KENT STATE UNIVERSITY

MIS 64036: Business Analytics

Lecture IV

Rouzbeh Razavi, PhD

Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

Agenda

- **Subsetting and Filtering Data**
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

Subsetting and Filtering Data

- In almost every data modeling projects, you need to select a subset of data for exploration, preprocessing, modelling or even scoring.

- The next few slides gives examples of data selection and filtering when dealing with R dataframes.

Selecting (Keeping) Variables

```
# select variables v1, v2, v3
myvars <- c("v1", "v2", "v3")
newdata <- mydata[myvars]

# select 1st and 5th thru 10th variables
newdata <- mydata[c(1,5:10)]
#Or even more explicit
newdata <- mydata[,c(1,5:10)]
```

Excluding (DROPPING) Variables

```
# exclude variables v1, v2, v3
myvars <- c("v1", "v2", "v3")
newdata <- mydata[!myvars]

# exclude 3rd and 5th variable
newdata <- mydata[c(-3,-5)]

# delete variables v3 and v5
mydata$v3 <- NULL
mydata$v5 <-NULL
```

Selecting Observations

```
# first 5 observations
newdata <- mydata[1:5,]

# based on variable values
newdata <- mydata[ which(mydata$gender=='F'
& mydata$age > 65), ]
```

You can even drop the which() command.

```
newdata <- mydata[ (mydata$gender=='F'
& mydata$age > 65), ]
```

Selection using the Subset Function

The subset( ) function is the easiest way to select variables and observations. In the following example, we select all rows that have a value of age greater than or equal to 20 or age less then 10. We keep the ID and Weight columns.

```
newdata <- subset(mydata, age >= 20 | age < 10,
select=c(ID, Weight))
```

Practice here:

https://campus.datacamp.com/courses/free-introduction-to-r/chapter-5-data-frames?ex=6

Random Samples

Use the sample( ) function to take a random sample of size n from a dataset.

```
# take a random sample of size 50 from a dataset mydata
# sample without replacement
mysample <- mydata[sample(1:nrow(mydata), 50,
        replace=FALSE),]
```

“dplyr” Package

dplyr is a powerful R-package to transform and summarize tabular data with rows and columns.

The package contains a set of functions (or “verbs”) that perform common data manipulation operations such as filtering for rows, selecting specific columns, re-ordering rows, adding new columns and summarizing data.

Compared to base functions in R, the functions in dplyr are easier to work with, are more consistent in the syntax

“dplyr” Package : examples

library(dplyr)

- sleepData <- select(mydata, age, ID)
- filter(msleep, age>= 16 & age <= 45 | age==33)
- sample_frac(mydata,0.1) #10% random samples
- x1 = distinct(mydata) #remove dublicated rows

#Selecting Variables contain 'I' in their names

- select(mydata, contains("I"))
- arrange(mydata, age, ID) Sort Data by Multiple Variables

More examples:

http://www.listendata.com/2016/08/dplyr-tutorial.html

Adding New Column/Rows

You can use the rbind() function to combine two dataframes by rows (or simply add new rows). Two dataframe should have identical columns.

New_dataframe=rbind(dataframe1,dataframe2)

Same can be done for columns. Two dataframe should have same number of rows.

New_dataframe=cbind(dataframe1,dataframe2)

Adding New Column/Rows : Example

```
# make two vectors and combine them as columns in a data.frame
sport <- c("Hockey", "Baseball", "Football")
league <- c("NHL", "MLB", "NFL")
trophy <- c("Stanley Cup", "Commissioner's Trophy", "Vince Lombardi Trophy")
trophies1 <- cbind(sport, league, trophy)
trophies1
# make another data.frame using data.frame()
trophies2 <- data.frame(sport=c("Basketball", "Golf"),   league=c("NBA", "PGA"),
                        trophy=c("Larry Brien Championship Trophy", "Wanamaker Trophy"),
                           stringsAsFactors=FALSE)
trophies2
# combine them into one data.frame with rbind
trophies <- rbind(trophies1, trophies2)
trophies
```

Adding New Column/Rows : Example

```
> # make two vectors and combine them as columns in a data.frame
> sport <- c("Hockey", "Baseball", "Football")
> league <- c("NHL", "MLB", "NFL")
> trophy <- c("Stanley Cup", "Commissioner's Trophy", "Vince Lombardi Trophy")
> trophies1 <- cbind(sport, league, trophy)
> trophies1
     sport      league trophy
[1,] "Hockey"   "NHL"  "Stanley Cup"
[2,] "Baseball" "MLB"  "Commissioner's Trophy"
[3,] "Football" "NFL"  "Vince Lombardi Trophy"
> # make another data.frame using data.frame()
> trophies2 <- data.frame(sport=c("Basketball", "Golf"),   league=c("NBA", "PGA"),
+                         trophy=c("Larry Brien Championship Trophy", "Wanamaker Trophy"),
tringsAsFactors=FALSE)
> trophies2
       sport league                          trophy
1 Basketball    NBA Larry Brien Championship Trophy
2       Golf    PGA                Wanamaker Trophy
> # combine them into one data.frame with rbind
> trophies <- rbind(trophies1, trophies2)
> trophies
       sport league                          trophy
1     Hockey    NHL                     Stanley Cup
2   Baseball    MLB           Commissioner's Trophy
3   Football    NFL           Vince Lombardi Trophy
4 Basketball    NBA Larry Brien Championship Trophy
5       Golf    PGA                Wanamaker Trophy
```

“dplyr” Package : examples

library(dplyr)

- sleepData <- select(mydata, age, ID)
- filter(msleep, age>= 16 & age <= 45 | age==33)
- sample_frac(mydata,0.1) #10% random samples
- x1 = distinct(mydata) #remove dublicated rows

#Selecting Variables contain 'I' in their names

- select(mydata, contains("I"))
- arrange(mydata, age, ID) Sort Data by Multiple Variables

More examples:

http://www.listendata.com/2016/08/dplyr-tutorial.html

Aggregate Functions

- Aggregate functions are functions that take a **collection of values** as input and return a **single** value.
- Behavior of Aggregate Functions:
    - Operates - on a **single** column (variable)
    - Return - a **single** value.
- Examples:
    - Sum , Mean (Numeric Values Only)
    - Min, Max, Count (Naluesumeric and Non-Numeric values i.e. Aplphabetical sorting for Max and Min)

Aggregate Row Functions

Aggregate Row functions give the user the ability to answer business questions such as:

- What is the average salary of an employee in the company?
- What were the total salaries for a particular year?
- What are the maximum and minimum salaries in the Computer's Department?

Aggregate Row Functions

- Aggregate functions perform a variety of actions such as counting all the rows in a table, averaging a column's data, and summing numeric data.
- Aggregates can also search a table to find the highest "Max" or lowest "Max" values in a column.
- All group functions ignore Null (NA) values except Count()

Group By Function

- The group by statement is especially useful for applying aggregating functions.
- In "dplyr" library, group_by()function takes an existing table or dataframe and converts it into a grouped table where operations are performed "by group".
- The summarise() function can be then applied to the grouped dataframe with appropriate function e.g. mean, sum, max, min, n (for count).

Group By Function: Example I

```
> library(dplyr)
> 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
> Results<-summarise(group_by(mtcars, cyl), mean(hp))
> Results
# A tibble: 3 x 2
    cyl `mean(hp)`
  <dbl>      <dbl>
1     4   82.63636
2     6  122.28571
3     8  209.21429
```

Group By Function: Example II

```
> Results<-summarise(group_by(mtcars, cyl), max(hp))
> Results
# A tibble: 3 x 2
    cyl `max(hp)`
  <dbl>     <dbl>
1     4       113
2     6       175
3     8       335
> Results<-summarise(group_by(mtcars, cyl),  n())
> Results
# A tibble: 3 x 2
    cyl `n()`
  <dbl> <int>
1     4    11
2     6     7
3     8    14
```

Count()

Ugly! Should use Column name see next slide.

