binary variables, Categorical Variables, Factor Variables and Easier Cutting are some methods of creating new variables



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



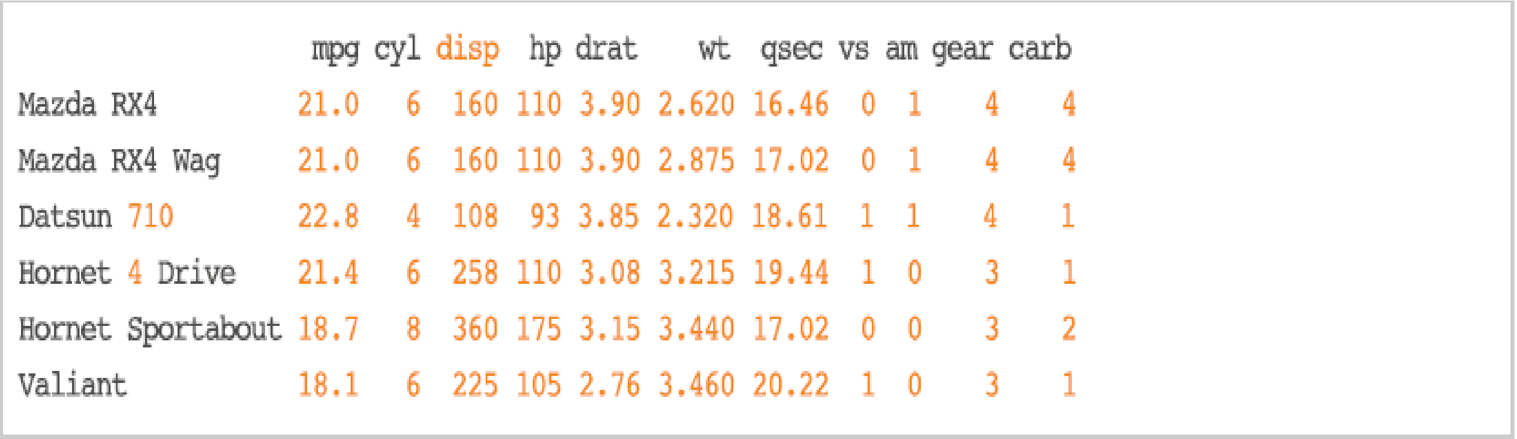
```
      FALSE TRUE
FALSE  1326    0
TRUE      0    1
```

Reshaping Data

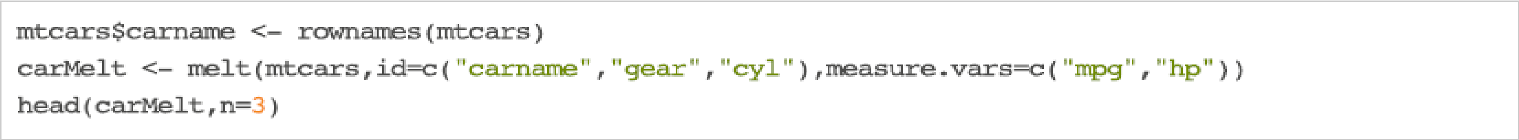
- Reduce big table to small table
- (must lose information)
- Each cell in the new table corresponds to
- multiple cells in the old table
- Different approaches like Melting data frames, Casting data frames, Averaging values, Split, Apply and using plyr package

Source : *http://www.statmethods.net*

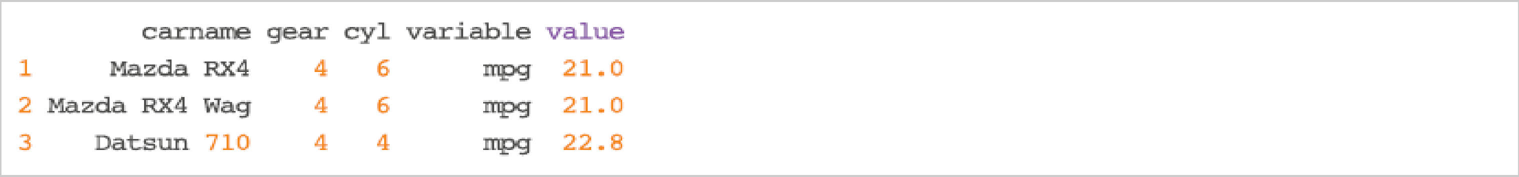
Reshaping Sample



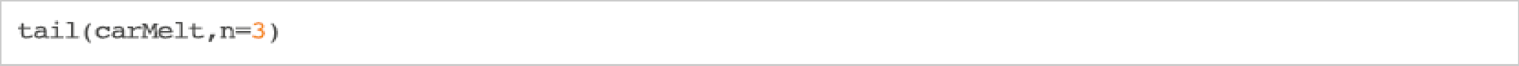
```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   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
```



