DATA Preparation

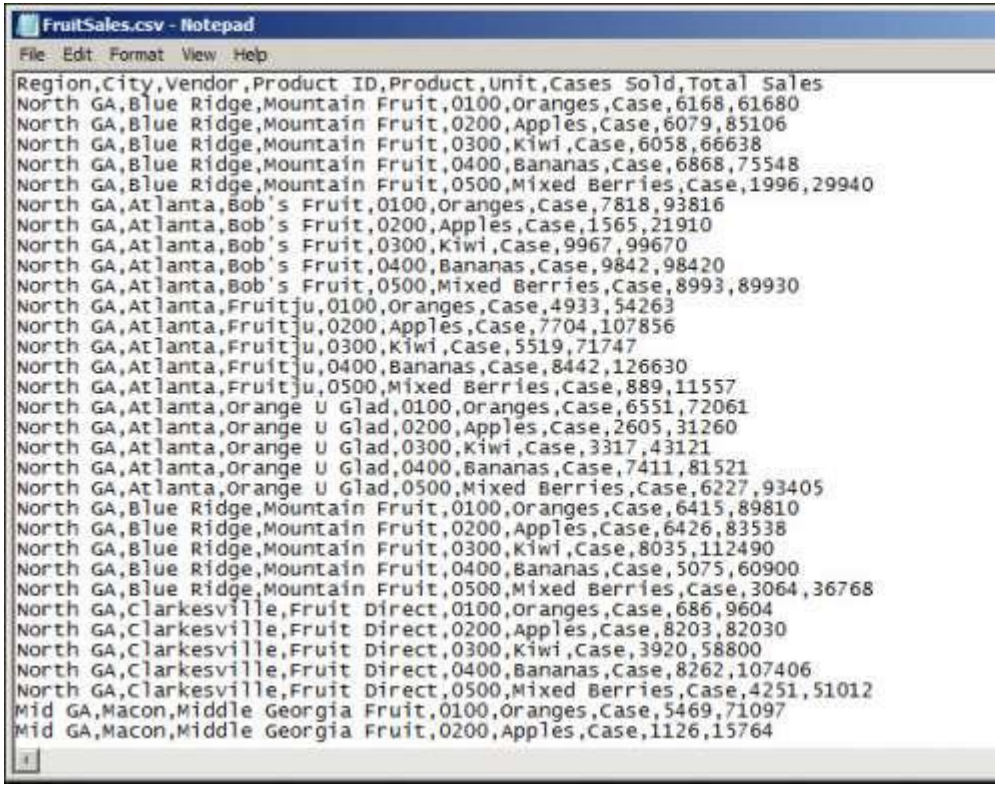
SIT22009: Data Science

Presented by Hyebong Choi

Loading Data into R

File format

- CSV: comma separated values
  - Columns are separated by comma (,)
  - Rows are separated by line
- Similarly TSV file
  - (tab separated values)
- Data a normal text file
  - With different delimiters
- You could read data from spread sheet
  - Like a MS excel


FruitSales.csv - Notepad

File Edit Format View Help

```
Region,City,Vendor,Product ID,Product,Unit,Cases Sold,Total Sales
North GA,Blue Ridge,Mountain Fruit,0100,Oranges,Case,6168,61680
North GA,Blue Ridge,Mountain Fruit,0200,Apples,Case,6079,85106
North GA,Blue Ridge,Mountain Fruit,0300,Kiwi,Case,6058,66638
North GA,Blue Ridge,Mountain Fruit,0400,Bananas,Case,6868,75548
North GA,Blue Ridge,Mountain Fruit,0500,Mixed Berries,Case,1996,29940
North GA,Atlanta,Bob's Fruit,0100,Oranges,Case,7818,93816
North GA,Atlanta,Bob's Fruit,0200,Apples,Case,1565,21910
North GA,Atlanta,Bob's Fruit,0300,Kiwi,Case,9967,99670
North GA,Atlanta,Bob's Fruit,0400,Bananas,Case,9842,98420
North GA,Atlanta,Bob's Fruit,0500,Mixed Berries,Case,8993,89930
North GA,Atlanta,Fruitju,0100,Oranges,Case,4933,54263
North GA,Atlanta,Fruitju,0200,Apples,Case,7704,107856
North GA,Atlanta,Fruitju,0300,Kiwi,Case,5519,71747
North GA,Atlanta,Fruitju,0400,Bananas,Case,8442,126630
North GA,Atlanta,Fruitju,0500,Mixed Berries,Case,889,11557
North GA,Atlanta,Orange U Glad,0100,Oranges,Case,6551,72061
North GA,Atlanta,Orange U Glad,0200,Apples,Case,2605,31260
North GA,Atlanta,Orange U Glad,0300,Kiwi,Case,3317,43121
North GA,Atlanta,Orange U Glad,0400,Bananas,Case,7411,81521
North GA,Atlanta,Orange U Glad,0500,Mixed Berries,Case,6227,93405
North GA,Blue Ridge,Mountain Fruit,0100,Oranges,Case,6415,89810
North GA,Blue Ridge,Mountain Fruit,0200,Apples,Case,6426,83538
North GA,Blue Ridge,Mountain Fruit,0300,Kiwi,Case,8035,112490
North GA,Blue Ridge,Mountain Fruit,0400,Bananas,Case,5075,60900
North GA,Blue Ridge,Mountain Fruit,0500,Mixed Berries,Case,3064,36768
North GA,Clarkesville,Fruit Direct,0100,Oranges,Case,686,9604
North GA,Clarkesville,Fruit Direct,0200,Apples,Case,8203,82030
North GA,Clarkesville,Fruit Direct,0300,Kiwi,Case,3920,58800
North GA,Clarkesville,Fruit Direct,0400,Bananas,Case,8262,107406
North GA,Clarkesville,Fruit Direct,0500,Mixed Berries,Case,4251,51012
Mid GA,Macon,Middle Georgia Fruit,0100,Oranges,Case,5469,71097
Mid GA,Macon,Middle Georgia Fruit,0200,Apples,Case,1126,15764
```

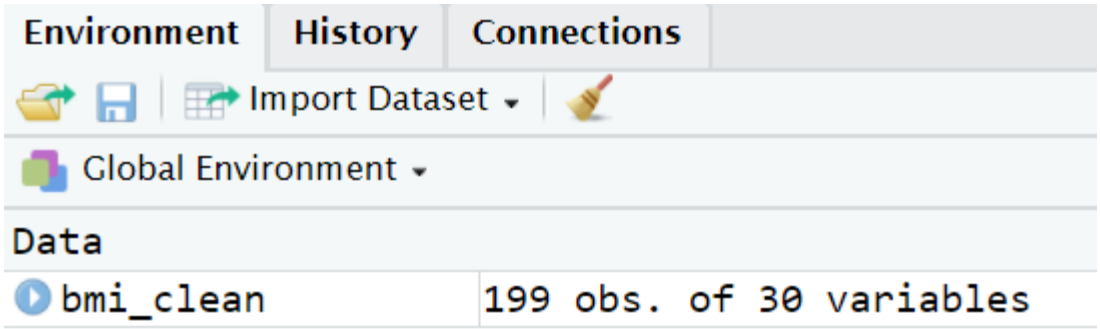
read.csv

- Function to read csv files

```
bmi_clean <- read.csv(file = 'bmi_clean.csv', header = TRUE)
```

File name

First line contains header info?