Group By Function: Example III

```
> Results<-summarise(group_by(mtcars, cyl),  n(),min(hp),mean(mpg))
> Results
# A tibble: 3 x 4
    cyl `n()` `min(hp)` `mean(mpg)`
  <dbl> <int>     <dbl>       <dbl>
1     4    11        52    26.66364
2     6     7       105    19.74286
3     8    14       150    15.10000
```

```
> Results<-summarise(group_by(mtcars, cyl),  count=n(),Minimum_HP=min(hp),Average_MPG=mean(mpg))
> Results
# A tibble: 3 x 4
    cyl count Minimum_HP Average_MPG
  <dbl> <int>      <dbl>       <dbl>
1     4    11         52    26.66364
2     6     7        105    19.74286
3     8    14        150    15.10000
```

Group By Function: Example III

```
> Results<-summarise(group_by(mtcars, cyl),  count=n(),Minimum_HP=min(hp),Average_MPG=mean(mpg))
> as.data.frame(Results)
  cyl count Minimum_HP Average_MPG
1   4    11         52    26.66364
2   6     7        105    19.74286
3   8    14        150    15.10000
```

Convert the results back to dataframe

Piping

- The pipe operator in R, represented by %>% can be used to chain code together.

- It is very useful when you are performing several operations on data, and don't want to save the output at each intermediate step.

- For example, let's say we want to remove all the data corresponding to cyl= 6, group the data by gear, and then find the mean of the MPG for each gear value. The conventional way to write the code for this would be:

Piping II

Unnecessary variables.

```
> filteredData <- filter(mtcars, cyl != 6)
> groupedData <- group_by(filteredData, gear)
> summarise(groupedData, Average_mpg=mean(mpg, na.rm = TRUE))
# A tibble: 3 x 2
   gear Average_mpg
  <dbl>       <dbl>
1     3    15.54615
2     4    26.92500
3     5    21.80000
```

- With piping, the above code can be rewritten as:

```
> mtcars %>% filter(cyl != 6) %>% group_by(gear) %>% summarise(Average_mpg=mean(mpg, na.rm = TRUE))
# A tibble: 3 x 2
   gear Average_mpg
  <dbl>       <dbl>
1     3    15.54615
2     4    26.92500
3     5    21.80000
```

Agenda

- Subsetting and Filtering Data
- **Understanding Data**
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

Get to Know Your Data First

- Before any attempt to model data or even deciding a data pre-processing approach, try to familiarize your self with your data. This includes:

  - Understanding the variables in your dataset
  - Understanding their data types
  - Developing a high level understanding of the ranges, frequencies and distribution of data variables
  - Understanding correlations amongst variables

- Deploy appropriate summary functions as well as visualization.

Univariate: summary() Function

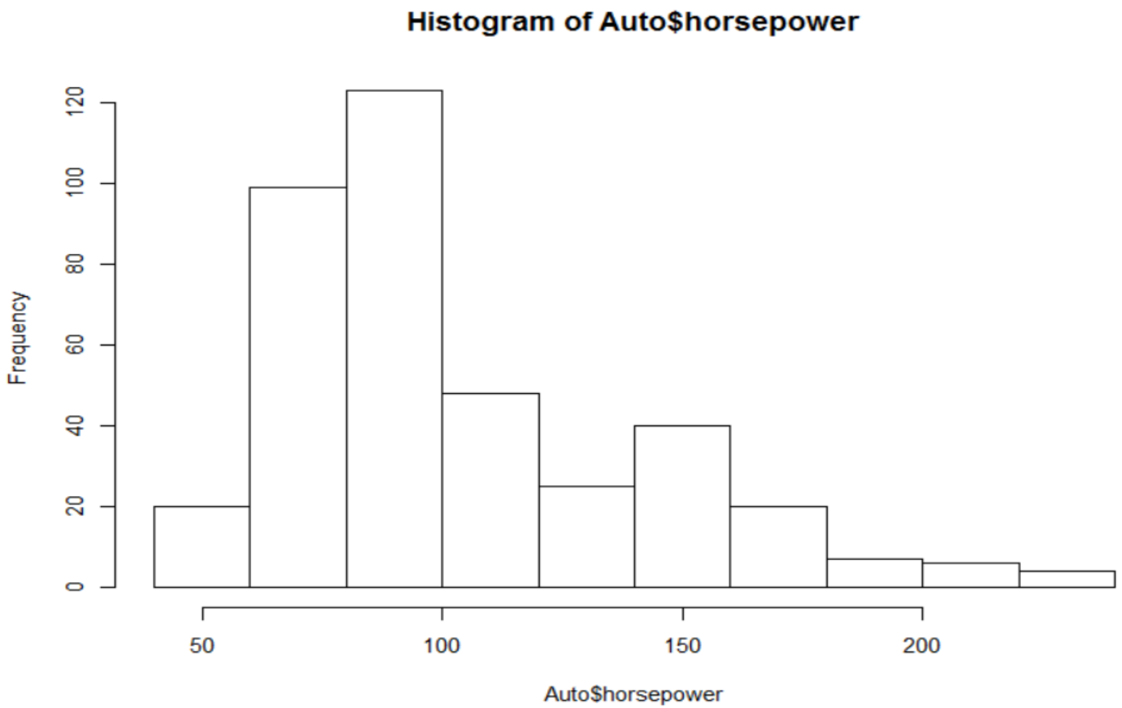
```
> library(ISLR)
> summary(Auto)
      mpg          cylinders      displacement     horsepower   
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0  
 1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0  
 Median :22.75   Median :4.000   Median :151.0   Median : 93.5  
 Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5  
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0  
 Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0  

     weight      acceleration        year           origin     
 Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000  
 1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000  
 Median :2804   Median :15.50   Median :76.00   Median :1.000  
 Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577  
 3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000  
 Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000  

                 name    
 amc matador       :  5  
 ford pinto        :  5  
 toyota corolla    :  5  
 amc gremlin       :  4  
 amc hornet        :  4  
 chevrolet chevette:  4  
 (Other)           :365  
> 
```

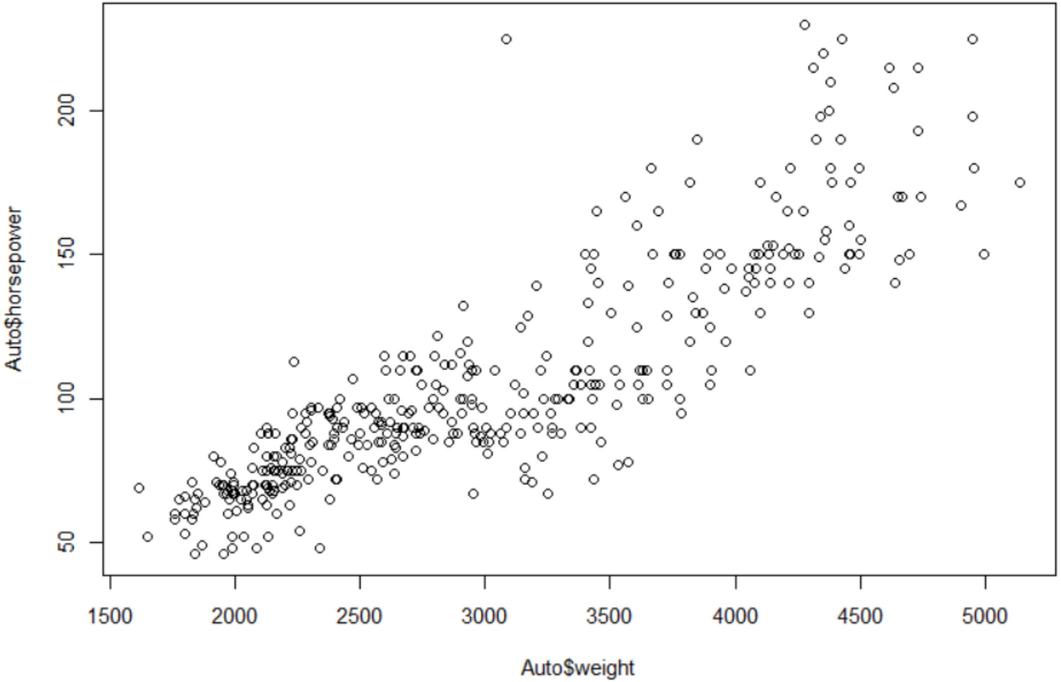
Univariate: Histogram plot

```
> library(ISLR)
> hist(Auto$horsepower)
```