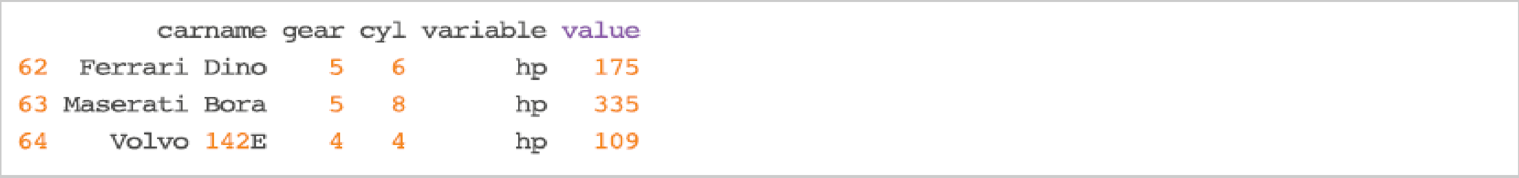
```
mtcars$carname <- rownames(mtcars)
carMelt <- melt(mtcars,id=c("carname","gear","cyl"),measure.vars=c("mpg","hp"))
head(carMelt,n=3)
```



```
        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)
```

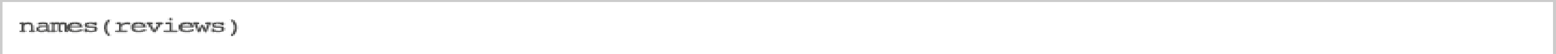


```
         carname gear cyl variable value
62  Ferrari Dino    5   6       hp   175
63 Maserati Bora    5   8       hp   335
64    Volvo 142E    4   4       hp   109
```

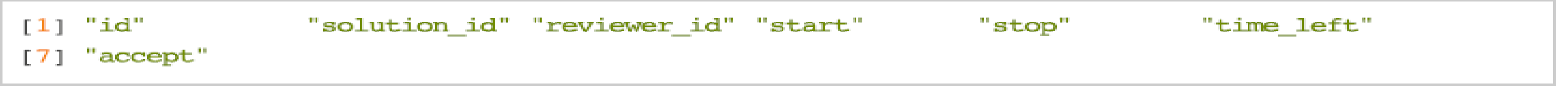
Merging Data

- In R you use the merge() function to combine data frames. This powerful function tries to identify columns or rows that are common between the two different data frames.
- **x:** A data frame.
- **y:** A data frame.
- **by, by.x, by.y:** The names of the columns that are common to both x and y. The default is to use the columns with common names between the two data frames.
- **all, all.x, all.y:** Logical values that specify the type of merge. The default value is all=FALSE (meaning that only the matching rows are returned).

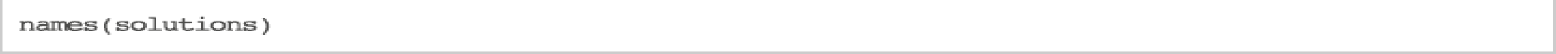
Merging Data Example



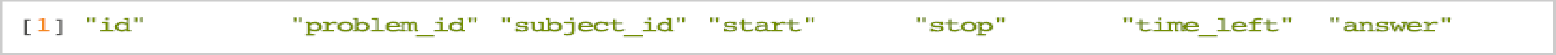
```
names(reviews)
```



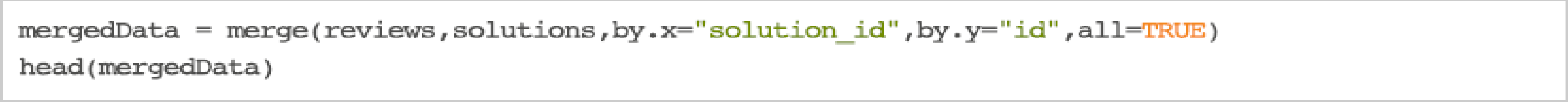
```
[1] "id"           "solution_id"  "reviewer_id"  "start"        "stop"         "time_left"
[7] "accept"
```