Data are now loaded

Try header = FALSE and see what happens!

read.csv

- Function to read csv files
  - You can look through what is in the data by clicking the object in the environment window

	Country	Y1980	Y1981	Y1982	Y1983	Y1984	Y1985	Y1986	Y1987
1	Afghanistan	21.48678	21.46552	21.45145	21.43822	21.42734	21.41222	21.40132	21.37679
2	Albania	25.22533	25.23981	25.25636	25.27176	25.27901	25.28669	25.29451	25.30217
3	Algeria	22.25703	22.34745	22.43647	22.52105	22.60633	22.69501	22.76979	22.84096
4	Andorra	25.66652	25.70868	25.74681	25.78250	25.81874	25.85236	25.89089	25.93414
5	Angola	20.94876	20.94371	20.93754	20.93187	20.93569	20.94857	20.96030	20.98025
6	Antigua and Barbuda	23.31424	23.39054	23.45883	23.53735	23.63584	23.73109	23.83449	23.93649
7	Argentina	25.37913	25.44951	25.50242	25.55644	25.61271	25.66593	25.72364	25.78529

Load data from web

```
uciCar <- read.csv('http://www.win-vector.com/dfiles/car.data.csv',
header = T)
```

	buying	maint	doors	persons	lug_boot	safety	rating
1	vhigh	vhigh	2	2	small	low	unacc
2	vhigh	vhigh	2	2	small	med	unacc
3	vhigh	vhigh	2	2	small	high	unacc
4	vhigh	vhigh	2	2	med	low	unacc
5	vhigh	vhigh	2	2	med	med	unacc
6	vhigh	vhigh	2	2	med	high	unacc
7	vhigh	vhigh	2	2	big	low	unacc
8	vhigh	vhigh	2	2	big	med	unacc
9	vhigh	vhigh	2	2	big	high	unacc
10	vhigh	vhigh	2	4	small	low	unacc
11	vhigh	vhigh	2	4	small	med	unacc
12	vhigh	vhigh	2	4	small	high	unacc
13	vhigh	vhigh	2	4	med	low	unacc
14	vhigh	vhigh	2	4	med	med	unacc

Car evaluation dataset

read.table

- Functions to read data in a free format

```
d <- read.table('http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data', stringsAsFactors = F, header = F)
```

```
> head(d)
  V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17 V18  V19  V20 V21
1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173   1 A192 A201   1
2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173   1 A191 A201   2
3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172   2 A191 A201   1
4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173   2 A191 A201   1
5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173   2 A191 A201   2
6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172   2 A192 A201   1
```

- read.csv, read.table convert all text values into factors by default
  - stringAsFactors option is to cancel that
- read.table function with sep option to specify the delimiters
  - can be comma(,) tab(/t) or other thing

Try to read csv file with read.table function!

Exploring Raw Data

Exploring raw data

- First thing to do once data are acquired to analyze
  1. Understand the structure of your data
  2. Look at your data
  3. Visualize your data

- Step further toward data cleaning

Understanding the structure of your data

```
# Load the bmi data
bmi <- read.csv('bmi_clean.csv')

# View its class
class(bmi)

## [1] "data.frame"

# View its dimensions
dim(bmi)

## [1] 199  30
```

Rows **Columns**

```
# Look at column names
names(bmi)

##  [1] "Country" "Y1980"   "Y1981"   "Y1982"   "Y1983"   "Y1984"   "Y1985"
##  [8] "Y1986"   "Y1987"   "Y1988"   "Y1989"   "Y1990"   "Y1991"   "Y1992"
## [15] "Y1993"   "Y1994"   "Y1995"   "Y1996"   "Y1997"   "Y1998"   "Y1999"
## [22] "Y2000"   "Y2001"   "Y2002"   "Y2003"   "Y2004"   "Y2005"   "Y2006"
## [29] "Y2007"   "Y2008"
```

Understanding the structure of your data

```
str(bmi)
```

```
## 'data.frame':    199 obs. of  30 variables:
##  $ Country: Factor w/ 199 levels "Afghanistan",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Y1980  : num  21.5 25.2 22.3 25.7 20.9 ...
##  $ Y1981  : num  21.5 25.2 22.3 25.7 20.9 ...
##  $ Y1982  : num  21.5 25.3 22.4 25.7 20.9 ...
##  $ Y1983  : num  21.4 25.3 22.5 25.8 20.9 ...
##  $ Y1984  : num  21.4 25.3 22.6 25.8 20.9 ...
##  $ Y1985  : num  21.4 25.3 22.7 25.9 20.9 ...
##  $ Y1986  : num  21.4 25.3 22.8 25.9 21 ...
##  $ Y1987  : num  21.4 25.3 22.8 25.9 21 ...
##  $ Y1988  : num  21.3 25.3 22.9 26 21 ...
##  $ Y1989  : num  21.3 25.3 23 26 21.1 ...
##  $ Y1990  : num  21.2 25.3 23 26.1 21.1 ...
##  $ Y1991  : num  21.2 25.3 23.1 26.2 21.1 ...
##  $ Y1992  : num  21.1 25.2 23.2 26.2 21.1 ...
##  $ Y1993  : num  21.1 25.2 23.3 26.3 21.1 ...
##  $ Y1994  : num  21 25.2 23.3 26.4 21.1 ...
##  $ Y1995  : num  20.9 25.3 23.4 26.4 21.2 ...
##  $ Y1996  : num  20.9 25.3 23.5 26.5 21.2 ...
##  $ Y1997  : num  20.8 25.3 23.5 26.6 21.2 ...
##  $ Y1998  : num  20.8 25.4 23.6 26.7 21.3 ...
##  $ Y1999  : num  20.8 25.5 23.7 26.8 21.3 ...
```

Understanding the structure of your data

```
# Load dplyr
library(dplyr)

# View structure of lunch, the dplyr way
glimpse(bmi)
```

```
## Observations: 199
## Variables: 30
## $ Country <fctr> Afghanistan, Albania, Algeria, Andorra, Angola, Antig...
## $ Y1980   <dbl> 21.48678, 25.22533, 22.25703, 25.66652, 20.94876, 23.3...
## $ Y1981   <dbl> 21.46552, 25.23981, 22.34745, 25.70868, 20.94371, 23.3...
## $ Y1982   <dbl> 21.45145, 25.25636, 22.43647, 25.74681, 20.93754, 23.4...
## $ Y1983   <dbl> 21.43822, 25.27176, 22.52105, 25.78250, 20.93187, 23.5...
## $ Y1984   <dbl> 21.42734, 25.27901, 22.60633, 25.81874, 20.93569, 23.6...
## $ Y1985   <dbl> 21.41222, 25.28669, 22.69501, 25.85236, 20.94857, 23.7...
## $ Y1986   <dbl> 21.40132, 25.29451, 22.76979, 25.89089, 20.96030, 23.8...
## $ Y1987   <dbl> 21.37679, 25.30217, 22.84096, 25.93414, 20.98025, 23.9...
## $ Y1988   <dbl> 21.34018, 25.30450, 22.90644, 25.98477, 21.01375, 24.0...
## $ Y1989   <dbl> 21.29845, 25.31944, 22.97931, 26.04450, 21.05269, 24.1...
## $ Y1990   <dbl> 21.24818, 25.32357, 23.04600, 26.10936, 21.09007, 24.2...
## $ Y1991   <dbl> 21.20269, 25.28452, 23.11333, 26.17912, 21.12136, 24.3...
```

…

Understanding the structure of your data

```
# View a summary
summary(bmi)
```

```
##                 Country        Y1980           Y1981           Y1982
##  Afghanistan        :  1   Min.   :19.01   Min.   :19.04   Min.   :19.07
##  Albania            :  1   1st Qu.:21.27   1st Qu.:21.31   1st Qu.:21.36
##  Algeria            :  1   Median :23.31   Median :23.39   Median :23.46
##  Andorra            :  1   Mean   :23.15   Mean   :23.21   Mean   :23.26
##  Angola             :  1   3rd Qu.:24.82   3rd Qu.:24.89   3rd Qu.:24.94
##  Antigua and Barbuda:  1   Max.   :28.12   Max.   :28.36   Max.   :28.58
##  (Other)            :193
##      Y1983           Y1984           Y1985           Y1986
##  Min.   :19.10   Min.   :19.13   Min.   :19.16   Min.   :19.20
##  1st Qu.:21.42   1st Qu.:21.45   1st Qu.:21.47   1st Qu.:21.49
##  Median :23.57   Median :23.64   Median :23.73   Median :23.82
##  Mean   :23.32   Mean   :23.37   Mean   :23.42   Mean   :23.48
##  3rd Qu.:25.02   3rd Qu.:25.06   3rd Qu.:25.11   3rd Qu.:25.20
##  Max.   :28.82   Max.   :29.05   Max.   :29.28   Max.   :29.52
##
```