Bivariate (Numeric-Numeric): plot

```
> library(ISLR)
> plot(Auto$weight, Auto$horsepower)
```



Bivariate (Numeric-Categorical): describeBy

```
> library(ISLR)
> library(psych) #install first
> describeBy(Wage$wage,Wage$education)

 Descriptive statistics by group
group: 1. < HS Grad
   vars   n mean    sd median trimmed   mad   min    max  range ske
X1    1 268 84.1 21.58  81.28   83.21 18.77 20.93 152.22 131.28  0.
------------------------------------------------------------------
group: 2. HS Grad
   vars   n  mean    sd median trimmed   mad   min    max  range sk
X1    1 971 95.78 28.57  94.07   94.18 23.37 23.27 318.34 295.07 1.
------------------------------------------------------------------
group: 3. Some College
   vars   n   mean    sd median trimmed   mad   min    max  range s
X1    1 650 107.76 32.47 104.92  105.93 23.25 20.09 314.33 294.24 1
------------------------------------------------------------------
group: 4. College Grad
   vars   n   mean    sd median trimmed   mad   min    max  range s
X1    1 685 124.43 41.19 118.88  121.14 33.94 32.37 281.75 249.38 1
------------------------------------------------------------------
group: 5. Advanced Degree
   vars   n   mean   sd median trimmed   mad   min    max  range sk
X1    1 426 150.92 53.9 141.78  144.08 36.67 38.61 318.34 279.74  1
```

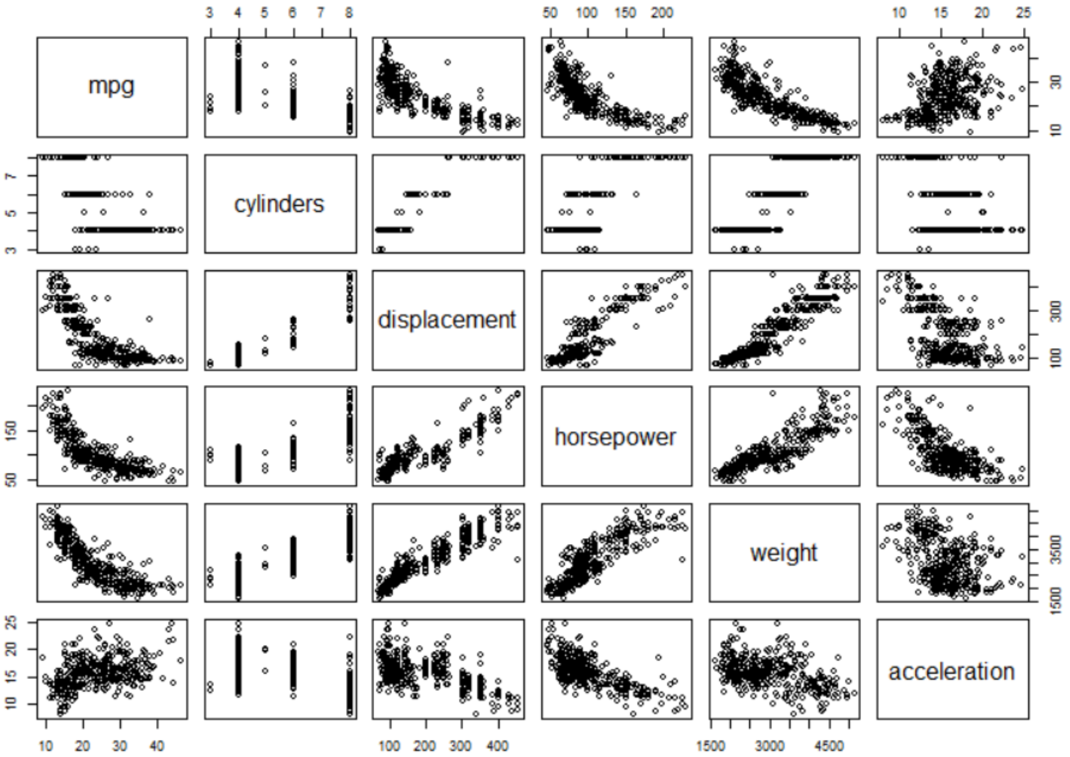
Bivariate (Categorical-Categorical): table

```
> library(ISLR)
> table(Wage$education,Wage$maritl)

                     1. Never Married 2. Married 3. Widowed 4. Divorced 5. Separated
  1. < HS Grad                     62        174          2          16           14
  2. HS Grad                      219        651          8          73           20
  3. Some College                 164        421          2          52           11
  4. College Grad                 143        487          5          41            9
  5. Advanced Degree               60        341          2          22            1
```

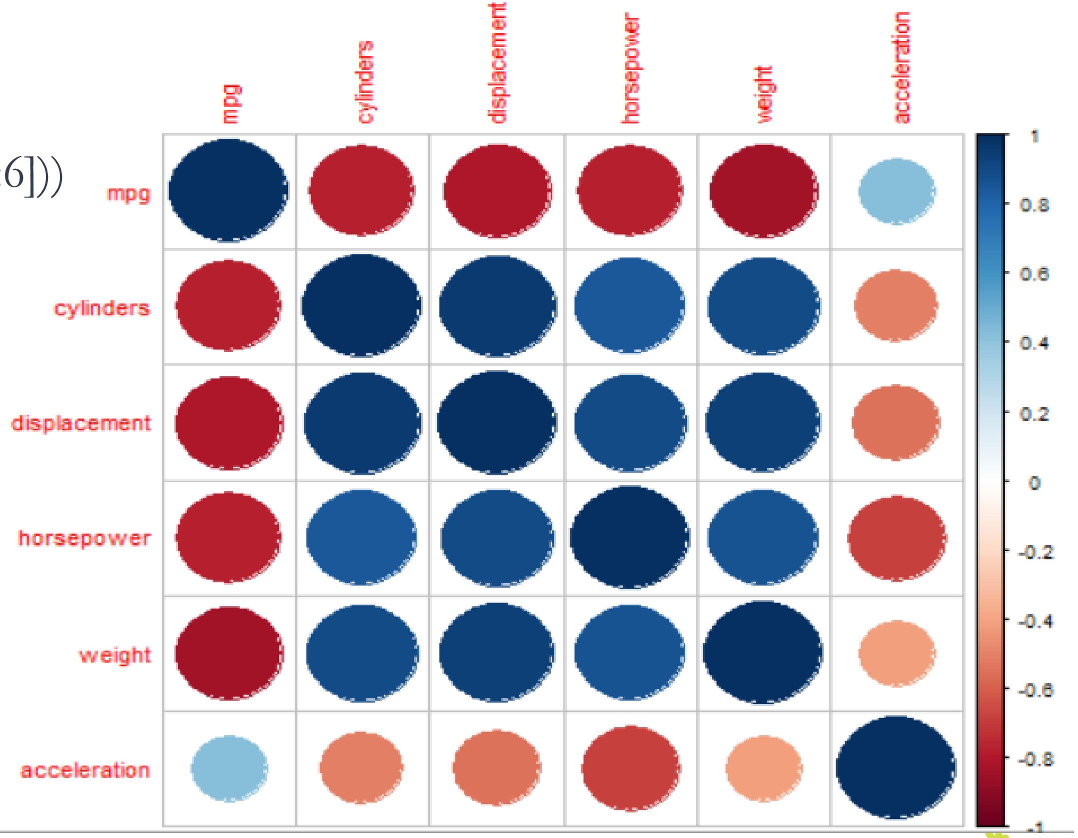
Multivariate: pairs() Function.