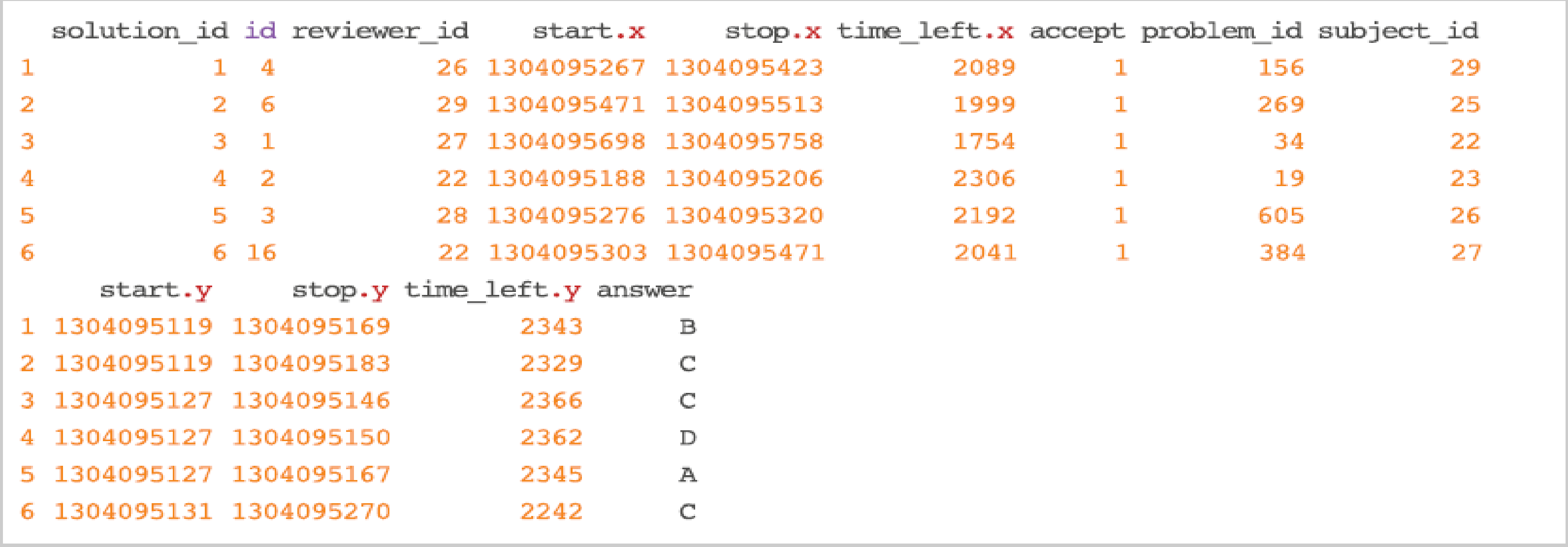
```
names(solutions)
```



```
[1] "id"          "problem_id"  "subject_id"  "start"       "stop"        "time_left"   "answer"
```



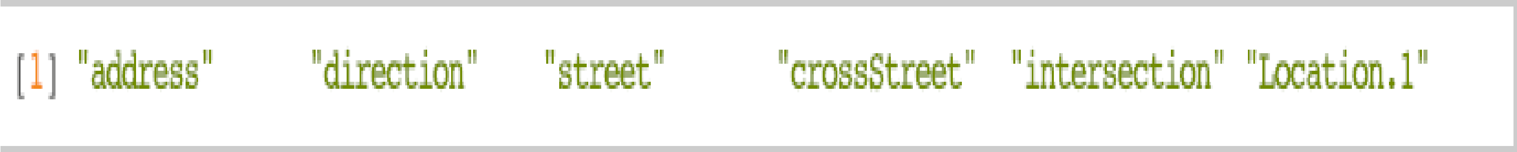
```
mergedData = merge(reviews,solutions,by.x="solution_id",by.y="id",all=TRUE)
head(mergedData)
```



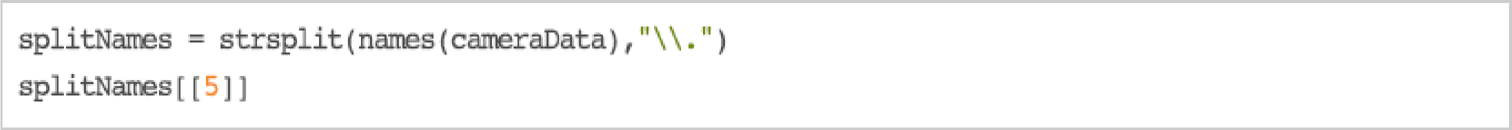
```
  solution_id id reviewer_id    start.x     stop.x time_left.x accept problem_id subject_id
1           1  4          26 1304095267 1304095423        2089      1        156         29
2           2  6          29 1304095471 1304095513        1999      1        269         25
3           3  1          27 1304095698 1304095758        1754      1         34         22
4           4  2          22 1304095188 1304095206        2306      1         19         23
5           5  3          28 1304095276 1304095320        2192      1        605         26
6           6 16          22 1304095303 1304095471        2041      1        384         27
     start.y     stop.y time_left.y answer
1 1304095119 1304095169        2343      B
2 1304095119 1304095183        2329      C
3 1304095127 1304095146        2366      C
4 1304095127 1304095150        2362      D
5 1304095127 1304095167        2345      A
6 1304095131 1304095270        2242      C
```