…

Understanding the structure of your data

- class() : Class of data object
- dim() : Dimensions of data
- names() : Column names
- str() : Preview of data with helpful details
- glimpse() : Better version of str() from dplyr
- summary() : Summary of data

Looking at your data

```
# View the top
head(bmi)
```

```
##                Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 1          Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734
## 2              Albania 25.22533 25.23981 25.25636 25.27176 25.27901
## 3              Algeria 22.25703 22.34745 22.43647 22.52105 22.60633    …
## 4              Andorra 25.66652 25.70868 25.74681 25.78250 25.81874
## 5               Angola 20.94876 20.94371 20.93754 20.93187 20.93569
## 6 Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584
```

```
# View the top 10 records
head(bmi, n = 10)
```

```
##                Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 1           Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734
## 2               Albania 25.22533 25.23981 25.25636 25.27176 25.27901
## 3               Algeria 22.25703 22.34745 22.43647 22.52105 22.60633
## 4               Andorra 25.66652 25.70868 25.74681 25.78250 25.81874     …
## 5                Angola 20.94876 20.94371 20.93754 20.93187 20.93569
## 6  Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584
## 7             Argentina 25.37913 25.44951 25.50242 25.55644 25.61271
## 8               Armenia 23.82469 23.86401 23.91023 23.95649 24.00181
## 9             Australia 24.92729 25.00216 25.07660 25.14938 25.22894
## 10              Austria 24.84097 24.88110 24.93482 24.98118 25.02208
```

Looking at your data

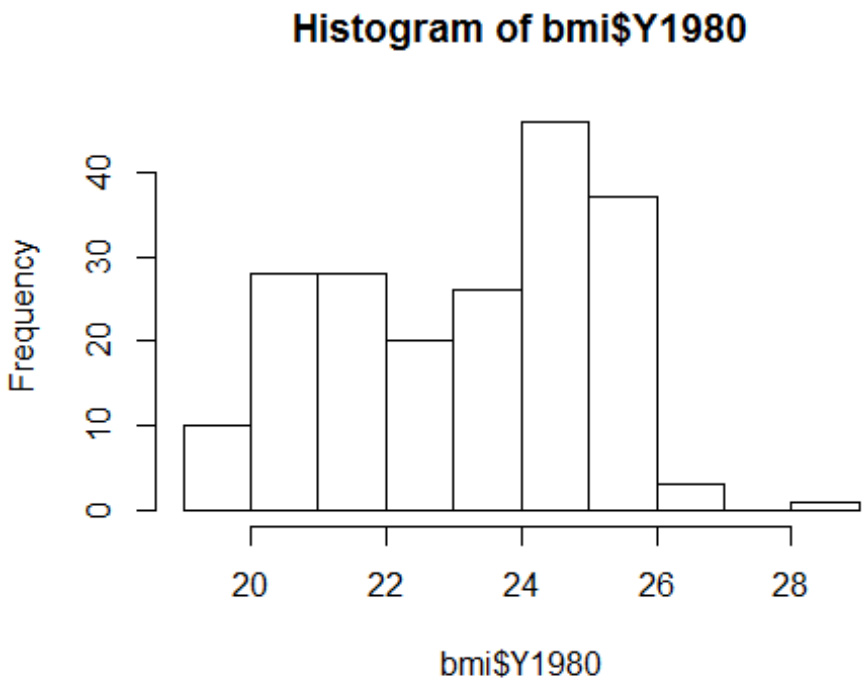
```
# View the bottom
tail(bmi)
```

```
##                Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 194          Venezuela 24.58052 24.69666 24.80082 24.89208 24.98440
## 195            Vietnam 19.01394 19.03902 19.06804 19.09675 19.13046
## 196 West Bank and Gaza 24.31624 24.40192 24.48713 24.57107 24.65582
## 197        Yemen, Rep. 22.90384 22.96813 23.02669 23.07279 23.12566
## 198             Zambia 19.66295 19.69512 19.72538 19.75420 19.78070
## 199           Zimbabwe 21.46989 21.48867 21.50738 21.52936 21.53383
```

...

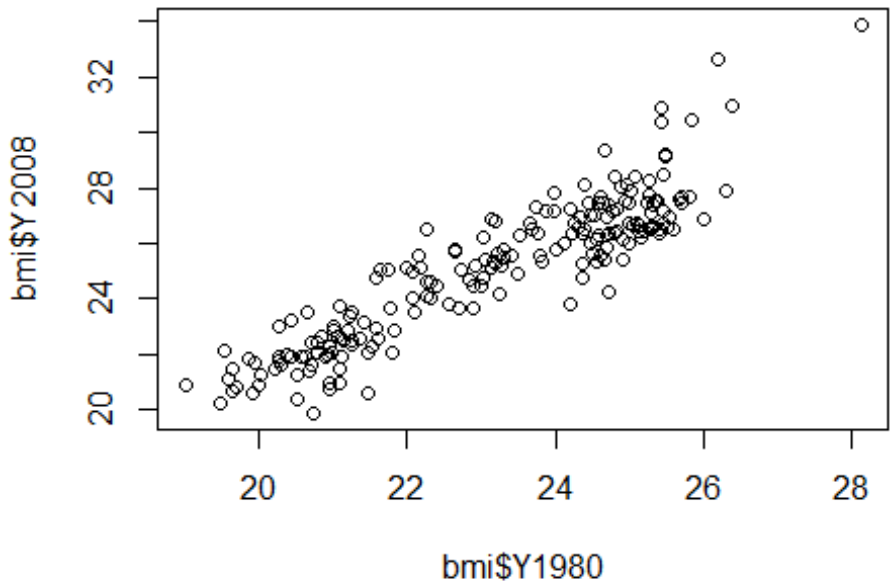
Visualizing your data

```
# View histogram
hist(bmi$Y1980)
```



Visualizing your data

```
# View plot of two variables
plot(x = bmi$Y1980, y= bmi$Y2008)
```



- **Looking at your data**
  - head() - View top of dataset
  - tail() - View bo!om of dataset
  - print() - View entire dataset (not recommended!)

- **Visualizing your data**
  - hist() - View histogram of a single variable
  - plot() - View plot of two variables