> pairs(Auto[,1:6]) #plot pair plots of columns 1-6



Multivariate: corrplot() Function.

```
library(ISLR)
library(corrplot)
corrplot(cor(Auto[,1:6]))
```



Agenda

- Subsetting and Filtering Data
- Understanding Data
- **Data Preprocessing and Quality**
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

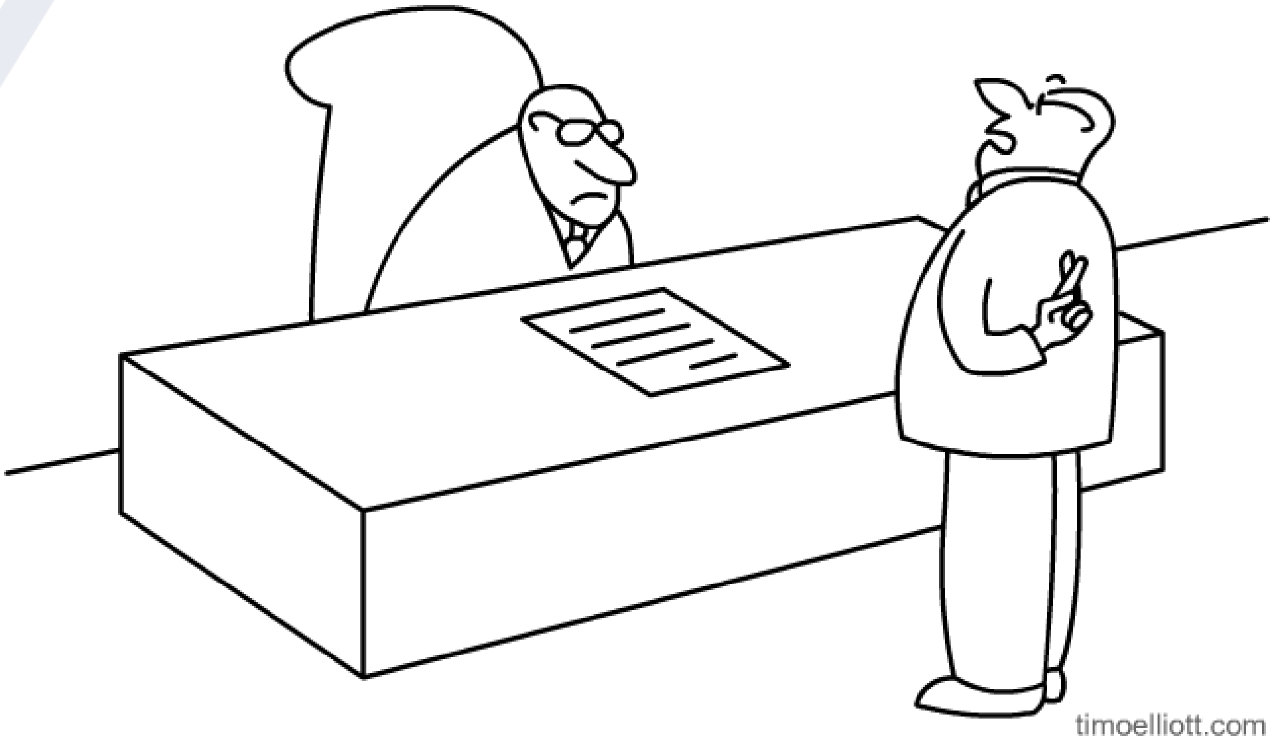
Why Data Preprocessing?

- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - noisy: containing errors or outliers
  - inconsistent: containing discrepancies in codes or names
- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
  - Data warehouse needs consistent integration of quality data

Multi-Dimensional Measure of Data Quality

- **A well-accepted multidimensional view:**
  - Accuracy
  - Completeness
  - Consistency
  - Timeliness
  - Interpretability
  - Accessibility

Data Quality



"Yes sir, you can absolutely trust those numbers"

Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- **Data Quality: Missing Values**
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

Data Cleaning

- **Data requires cleaning tasks because in most applications it is**
  - Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data e.g., occupation=“”
  - Noisy: containing errors or outliers e.g., Salary=“-10”, Age=“222”
  - Inconsistent: containing discrepancies in codes or names e.g., Age=“42” Birthday=“03/07/1997” or Was rating “1,2,3”, now rating “A, B, C”

Missing Values !

- **What is certain in life?**
  - Death
  - Taxes
- **What is certain in modelling?**
  - Measurement errors
  - Missing data

Missing Values

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing at Random (MR) versus Missing Not at Random (MNR)

Why Missing Not at Random (MNAR)?

- There might be a reason to believe that responders differ from non-responders.

- Some examples:
  - Income - some people may not reveal their salaries
  - Blood pressure - the blood pressure is measured less frequently for patients with lower blood pressures

- MNAR values can contain high predictive power but field expertise is required to detect them.

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing or small percentage of missing values.
- Fill in the missing value manually: tedious + infeasible?
- For categorical variables: use a global constant to fill in the missing value: e.g., “unknown”, a new class?!
- For numerical variables: use the attribute mean to fill in the missing value.
- For numerical or categorical variables : use the most probable value to fill in the missing value.

To Drop or Not to Drop?

- If the number of missing values is small, it may be safe to simply drop rows containing missing values.
- The function is.na() frame returns TRUE for every element of the input that NA and FALSE otherwise.
- The function rowMeans()and colMeans() calculate the average values across rows and columns. We can use a combination to calculate the percentage of missing values per row or per column.

a<- c(1,NA,NA,4)

b<- c(NA,2,8,7)

c<-c(NA,1,2,9)

x<- data.frame(a,b,c)

```
> a<- c(1,NA,NA,4)
> b<- c(NA,2,8,7)
> c<-c(NA,1,2,9)
> x<- data.frame(a,b,c)
> x
   a  b  c
1  1 NA NA
2 NA  2  1
3 NA  8  2
4  4  7  9
> rowMeans(is.na(x))
[1] 0.6666667 0.3333333 0.3333333 0.00
> colMeans(is.na(x))
   a    b    c
0.50 0.25 0.25
```

Dropping Rows/Columns

- If a row has more than a certain ratio of missing values (e.g. 40%), and it is believed that the missing values are at random, it is wise to drop the row.
- Same is true for columns. However, take care: a few missing values in this variable and a few missing values in that variable can quickly add up to a lot of incomplete data.

```
> x
   a  b  c
1  1 NA NA
2 NA  2  1
3 NA  8  2
4  4  7  9
> x_rowfiltered<-x[rowMeans(is.na(x))<0.4,]
> x_rowfiltered
   a b c
2 NA 2 1
3 NA 8 2
4  4 7 9
> x_colfiltered<-x[,colMeans(is.na(x))<0.4,]
> x_colfiltered
   b  c
1 NA NA
2  2  1
3  8  2
4  7  9
>
```