Editing Text Variables

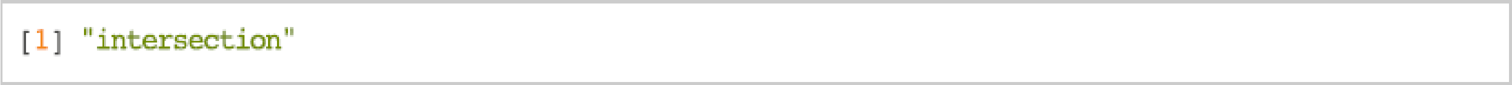
- ❑ Data can be modified as required for analysis
- ❑ Different functions can used for editing like tolower(),toupper(), strsplit(),sapply() and String functions
- ❑ Search and Finding can be done using grep() and grepl() functions



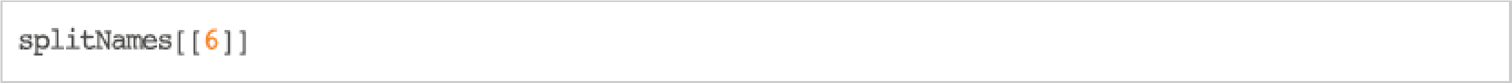
```
[1] "address"      "direction"    "street"       "crossStreet"  "intersection" "Location.1"
```



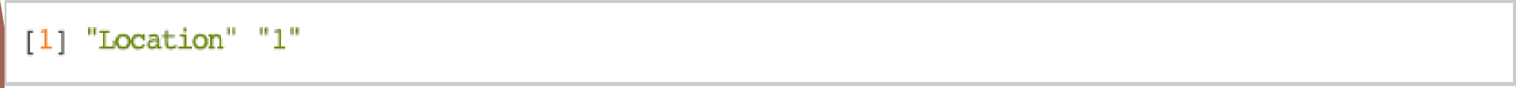
```
splitNames = strsplit(names(cameraData),"\\.")
splitNames[[5]]
```



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

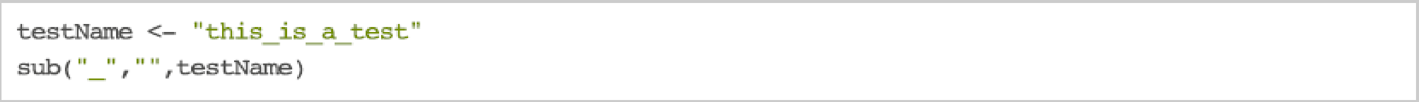


```
splitNames[[6]]
```

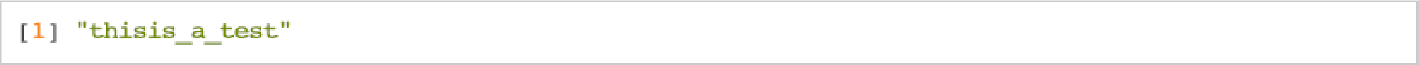


```
[1] "Location" "1"
```

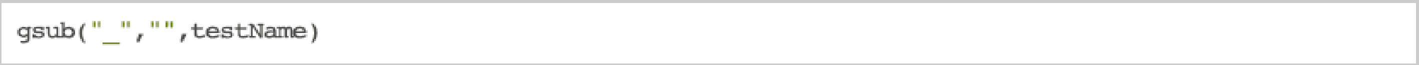
Editing Text Variables Examples



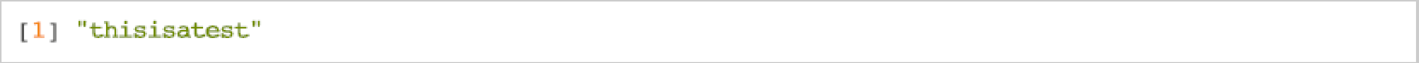
```
testName <- "this_is_a_test"
sub("_","",testName)
```