Data Cleaning

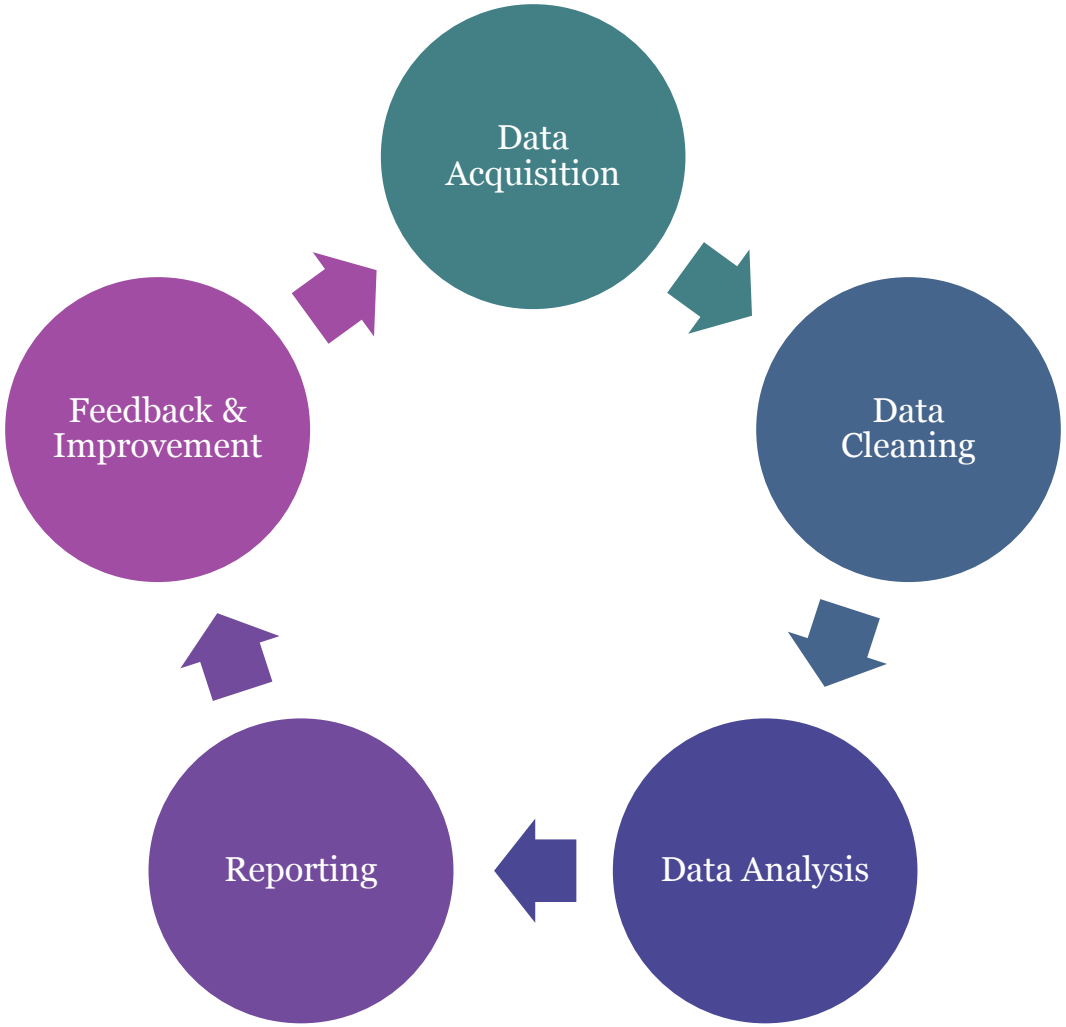
A look at some dirty data

```
> head(weather)
  X year month           measure X1 X2 X3 X4 X5 X6 X7 X8 X9 ...
1 1 2014    12  Max.TemperatureF 64 42 51 43 42 45 38 29 49 ...
2 2 2014    12 Mean.TemperatureF 52 38 44 37 34 42 30 24 39 ...
3 3 2014    12  Min.TemperatureF 39 33 37 30 26 38 21 18 29 ...
4 4 2014    12    Max.Dew.PointF 46 40 49 24 37 45 36 28 49 ...
5 5 2014    12    MeanDew.PointF 40 27 42 21 25 40 20 16 41 ...
6 6 2014    12     Min.DewpointF 26 17 24 13 12 36 -3  3 28 ...

> tail(weather)
      X year month            measure   X1   X2   X3   X4 ...
281 281 2015    12 Mean.Wind.SpeedMPH    6 <NA> <NA> <NA> ...
282 282 2015    12  Max.Gust.SpeedMPH   17 <NA> <NA> <NA> ...
283 283 2015    12    PrecipitationIn 0.14 <NA> <NA> <NA> ...
284 284 2015    12         CloudCover    7 <NA> <NA> <NA> ...
285 285 2015    12             Events Rain <NA> <NA> <NA> ...
286 286 2015    12     WindDirDegrees  109 <NA> <NA> <NA> ...
```

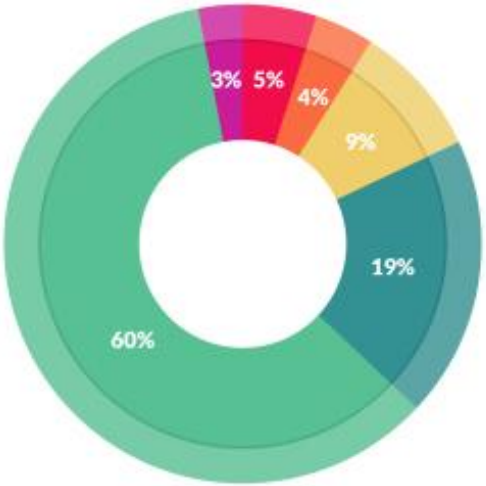
We need to get this thing ready for analysis!

“**Data Cleaning**”

Why cleaning data?

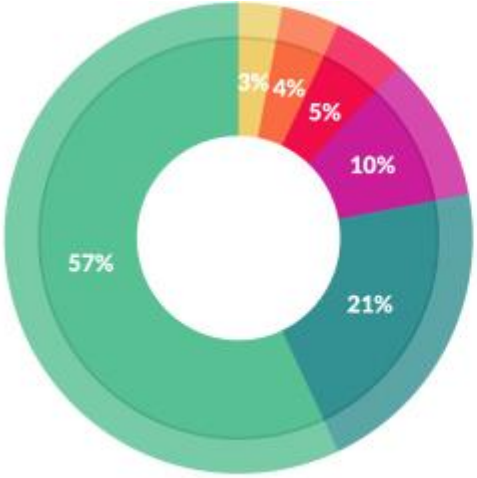


Why cleaning data?



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*



What's the least enjoyable part of data science?

- *Building training sets: 10%*
- *Cleaning and organizing data: 57%*
- *Collecting data sets: 21%*
- *Mining data for patterns: 3%*
- *Refining algorithms: 4%*
- *Other: 5%*

https://whatsthebigdata.com/2016/05/01/data-scientists-spend-most-of-their-time-cleaning-data/

Tidying data

Principles of tidy data

name	age	eye_color	height
Jake	34	Other	6'1"
Alice	55	Blue	5'9"
Tim	76	Brown	5'7"
Denise	19	Other	5'1"

Observation

Variable or Attribute

- A dataset is a collection of values
- Each value belongs to a variable and an observation
- Each observation forms a row, each variable forms a column, and each type of observational unit forms a table

1. Observations as rows
2. Variables as columns
3. One type of observational unit per table

Some Dirty Data

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"

- "brown", "blue", and "other" are supposed to be values for a variable eye color

Common symptoms of messy data

Column headers are values, not variable names

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"

name	age	eye_color	height
Jake	34	Other	6'1"
Alice	55	Blue	5'9"
Tim	76	Brown	5'7"
Denise	19	Other	5'1"

Variables are stored in both rows and columns

name	measurement	value
Jake	n_dogs	1
Jake	n_cats	0
Jake	n_birds	1
Alice	n_dogs	1
Alice	n_cats	2
Alice	n_birds	0

name	n_dogs	n_cats	n_birds
Jake	1	0	1
Alice	1	2	0

Multiple variables are stored in one column

name	sex_age	eye_color	height
Jake	M.34	Other	6'1"
Alice	F.55	Blue	5'9"
Tim	M.76	Brown	5'7"
Denise	F.19	Other	5'1"

name	sex	age	eye_color	height
Jake	M	34	Other	6'1"
Alice	F	55	Blue	5'9"
Tim	M	76	Brown	5'7"
Denise	F	19	Other	5'1"

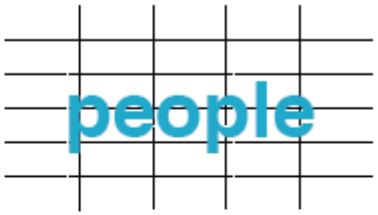
Other common symptoms

- A single observational unit is stored in multiple tables
  - Need to be joined (could use merge function)
- Multiple types of observational units are stored in the same table

name	age	height
Jake	34	6'1"
Jake	34	6'1"
Alice	55	5'9"
Alice	55	5'9"
Alice	55	5'9"

pet_name	pet_type	pet_height
Larry	Dog	25"
Chirp	Bird	3"
Wally	Dog	30"
Sugar	Cat	10"
Spice	Cat	12"

Alice's name, age, and height are duplicated 3x



tidyr package

Gather columns into key-value pairs

```
library(tidyr)
wide_df <- data.frame(col = c('X', 'Y'), A = c(1,4), B = c(2,5), C = c(3,6))

# Look at wide_df
wide_df
```

```
##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