First row is dropped

Column “a” is dropped

Dropping/Imputing Missing Values

```
original_data<-airquality

colMeans(is.na(original_data))

missing_dropped<-original_data[complete.cases(original_data),]

colMeans(is.na(missing_dropped))

##############################

missing_imputed<-original_data

missing_imputed[is.na(original_data$Ozone),'Ozone'] <-
mean(original_data$Ozone,na.rm=TRUE)

colMeans(is.na(missing_dropped))
```

Dropping/Imputing Missing Values

```
> original_data<-airquality
> colMeans(is.na(original_data))
     Ozone    Solar.R       Wind       Temp      Month        Day 
0.24183007 0.04575163 0.00000000 0.00000000 0.00000000 0.00000000 
> missing_dropped<-original_data[complete.cases(original_data),]
> colMeans(is.na(missing_dropped))
  Ozone Solar.R    Wind    Temp   Month     Day 
      0       0       0       0       0       0 
> ###########################
> missing_imputed<-original_data
> missing_imputed[is.na(original_data$Ozone),'Ozone'] <-mean(original_data$Ozone,na.rm=TRUE)
> colMeans(is.na(missing_dropped))
  Ozone Solar.R    Wind    Temp   Month     Day 
      0       0       0       0       0       0 
```

Imputing Missing Values Using Models

- You can also build models to determine the most probable value to fill in the missing value.

- This will be the most accurate approach for MR missing values. However, it may requires more complex computation.

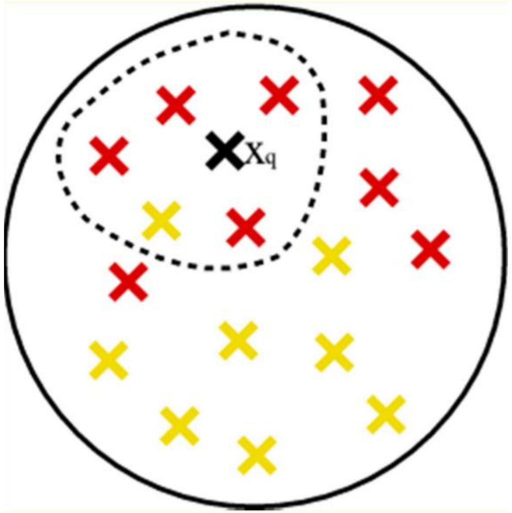
- The following modeling approaches are common for missing vale imputation:
  - k-Nearest Neighbors (k-NN)
  - Linear Regression
  - Decision Trees

k-Nearest Neighbors

- Finding the k neighbors nearest to the missing point and fill in the missing value with the most frequent value or the average value of neighbors.

- Finding neighbors in a large dataset may be slow

- Defining the right distance metric can be tricky especially when you have a mix of numerical and categorical variables.



k-Nearest Neighbors Imputation in R

```
library(VIM)
colMeans(is.na(airquality))
#Just imputing the 'Ozone' variable
missing_imputed_knn<-kNN(airquality,variable=c("Ozone"),k=4)
colMeans(is.na(missing_imputed_knn))
#Imputing all variables with missing values
missing_imputed_knn<-kNN(airquality,k=4)
colMeans(is.na(missing_imputed_knn))
```

k-Nearest Neighbors Imputation in R

```
> library(VIM)
> colMeans(is.na(airquality))
     Ozone    Solar.R       Wind       Temp      Month        Day
0.24183007 0.04575163 0.00000000 0.00000000 0.00000000 0.00000000
> #Just imputing the 'Ozone' variable
> missing_imputed_knn<-kNN(airquality,variable=c("Ozone"),k=4)
```

```
> colMeans(is.na(missing_imputed_knn))
     Ozone    Solar.R       Wind       Temp      Month        Day  Ozone_imp
0.00000000 0.04575163 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
> #Imputing all variables with missing values
> missing_imputed_knn<-kNN(airquality,k=4)
```

```
> colMeans(is.na(missing_imputed_knn))
      Ozone     Solar.R        Wind        Temp       Month         Day   Ozone_imp Solar.R_imp
          0           0           0           0           0           0           0           0
   Wind_imp    Temp_imp   Month_imp     Day_imp
          0           0           0           0
```

k-Nearest Neighbors Imputation in R

```
> head(missing_imputed_knn)
  Ozone Solar.R Wind Temp Month Day Ozone_imp Solar.R_imp Wind_imp Temp_imp Month_imp Day_imp
1    41     190  7.4   67     5   1     FALSE       FALSE    FALSE    FALSE     FALSE   FALSE
2    36     118  8.0   72     5   2     FALSE       FALSE    FALSE    FALSE     FALSE   FALSE
3    12     149 12.6   74     5   3     FALSE       FALSE    FALSE    FALSE     FALSE   FALSE
4    18     313 11.5   62     5   4     FALSE       FALSE    FALSE    FALSE     FALSE   FALSE
5    20     199 14.3   56     5   5      TRUE        TRUE    FALSE    FALSE     FALSE   FALSE
6    28     224 14.9   66     5   6     FALSE        TRUE    FALSE    FALSE     FALSE   FALSE
```

Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- **Data Quality: Outliers**
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

Outliers

- Outliers are values thought to be out of range.

  "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism"

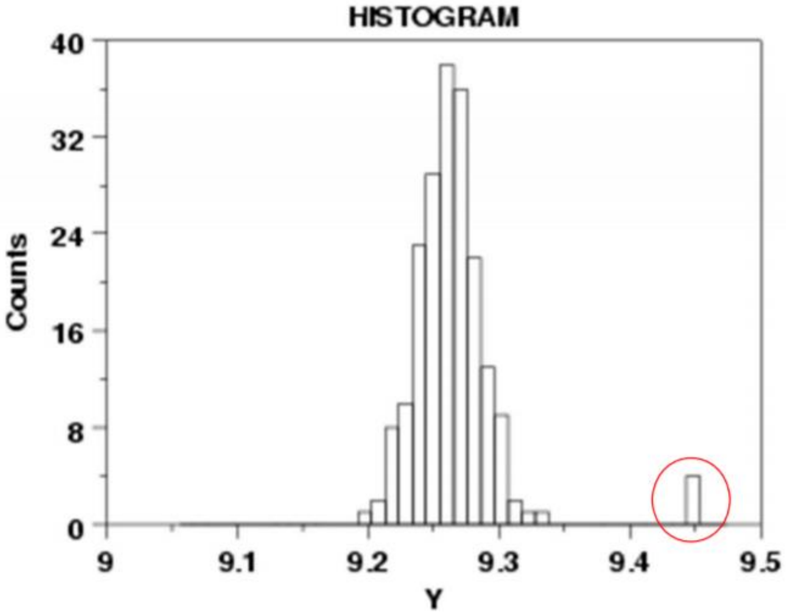
- Can be detected by standardizing observations and label the standardized values outside a predetermined bound as outliers

- Approaches:
  - do nothing i.e. let modeling algorithm handle the problem
  - enforce upper and lower bounds