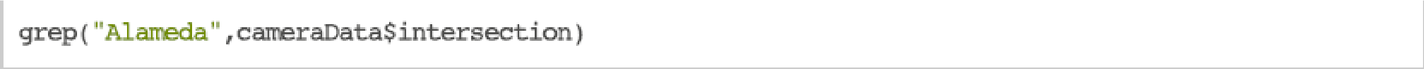
```
[1] "thisis_a_test"
```



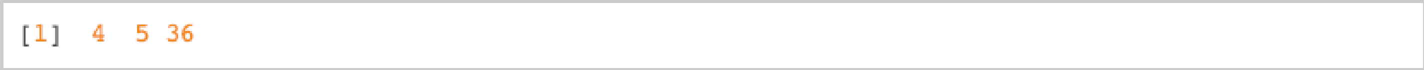
```
gsub("_","",testName)
```



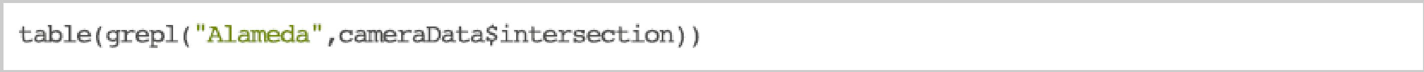
```
[1] "thisisatest"
```



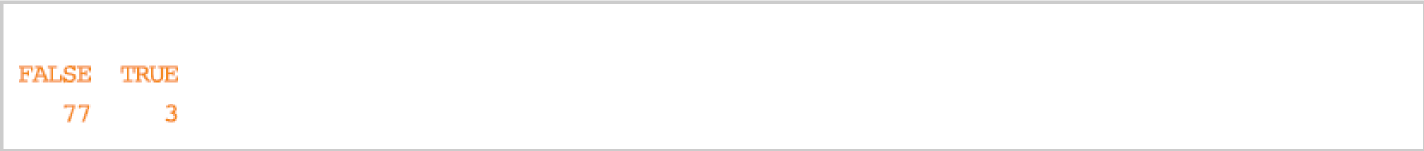
```
grep("Alameda",cameraData$intersection)
```



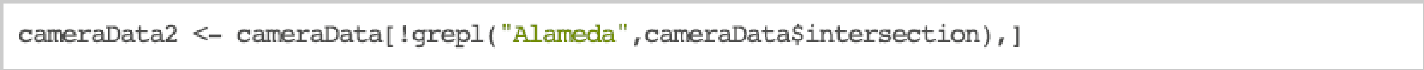
```
[1]  4  5 36
```



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



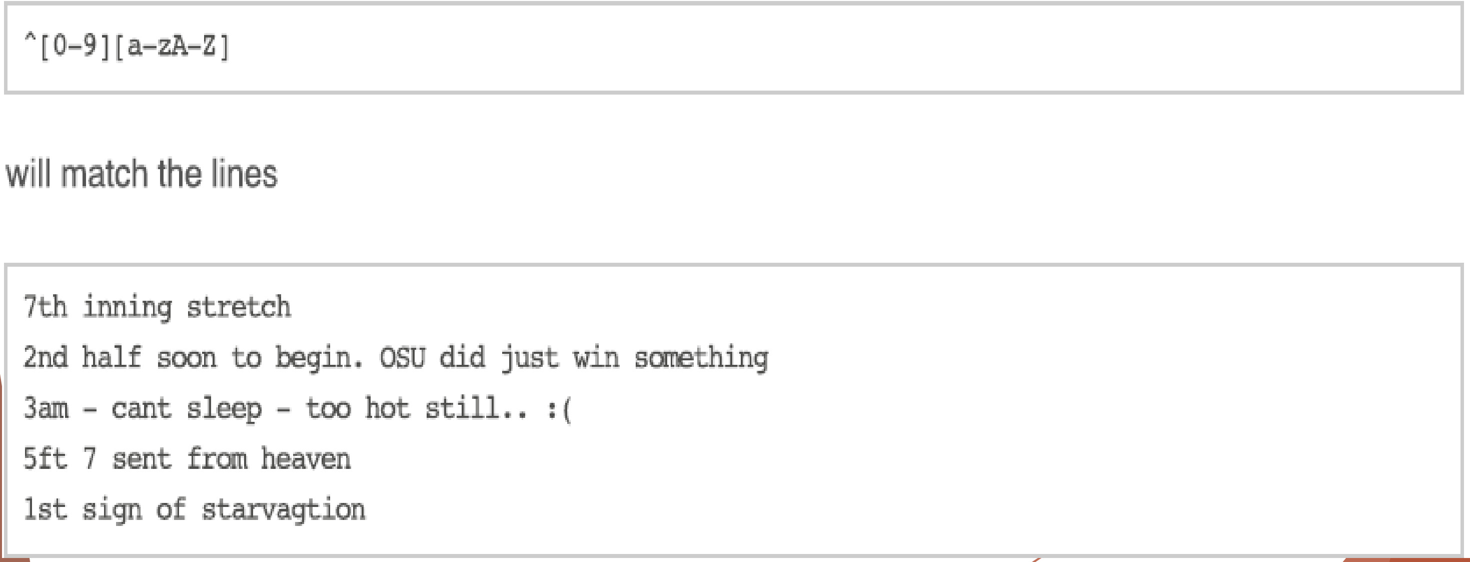
```

FALSE  TRUE 
   77     3 
```



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

Regular Expressions

- Regular expressions are used in many different languages and not restricted to R
- Composed of literals and metacharacters that represent sets or classes of characters / words
- Text processing via regular expressions is a powerful way to extract data from unstructured and semi structured data
- Used with string fucntions and search functions like grep(),grepl()