```
# Gather the columns of wide_df
gather(wide_df, my_key, my_val, -col)
```

```
##   col my_key my_val
## 1   X      A      1
## 2   Y      A      4
## 3   X      B      2
## 4   Y      B      5
## 5   X      C      3
## 6   Y      C      6
```

gather(data, key, value, …)

data: a data frame

key: bare name of new key column

value: bare name of new value column

…: bare names of columns to gather (or not)

Spread key-value pairs into columns

```
long_df <- gather(wide_df, my_key, my_val, -col)

# Look at long_df
long_df
```

```
##   col my_key my_val
## 1   X      A      1
## 2   Y      A      4
## 3   X      B      2
## 4   Y      B      5
## 5   X      C      3
## 6   Y      C      6
```

spread(data, key, value)

data: a data frame

key: bare name of column containing keys

value: bare name of column containing values

```
# Spread the key-value pairs of long_df
spread(long_df, my_key, my_val)
```

```
##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

Separate columns

```
treatments <- data.frame(patient = rep(c('X', 'Y'),3) ,
                         treatment = rep(c('A', 'B'), each = 3),
                         year_mo = rep(c('2010-10', '2012-08', '2014-12'), each = 2),
                         response = c(1,4,2,5,3,6))

# View the treatments data
treatments
```

```
##   patient treatment year_mo response
## 1       X         A 2010-10        1
## 2       Y         A 2010-10        4
## 3       X         A 2012-08        2
## 4       Y         B 2012-08        5
## 5       X         B 2014-12        3
## 6       Y         B 2014-12        6
```

separate(data, col, into)

data: a data frame — **sep = "-"**

col: bare name of column to separate

into: character vector of new column names

```
# Separate year_mo into two columns
separate(treatments, year_mo, c("year", "month"))
```

```
##   patient treatment year month response
## 1       X         A 2010    10        1
## 2       Y         A 2010    10        4
## 3       X         A 2012    08        2
## 4       Y         B 2012    08        5
## 5       X         B 2014    12        3
## 6       Y         B 2014    12        6
```

Unite columns

```
treatments2 <- separate(treatments, year_mo, c("year", "month"))

# View treatments2 data
treatments2
```

```
##   patient treatment year month response
## 1       X         A 2010    10        1
## 2       Y         A 2010    10        4
## 3       X         A 2012    08        2
## 4       Y         B 2012    08        5
## 5       X         B 2014    12        3
## 6       Y         B 2014    12        6
```

```
# Unite year and month to form year_mo column
unite(treatments2, year_mo, year, month)
```

```
##   patient treatment year_mo response
## 1       X         A 2010_10        1
## 2       Y         A 2010_10        4
## 3       X         A 2012_08        2
## 4       Y         B 2012_08        5
## 5       X         B 2014_12        3
## 6       Y         B 2014_12        6
```

unite(data, col, …)

data: a data frame **sep = "-"**

col: bare name of new column

…: bare names of columns to unite

Try different sep option (by default sep = '_')

Summary of key tidyr functions

- gather() - Gather columns into key-value pairs
- spread() - Spread key-value pairs into columns
- separate() - Separate one column into multiple
- unite() - Unite multiple columns into one

Exercise 1

- Data in the “bmi_clean.csv” file represent average bmi of sample population of each country measured in year for 1980 to 2008
- Columns named Y#### implies the year of the bmi values are observed hence it is rather values of “year” variable than variable names
- Let us practice gather and separate function to tidy the dataset

```
bmi <- read.csv(file = 'bmi_clean.csv', header = TRUE)

head(bmi)
```

```
##                Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 1          Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734
## 2              Albania 25.22533 25.23981 25.25636 25.27176 25.27901
## 3              Algeria 22.25703 22.34745 22.43647 22.52105 22.60633
## 4              Andorra 25.66652 25.70868 25.74681 25.78250 25.81874
## 5               Angola 20.94876 20.94371 20.93754 20.93187 20.93569
## 6 Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584
```

```
bmi_long <- gather(bmi, <fill in this part>)

head(bmi_long)
```

```
##                Country  year  bmi_val
## 1          Afghanistan Y1980 21.48678
## 2              Albania Y1980 25.22533
## 3              Algeria Y1980 22.25703
## 4              Andorra Y1980 25.66652
## 5               Angola Y1980 20.94876
## 6 Antigua and Barbuda Y1980 23.31424
```

```
bmi_wide <- spread(bmi_long, <fill in this part>)

head(bmi_wide)
```

```
##                Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 1          Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734
## 2              Albania 25.22533 25.23981 25.25636 25.27176 25.27901
## 3              Algeria 22.25703 22.34745 22.43647 22.52105 22.60633
## 4              Andorra 25.66652 25.70868 25.74681 25.78250 25.81874
## 5               Angola 20.94876 20.94371 20.93754 20.93187 20.93569
## 6 Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584
```

Exercise 2

- Load Data in the “bmi_cc.csv” file. The **Country_ISO** column of bmi_cc has the name of each country as well its two-letter ISO country code, separated by a forward slash.

1. Apply the **separate()** function to Separate **Country_ISO** into two columns: **Country** and **ISO**

- Save the result to a new object called **bmi_cc_clean**

2. Apply the **unite()** function to **bmi_cc_clean** to Reunite the **Country** and **ISO** columns into a single column called **Country_ISO** using sep option of dash (**-**)

- Save the result as **bmi_cc2**

See how two datasets bmi_cc and bmi_cc2 are different

```
bmi_cc <- read.csv(file = 'bmi_cc.csv', header = TRUE)

head(bmi_cc)
```

```
##              Country_ISO  year  bmi_val
## 1         Afghanistan/AF Y1980 21.48678
## 2             Albania/AL Y1980 25.22533
## 3             Algeria/DZ Y1980 22.25703
## 4             Andorra/AD Y1980 25.66652
## 5              Angola/AO Y1980 20.94876
## 6 Antigua and Barbuda/AG Y1980 23.31424
```

```
# Apply separate() to bmi_cc
bmi_cc_clean <- separate(bmi_cc, <fill in this part>)

# Print the head of the result
head(bmi_cc_clean)
```

```
##              Country ISO  year  bmi_val
## 1         Afghanistan  AF Y1980 21.48678
## 2             Albania  AL Y1980 25.22533
## 3             Algeria  DZ Y1980 22.25703
## 4             Andorra  AD Y1980 25.66652
## 5              Angola  AO Y1980 20.94876
## 6 Antigua and Barbuda  AG Y1980 23.31424
```

```
# Apply unite() to bmi_cc_clean
bmi_cc2 <- unite(bmi_cc_clean, <fill in this part>)

# View the head of the result
head(bmi_cc2)
```

```
##              Country_ISO  year  bmi_val
## 1         Afghanistan-AF Y1980 21.48678
## 2             Albania-AL Y1980 25.22533
## 3             Algeria-DZ Y1980 22.25703
## 4             Andorra-AD Y1980 25.66652
## 5              Angola-AO Y1980 20.94876
## 6 Antigua and Barbuda-AG Y1980 23.31424
```

Type conversion

Types of variables in R

- character: "treatment", "123", "A"
- numeric: 23.44, 120, NaN, Inf
- integer: 4L, 1123L
- factor: factor("Hello"), factor(8)
- logical: TRUE, FALSE, NA

Type Check-Up and Conversion

```
class("hello")
## [1] "character"
class(3.844)
## [1] "numeric"
class(77L)
## [1] "integer"
class(factor("yes"))
## [1] "factor"
class(TRUE)
## [1] "logical"
```

```
as.character(2016)
## [1] "2016"
as.numeric(TRUE)
## [1] 1
as.integer(99)
## [1] 99
as.factor("something")
## [1] something
## Levels: something
as.logical(0)
## [1] FALSE
```

lubridate

- Package to convert strings into dates

```
# Load the lubridate package
library(lubridate)

# Experiment with basic lubridate functions
ymd("2015-08-25")

## [1] "2015-08-25"

ymd("2015 August 25")

## [1] "2015-08-05"

mdy("August 25, 2015")

## [1] "2015-08-05"

hms("13:33:09")

## [1] "13H 33M 9S"

ymd_hms("2015/08/25 13.33.09")

## [1] "2015-08-25 13:33:09 UTC"
```