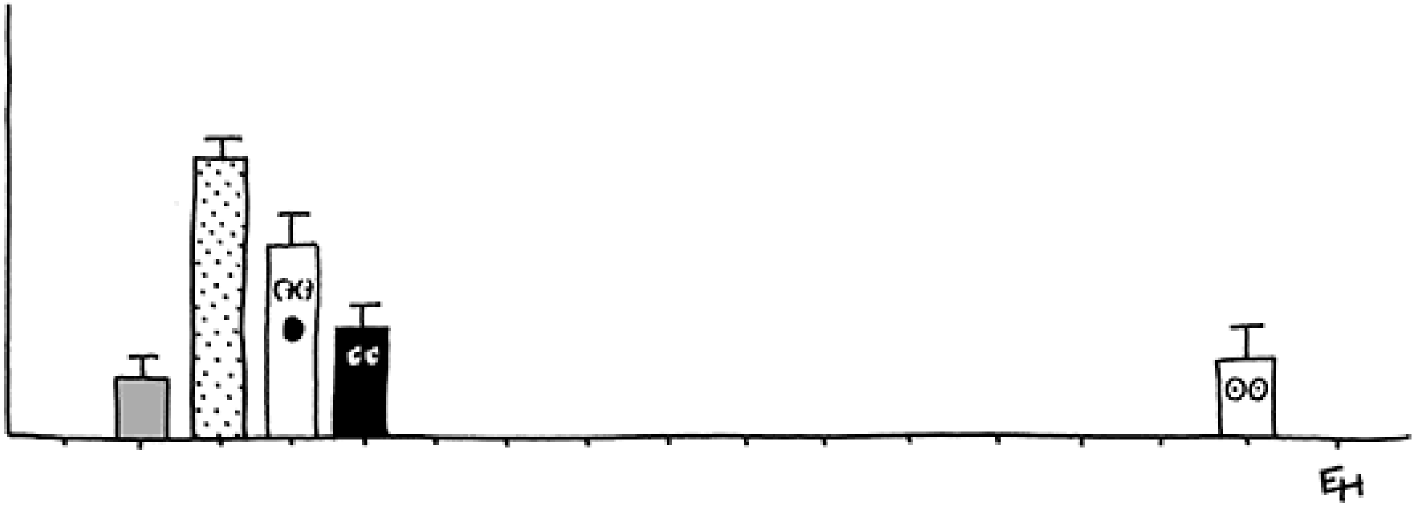
Outlier Detection

- Univariate Approach: Compute mean and std. deviation. For k=2 or 3, x is an outlier if outside limits (normal distribution assumed)

$$(\bar{x} - ks, \bar{x} + ks)$$



Outlier Detection



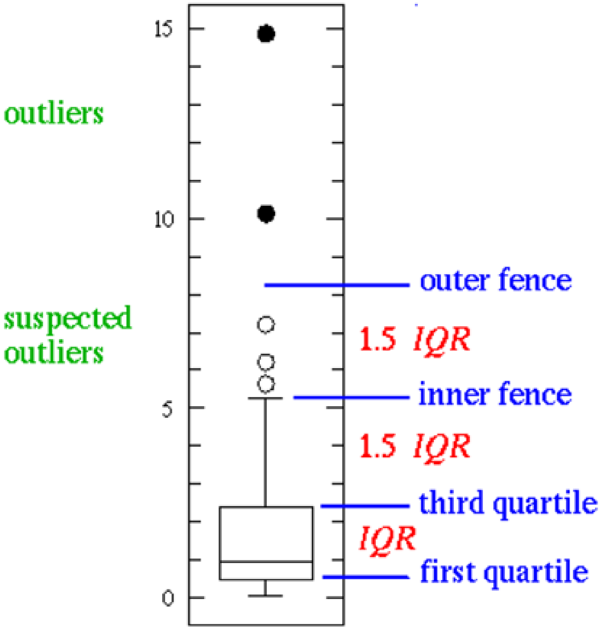
That's Jake...he's always been something of an outlier.

Outliers

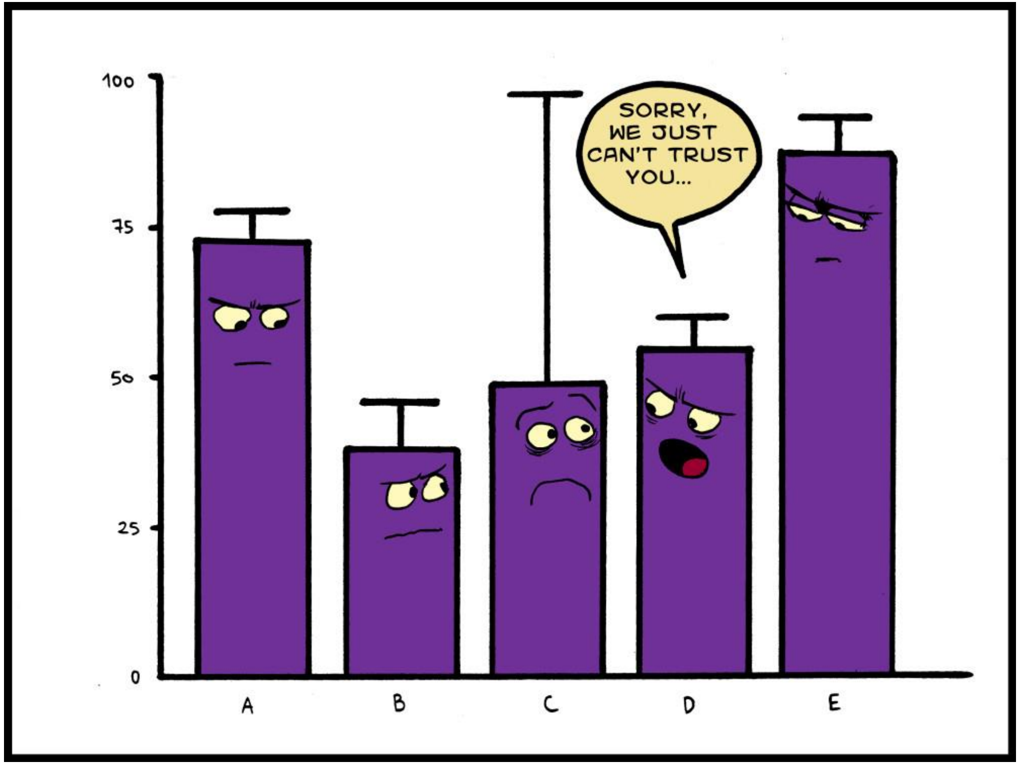
- Boxplot:

An observation is an extreme outlier if (Q1-3IQR, Q3+3IQR), where IQR=Q3-Q1

and declared a mild outlier if it lies outside of the interval (Q1-1.5IQR, Q3+1.5 IQR)