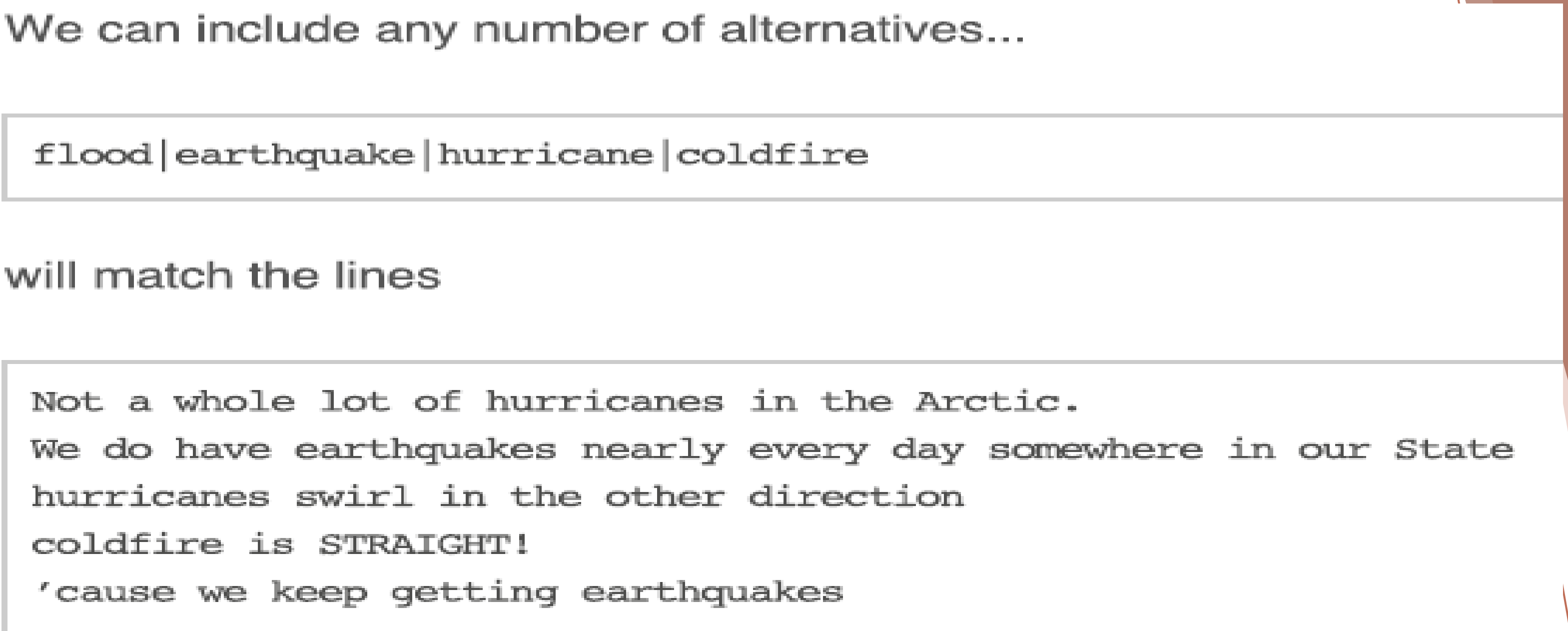


```
^[0-9][a-zA-Z]
```

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
```

Regular Expressions

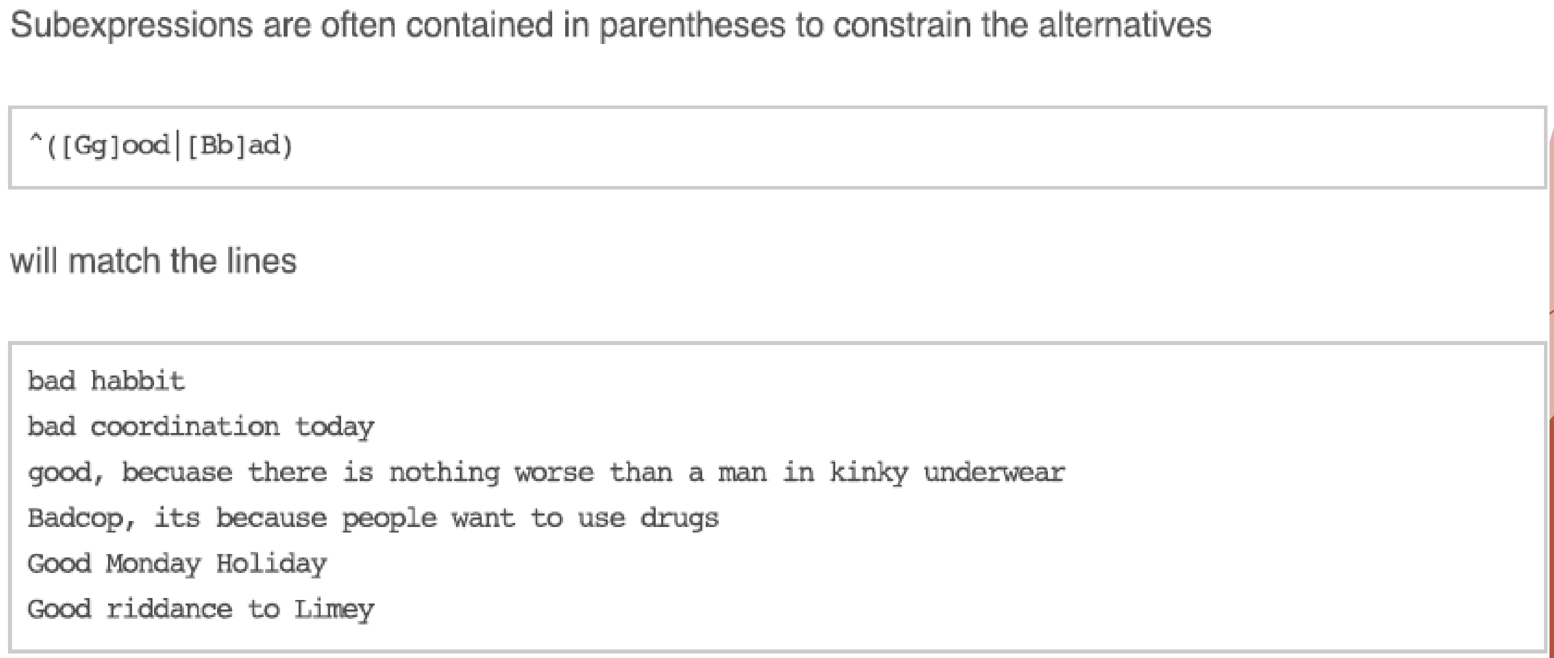


We can include any number of alternatives...

```