String manipulation

stringr

- Package for string manipulation
- Key functions
  - str_trim() - Trim leading and trailing white space
  - str_pad() - Pad with additional characters
  - str_detect() - Detect a pattern
  - str_replace() - Find and replace a pattern

stringr

```
library(stringr)

# Trim leading and trailing white space
str_trim(" this is a test ")

## [1] "this is a test"

# Pad string with zeros
str_pad("24493", width = 7, side = "left", pad = "0")

## [1] "0024493"

# Create character vector of names
friends <- c("Sarah", "Tom", "Alice")

# Search for string in vector
str_detect(friends, "Alice")

## [1] FALSE FALSE  TRUE

# Replace string in vector
str_replace(friends, "Alice", "David")

## [1] "Sarah" "Tom"   "David"
```

Other helpful functions in base R

- **tolower()** Make all lowercase
- **toupper()** Make all uppercase

```
# Make all lowercase
tolower("I AM TALKING LOUDLY!!")
```

```
## [1] "i am talking loudly!!"
```

```
# Make all uppercase
toupper("I am whispering...")
```

```
## [1] "I AM WHISPERING..."
```

Exercise 1

Load data of students2.csv into R. We have a date of birth of each student in the dob column. There's another column called nurse_visit, which gives a timestamp for each student's most recent visit to the school nurse.

1. Preview students2 with str(). Notice that dob and nurse_visit are both stored as character
2. Load the lubridate package
3. Coerce dob to a date (with no time)
4. Coerce nurse_visit to a date and time
5. Use str() to see the changes to students2

```
students2 <- read.csv(file = 'students2.csv', header = TRUE, stringsAsFactors = F)

str(students2)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ school      : chr  "GP" "GP" "GP" "GP" ...
##  $ sex         : chr  "F" "F" "F" "F" ...
##  $ dob         : chr  "2000-06-05" "1999-11-25" "1998-02-02" "1997-12-20" ...
##  $ address     : chr  "U" "U" "U" "U" ...
##  $ famsize     : chr  "GT3" "GT3" "LE3" "GT3" ...
…

##  $ goout       : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc        : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc        : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health      : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ nurse_visit: chr  "2014-04-10 14:59:54" "2015-03-12 14:59:54" "2015-09-21 14:59:54" "
##  $ absences    : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ Grades      : chr  "5/6/6" "5/5/6" "7/8/10" "15/14/15" ...
```

```
# Coerce dob to a date (with no time)
students2$dob <- <fill in this part>

# Coerce nurse_visit to a date and time
students2$nurse_visit <- <fill in this part>
```

```
# Look at students2 once more with str()
str(students2)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ school     : chr  "GP" "GP" "GP" "GP" ...
##  $ sex        : chr  "F" "F" "F" "F" ...
##  $ dob        : Date, format: "2000-06-05" "1999-11-25" ...
##  $ address    : chr  "U" "U" "U" "U" ...
##  $ famsize    : chr  "GT3" "GT3" "LE3" "GT3" ...
##  $ goout      : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc       : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc       : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health     : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ nurse_visit: POSIXct, format: "2014-04-10 14:59:54" "2015-03-12 14:59:54" ...
##  $ absences   : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ Grades     : chr  "5/6/6" "5/5/6" "7/8/10" "15/14/15" ...
```

Exercise 2

1. Look at the head()
2. Detect all dates of birth (dob) in 1997 using str_detect(). This should return a vector of TRUE and FALSE values.
3. Replace all instances of "F" with "Female" in students2$sex
4. Replace all instances of "M" with "Male" in students2$sex
5. View the head() of students2 to see the result of these replacements

```
library(stringr)

# Look at the head of students2
head(students2)
```

```
##   X school sex        dob address famsize Pstatus Medu Fedu     Mjob
## 1 1     GP   F 2000-06-05       U     GT3       A    4    4  at_home
## 2 2     GP   F 1999-11-25       U     GT3       T    1    1  at_home
## 3 3     GP   F 1998-02-02       U     LE3       T    1    1  at_home
## 4 4     GP   F 1997-12-20       U     GT3       T    4    2   health
## 5 5     GP   F 1998-10-04       U     GT3       T    3    3    other
## 6 6     GP   M 1999-06-16       U     LE3       T    4    3 services
##       Fjob     reason guardian traveltime studytime failures schoolsup
## 1  teacher     course   mother          2         2        0       yes
## 2    other     course   father          1         2        0        no
## 3    other      other   mother          1         2        3       yes
## 4 services       home   mother          1         3        0        no
## 5    other       home   father          1         2        0        no
## 6    other reputation   mother          1         2        0        no
##   famsup paid activities nursery higher internet romantic famrel freetime
## 1     no   no         no     yes    yes       no       no      4        3
## 2    yes   no         no      no    yes      yes       no      5        3
## 3     no  yes         no     yes    yes      yes       no      4        3
## 4    yes  yes        yes     yes    yes      yes      yes      3        2
## 5    yes  yes         no     yes    yes       no       no      4        3
## 6    yes  yes        yes     yes    yes      yes       no      5        4
##   goout Dalc Walc health         nurse_visit absences   Grades
## 1     4    1    1      3 2014-04-10 14:59:54        6    5/6/6
## 2     3    1    1      3 2015-03-12 14:59:54        4    5/5/6
## 3     2    2    3      3 2015-09-21 14:59:54       10   7/8/10
## 4     2    1    1      5 2015-09-03 14:59:54        2 15/14/15
## 5     2    1    2      5 2015-04-07 14:59:54        4  6/10/10
```

```
# Detect all dates of birth (dob) in 1997
str_detect(<fill in this part>)
```

```
##   [1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
##  [12] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [23]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
```

…

```
# In the sex column, replace "F" with "Female"...
students2$sex <- str_replace(<fill in this part>)

# ...And "M" with "Male"
students2$sex <- str_replace(<fill in this part>)

# View the head of students2
head(students2)
```

```
##   X school    sex        dob address famsize Pstatus Medu Fedu     Mjob
## 1 1     GP Female 2000-06-05       U     GT3       A    4    4  at_home
## 2 2     GP Female 1999-11-25       U     GT3       T    1    1  at_home
## 3 3     GP Female 1998-02-02       U     LE3       T    1    1  at_home
## 4 4     GP Female 1997-12-20       U     GT3       T    4    2   health
## 5 5     GP Female 1998-10-04       U     GT3       T    3    3    other
## 6 6     GP   Male 1999-06-16       U     LE3       T    4    3 services
##       Fjob     reason guardian traveltime studytime failures schoolsup
## 1  teacher     course   mother          2         2        0       yes
## 2    other     course   father          1         2        0        no
## 3    other      other   mother          1         2        3       yes
## 4 services       home   mother          1         3        0        no
## 5    other       home   father          1         2        0        no
## 6    other reputation   mother          1         2        0        no
```

…

Missing and special values

Missing values

- Data are missing for many reasons
  - The reason determine how we should deal with it
- Sometimes associated with variable/outcome of interest
- In R, represented as NA
- May appear in other forms
  - #N/A (Excel)
  - Single dot (SPSS, SAS)
  - Empty string

Special values

- Inf - "Infinite value" (indicative of outliers?)
  - 1/0
  - 1/0 + 1/0
  - 33333^33333
- NaN - "Not a number" (rethink a variable?)
  - 0/0
  - 1/0 - 1/0

Finding missing values

```
# Create small dataset
df <- data.frame(A = c(1, NA, 8, NA),
                 B = c(3, NA, 88, 23),
                 C = c(2, 45, 3, 1))
```

```
# Check for NAs
is.na(df)
```

```
##           A     B     C
## [1,] FALSE FALSE FALSE
## [2,]  TRUE  TRUE FALSE
## [3,] FALSE FALSE FALSE
## [4,]  TRUE FALSE FALSE
```

```
# Are there any NAs?
any(is.na(df))
```

```
## [1] TRUE
```

```
# Count number of NAs
sum(is.na(df))
```

```
## [1] 3
```

```
# Count number of NAs in each var.
colSums(is.na(df))
```

```
## A B C
## 2 1 0
```

```
# Use summary() to find NAs
summary(df)
```

```
##        A                B                C
##  Min.   :1.00   Min.   : 3.0   Min.   : 1.00
##  1st Qu.:2.75   1st Qu.:13.0   1st Qu.: 1.75
##  Median :4.50   Median :23.0   Median : 2.50
##  Mean   :4.50   Mean   :38.0   Mean   :12.75
##  3rd Qu.:6.25   3rd Qu.:55.5   3rd Qu.:13.50
##  Max.   :8.00   Max.   :88.0   Max.   :45.00
##  NA's   :2      NA's   :1
```