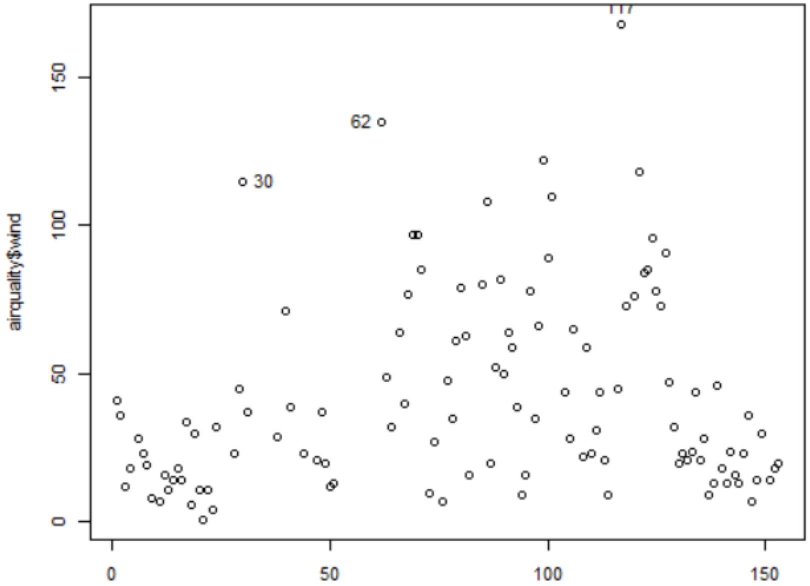
Outliers



Outliers : Interactive Demo

Multivariate

- plotting
- identify() function

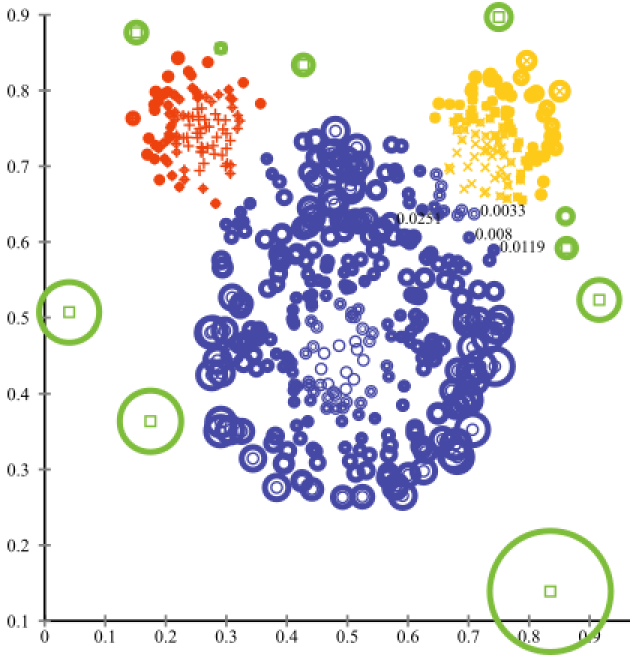


> plot(airquality$Ozone,airquality$wind)
> identify(airquality$Ozone,airquality$wind)

Outliers

Multivariate

- Clustering
- Very small clusters are outliers



Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- **Data Transformation: Normalization**
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

KENT STATE UNIVERSITY

Data Transformation: Normalization

- For a number of algorithms (e.g. distance based classifiers), normalization helps to prevent that attributes with large ranges out-weight attributes with small ranges

Common Methods:

- min-max normalization

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- z-score normalization

$$x = \frac{x - \bar{\bar{x}}}{s}$$

Normalization Example

Age	min-max (0-1)	z-score
44	0.421	0.450
35	0.184	-0.450
34	0.158	-0.550
34	0.158	-0.550
39	0.289	-0.050
41	0.342	0.150
42	0.368	0.250
31	0.079	-0.849
28	0.000	-1.149
30	0.053	-0.949
38	0.263	-0.150
36	0.211	-0.350
42	0.368	0.250
35	0.184	-0.450
33	0.132	-0.649
45	0.447	0.550
34	0.158	-0.550
65	0.974	2.548
66	1.000	2.648
38	0.263	-0.150

Agenda

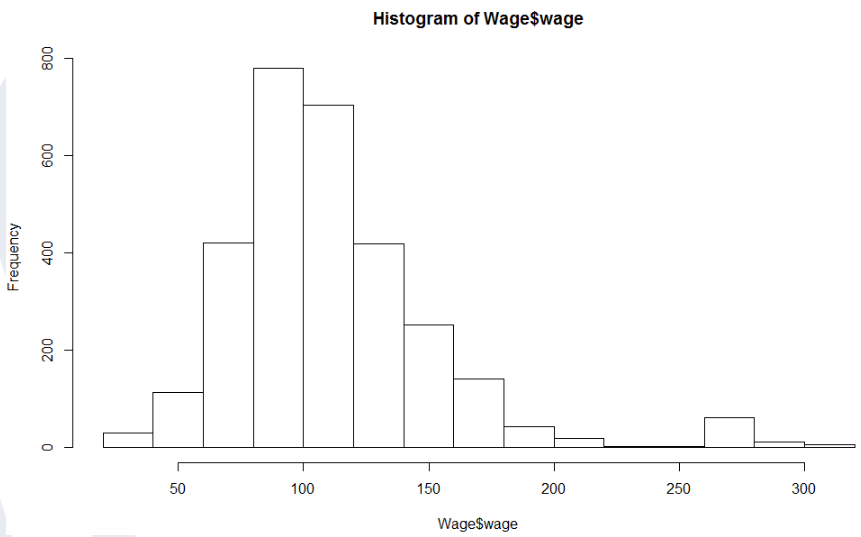
- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- **Data Transformation: Logarithmic Transformation**
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

Data Transformation: Log Transform

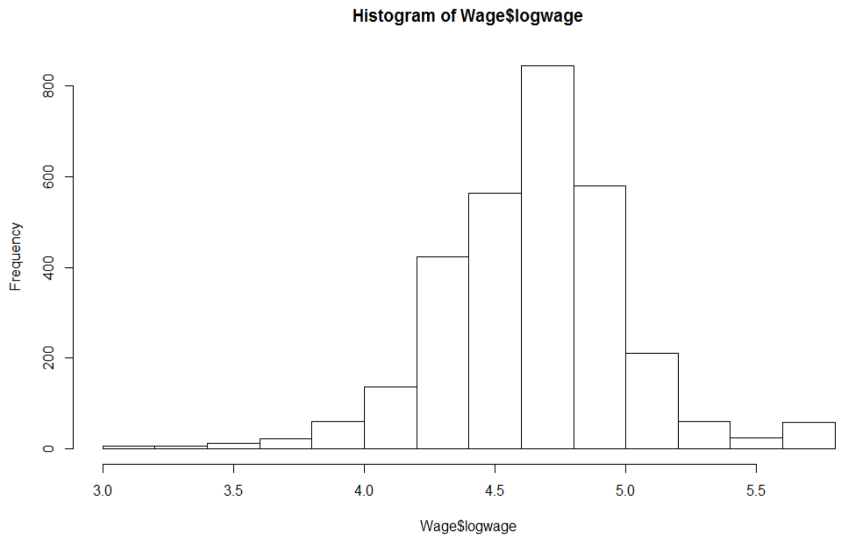
- It might also be beneficial to apply other transformation such as logarithmic transform to reduce the skewness of the data.
- This is a common practice for monetary values, for example which normally tend to have a long-tail distribution .
- You should make sure that all values are positive and greater than zero in the original variable to avoid invalid results.
- A fixed positive constant is normally added beforehand to avoid such scenarios.

Example From The Previous Lecture

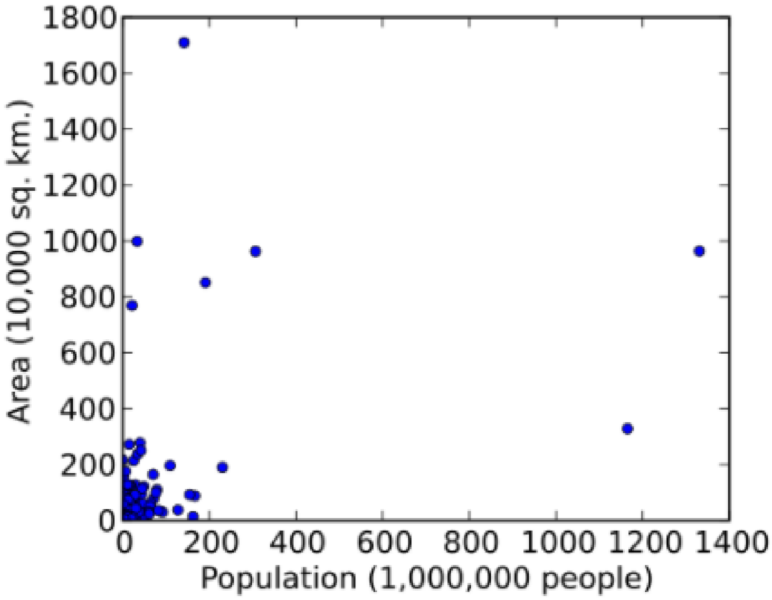
```
> hist(Wage$wage)
> library(moments)
> skewness(Wage$wage)
[1] 1.681489
```