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
```



Subexpressions are often contained in parentheses to constrain the alternatives

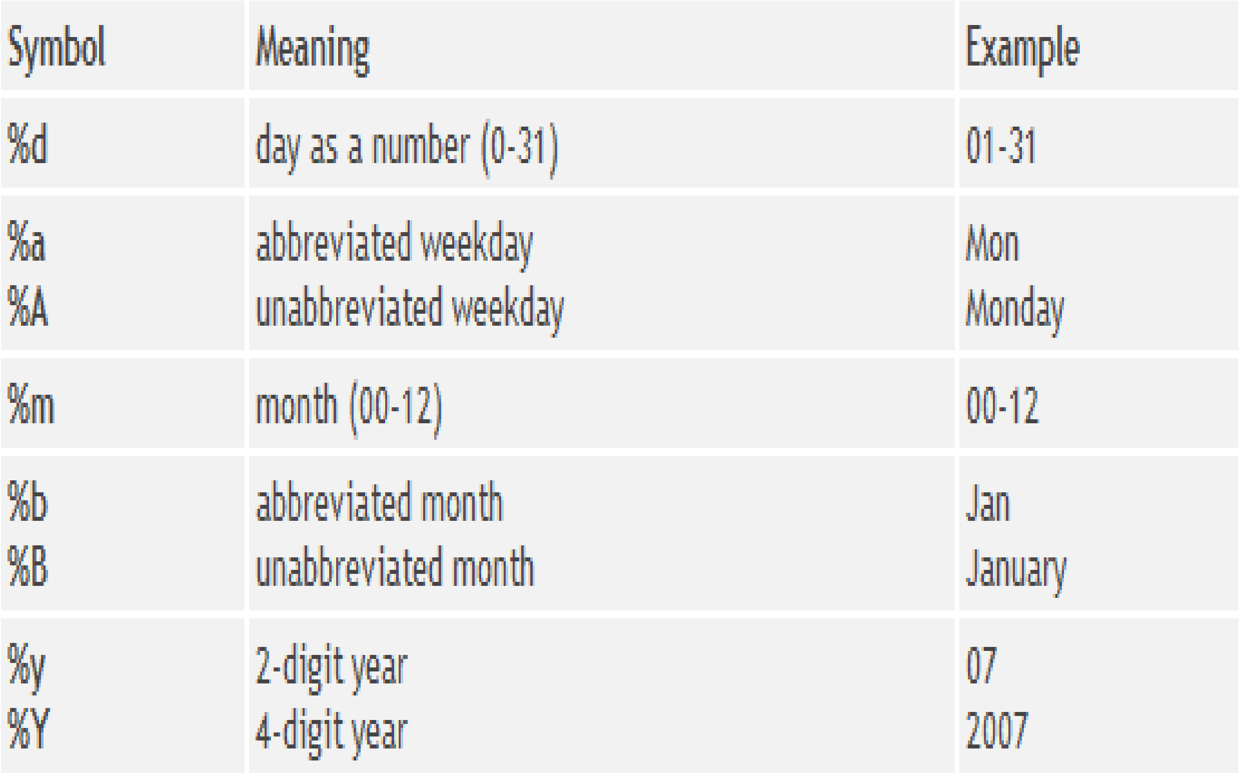
```
^([Gg]ood|[Bb]ad)
```

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
```