Dealing with missing values

```
# Find rows with no missing values
complete.cases(df)
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```
# Subset data, keeping only
complete cases
df[complete.cases(df), ]
```

```
##   A  B C
## 1 1  3 2
## 3 8 88 3
```

```
# Another way to remove rows with
NAs
na.omit(df)
```

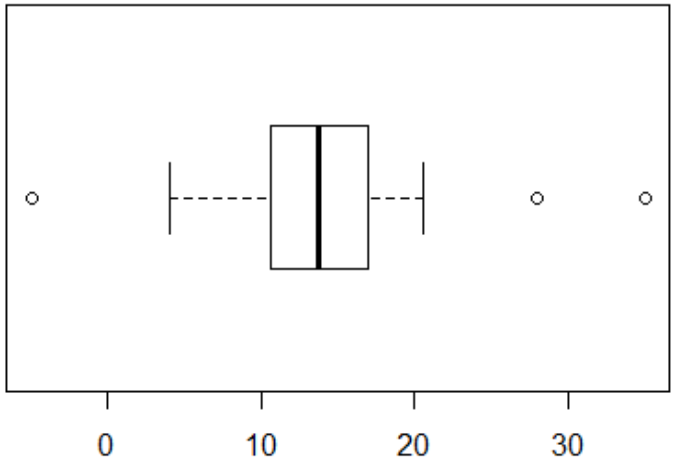
```
##   A  B C
## 1 1  3 2
## 3 8 88 3
```

Outliers and obvious errors

Outliers

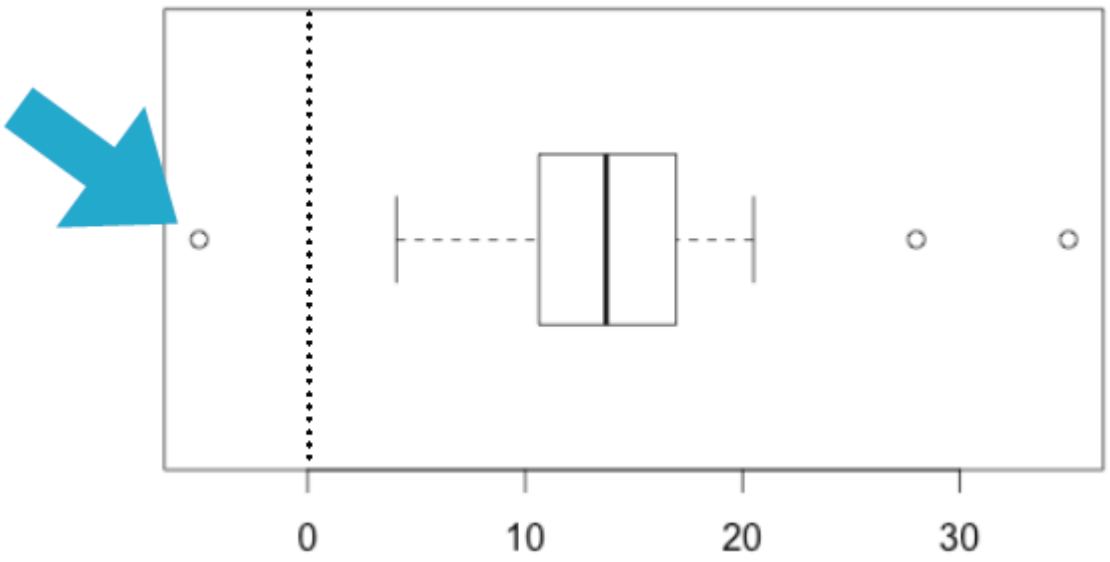
- Extreme values distant from other values
- Several causes
  - Valid measurements
  - Variability in measurement
  - Experimental error
  - Data entry error
- May be discarded or retained depending on cause

```
# Simulate some data
set.seed(10)
x <- c(rnorm(30, mean = 15, sd = 5), -5, 28, 35)
# View a boxplot
boxplot(x, horizontal = TRUE)
```



Obvious errors

What if these values are supposed to represent ages?



Obvious errors

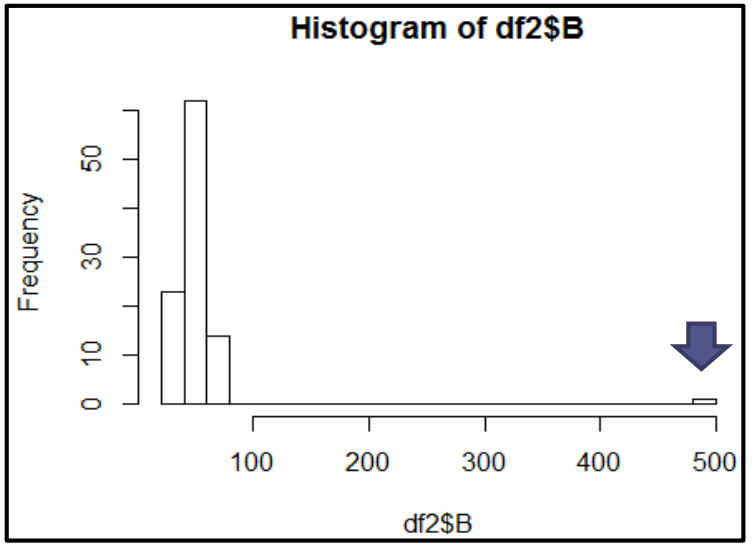
- May appear in many forms
  - Values so extreme they can't be plausible (e.g. person aged 243)
  - Values that don't make sense (e.g. negative age)
- Several causes
  - Measurement error
  - Data entry error
  - Special code for missing data (e.g. -1 means missing)
- Should generally be removed or replaced

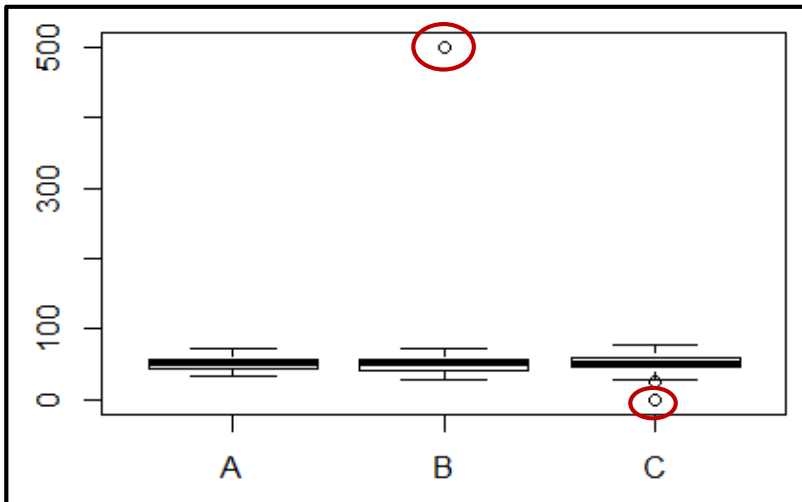
Finding outliers and errors

```
# Create another small dataset
df2 <- data.frame(A = rnorm(100, 50, 10),
                  B = c(rnorm(99, 50, 10), 500),
                  C = c(rnorm(99, 50, 10), -1))
# View a summary
summary(df2)
```

```
##        A                B                C
##  Min.   :31.46   Min.   : 26.79   Min.   :-1.00
##  1st Qu.:42.21   1st Qu.: 41.35   1st Qu.:45.29
##  Median :50.20   Median : 50.67   Median :51.06
##  Mean   :49.70   Mean   : 53.62   Mean   :50.88
##  3rd Qu.:57.12   3rd Qu.: 56.57   3rd Qu.:58.13
##  Max.   :72.21   Max.   :500.00   Max.   :76.44
```

```
# View a histogram
hist(df2$B, breaks = 20)

# View a boxplot
boxplot(df2)
```





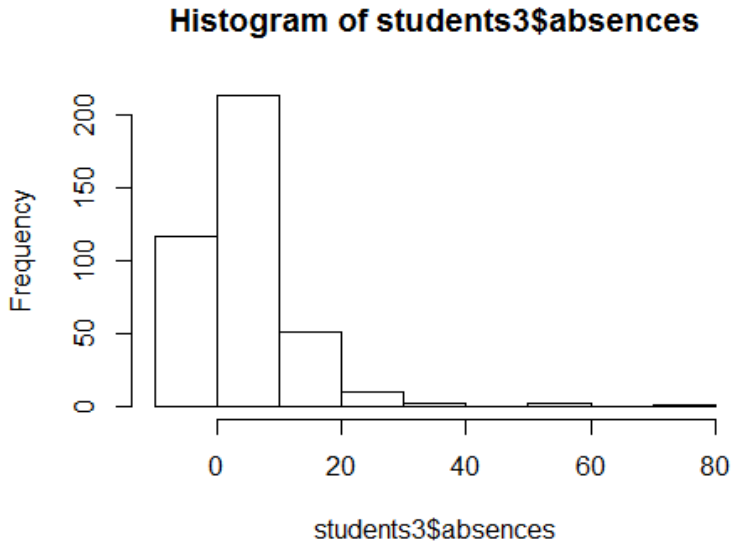
Exercise

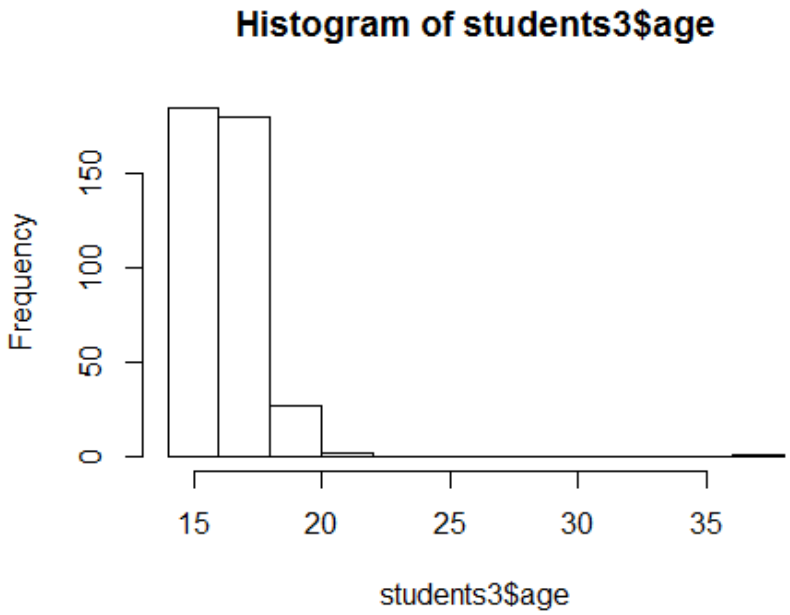
When dealing with strange values in your data, you often must decide whether they are just extreme or actually erroneous. Extreme values show up all over the place, but you, the data analyst, must figure out when they are plausible and when they are not.

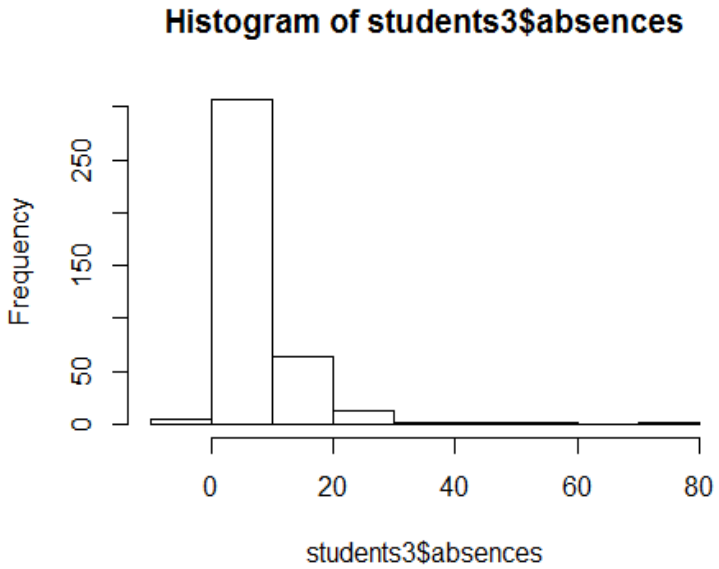
In the dataset students3, two variables appear to have suspicious values: age and absences. Let's explore these values further.

1. Call summary() on the full students3 dataset to expose the concerning values of age and absences.
2. View a histogram (using hist()) of the age variable.
3. View a histogram of the absences variable.
4. View another histogram of absences, but force values of zero to be bucketed to the right of zero on the x-axis with right = FALSE (see ?hist for more info).
5. View a boxplot() of the age variable from students3
6. View a boxplot() of the absences variable from students3

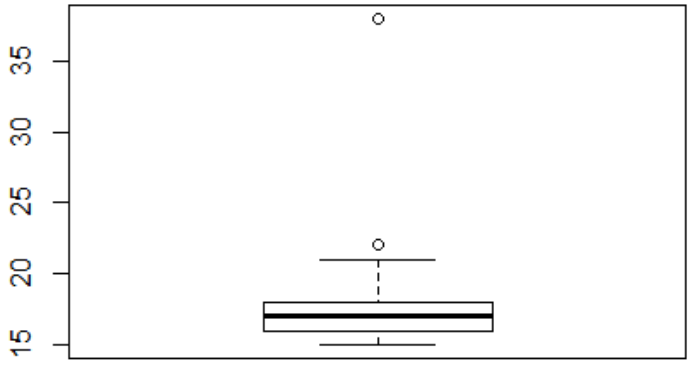
```
students3 <- read.csv(file = 'students3.csv', header = TRUE,
stringsAsFactors = F)
# Look at a summary() of students3

# View a histogram of the age variable

# View a histogram of the absences variable

# View a histogram of absences,
but force zeros to be bucketed to the right of zero
```

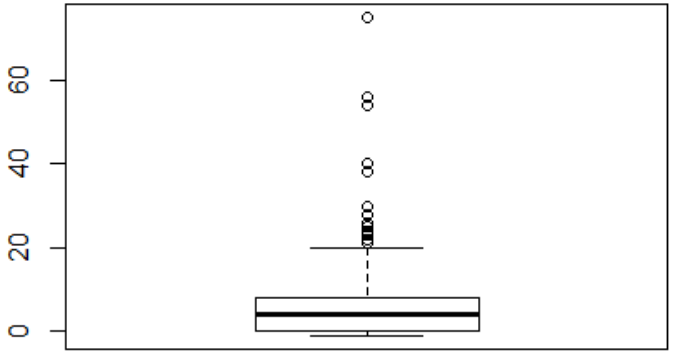






```
# View a boxplot of age

# View a boxplot of absences
```





In this situation, we are concerned about three things:

- Since this dataset is about students and the only student above the age of 22 is 38 years old, we must wonder whether this is an error in the data or just an older student (perhaps returning to school after working for several years)
- There are four values of -1 for the absences variable, which is either a mistake or an intentional coding meant to say, for example, "this value is missing"
- There are several extreme values of absences in the positive direction, with a maximum value of 75 (which is over 18 times the median value of 4)

References

- Practical Data Science with R, by Nina Zumel and John Mount
- R을 이용한 데이터 분석 실무, 서민구, 길벗
- [DBGUIDE 연재] ggplot2를 이용한 R 시각화
  - http://freesearch.pe.kr/archives/3134