Working With Dates

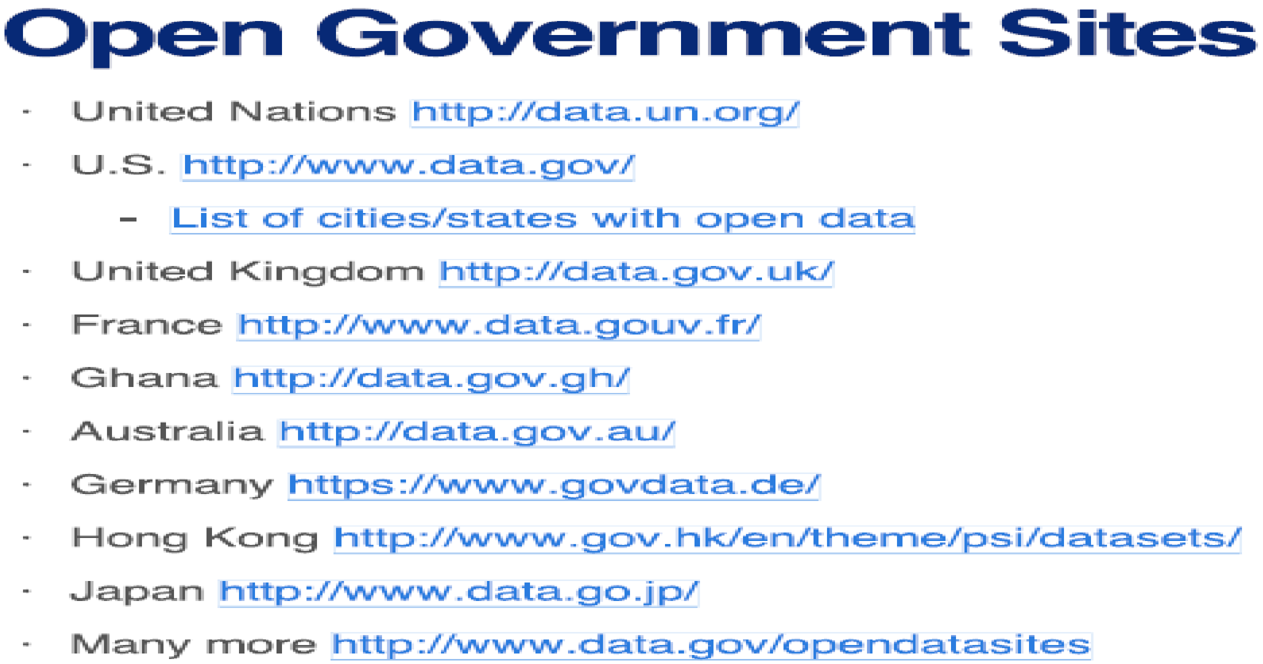
-
  ```
  # use as.Date( ) to convert strings to dates
  mydates <- as.Date(c("2007-06-22", "2004-02-13"))
  # number of days between 6/22/07 and 2/13/04
  days <- mydates[1] - mydates[2]
  ```
- **Sys.Date( )** returns today's date.
  date() returns the current date and time.



Symbol	Meaning	Example
%d	day as a number (0-31)	01-31
%a %A	abbreviated weekday unabbreviated weekday	Mon Monday
%m	month (00-12)	00-12
%b %B	abbreviated month unabbreviated month	Jan January
%y %Y	2-digit year 4-digit year	07 2007

Data Resources

- http://gapminder.org
- http://www.kaggle.com
- http://www.infochimps.com/marketplace
- http://www.asdfree.com
- Some API's with R interfaces twitter and twitter package, Facebook and RFacebook, Google maps and RGoogleMaps & rOpenSci



Open Government Sites

- United Nations http://data.un.org/
- U.S. http://www.data.gov/
  - List of cities/states with open data
- United Kingdom http://data.gov.uk/
- France http://www.data.gouv.fr/
- Ghana http://data.gov.gh/
- Australia http://data.gov.au/
- Germany https://www.govdata.de/
- Hong Kong http://www.gov.hk/en/theme/psi/datasets/
- Japan http://www.data.go.jp/
- Many more http://www.data.gov/opendatasites