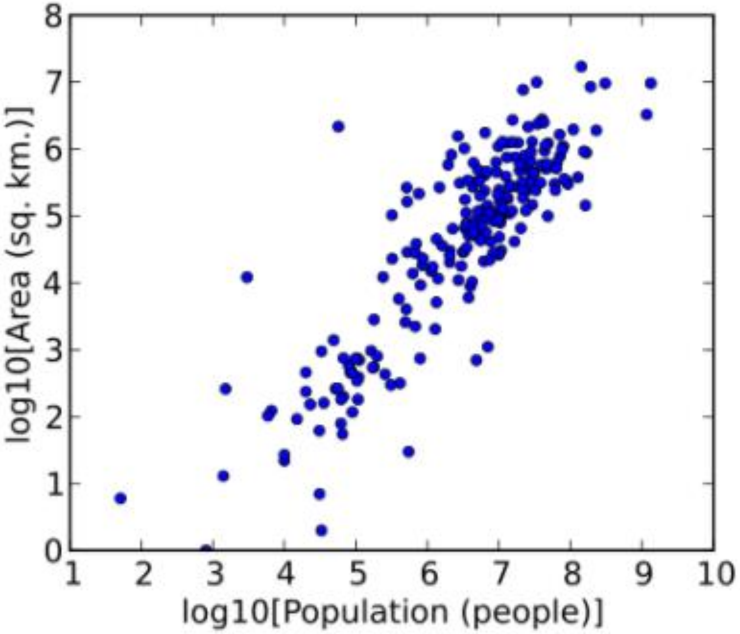



```
> hist(Wage$logwage)
> skewness(Wage$logwage)
[1] -0.1235535
```



Data Transformation: Log Transform





Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- **Data Transformation: Dummy Variables**
- Data Reduction: NZV Variable Removals
- Data Reduction: PCA Transformation

KENT STATE UNIVERSITY

Dummy Variables

- Dummy variables are artificially defined variables designed to convert a categorical variables with multiple levels into individual binary variables.

Original Data

```
     Price Mileage   Type
214  19981   24323  sedan
299  21757    1853  sedan
460  15047   12305  sedan
728  15327    4318  sedan
162  20628   20770  sedan
718  16714   26328  sedan
```

```
levels(carSubset$Type)
[1] "convertible" "coupe"       "hatchback"   "sedan"       "wagon"
```

Data with Dummy Variable

```
     Mileage convertible coupe hatchback sedan wagon
214    24323           0     0         0     1     0
299     1853           0     0         0     1     0
460    12305           0     0         0     1     0
728     4318           0     0         0     1     0
162    20770           0     0         0     1     0
718    26328           0     0         0     1     0
```

Dummy Variables: Example in R

```
> library(ISLR)
> summary(Wage)
      year            age                     maritl              race                  education
 Min.   :2003   Min.   :18.00   1. Never Married: 648   1. White:2480   1. < HS Grad        :268
 1st Qu.:2004   1st Qu.:33.75   2. Married      :2074   2. Black: 293   2. HS Grad          :971
 Median :2006   Median :42.00   3. Widowed      :  19   3. Asian: 190   3. Some College     :650
 Mean   :2006   Mean   :42.41   4. Divorced     : 204   4. Other:  37   4. College Grad     :685
 3rd Qu.:2008   3rd Qu.:51.00   5. Separated    :  55                   5. Advanced Degree:426
 Max.   :2009   Max.   :80.00

                     region               jobclass               health        health_ins      logwage
 2. Middle Atlantic   :3000   1. Industrial :1544   1. <=Good      : 858   1. Yes:2083   Min.   :3.000
 1. New England       :   0   2. Information:1456   2. >=Very Good:2142   2. No : 917   1st Qu.:4.447
 3. East North Central:   0                                                              Median :4.653
 4. West North Central:   0                                                              Mean   :4.654
 5. South Atlantic    :   0                                                              3rd Qu.:4.857
 6. East South Central:   0                                                              Max.   :5.763
 (Other)              :   0
```

Dummy Variables: Example in R

```
> simpleModel<-dummyVars(~ race, data = wage, levelsOnly = TRUE)
> race_convereted<-predict(simpleModel, wage)
> head(race_convereted)
       1. White 2. Black 3. Asian 4. Other
231655        1        0        0        0
86582         1        0        0        0
161300        1        0        0        0
155159        0        0        1        0
11443         1        0        0        0
376662        1        0        0        0
```

R Caret Package

- One of the most powerful R packages for data mining applications with several useful functions and utilities
- Highly recommended for automation and optimization of modeling building procedures
- Includes many functions for data preprocessing and transformation as well (see https://topepo.github.io/caret/pre-processing.html)

R Caret Package

```
> # Created:     Aug 26, 2017
> # @author      Rouzbeh Razavi, PhD (rrazavi@kent.edu)
> # @version:    1.0
> # File:        Data_Preprocessing_Caret.r
> # Comment:     Data Preprocessing Using Caret
> ################################################################
> rm(list = ls())
> library(caret)
> library(corrplot)
> library(RANN)
> ########################### Imputation ############################
> colMeans(is.na(airquality)) #determine the percentage of NA values per variable
     Ozone    Solar.R       Wind       Temp      Month        Day 
0.24183007 0.04575163 0.00000000 0.00000000 0.00000000 0.00000000 
> 
> # Median imputation of missing values
> preProc_1<-preProcess(airquality, method = c("medianImpute"))
> airquality_imputed<-predict(preProc_1, airquality)
> colMeans(is.na(airquality_imputed))
  Ozone Solar.R    Wind    Temp   Month     Day 
      0       0       0       0       0       0 
> summary(airquality_imputed)
     Ozone            Solar.R           Wind             Temp           Month            Day      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.   :5.000   Min.   : 1.0  
 1st Qu.: 21.00   1st Qu.:120.0   1st Qu.: 7.400   1st Qu.:72.00   1st Qu.:6.000   1st Qu.: 8.0  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00   Median :7.000   Median :16.0  
 Mean   : 39.56   Mean   :186.8   Mean   : 9.958   Mean   :77.88   Mean   :6.993   Mean   :15.8  
 3rd Qu.: 46.00   3rd Qu.:256.0   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000   Max.   :31.0  
```

R Caret Package

```
> # k-NN imputation of missing values
> # When using knnImpute data is scaled and centered by default
> preProc_2<-preProcess(airquality, method = c("knnImpute"),k=5)
> airquality_imputed<-predict(preProc_2, airquality)
> colMeans(is.na(airquality_imputed))
  Ozone Solar.R    Wind    Temp   Month     Day 
      0       0       0       0       0       0 
> summary(airquality_imputed)
     Ozone              Solar.R               Wind               Temp              Month          
 Min.   :-1.24680   Min.   :-1.98684   Min.   :-2.3439   Min.   :-2.3119   Min.   :-1.407294  
 1st Qu.:-0.67083   1st Qu.:-0.75430   1st Qu.:-0.7259   1st Qu.:-0.6215   1st Qu.:-0.701340  
 Median :-0.24643   Median : 0.13401   Median :-0.0731   Median : 0.1181   Median : 0.004614  
 Mean   : 0.00666   Mean   :-0.00895   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000  
 3rd Qu.: 0.63268   3rd Qu.: 0.77803   3rd Qu.: 0.4378   3rd Qu.: 0.7520   3rd Qu.: 0.710568  
 Max.   : 3.81566   Max.   : 1.64414   Max.   : 3.0492   Max.   : 2.0198   Max.   : 1.416522  
      Day          
 Min.   :-1.67002  
 1st Qu.:-0.88035  
 Median : 0.02212  
 Mean   : 0.00000  
 3rd Qu.: 0.81178  
 Max.   : 1.71426  
```

R Caret Package

```
> # Usinng Bagged Tree imputation of missing values
> preProc_3<-preProcess(airquality, method = c("bagImpute"))
> airquality_imputed<-predict(preProc_3, airquality)
> colMeans(is.na(airquality_imputed))
  Ozone Solar.R    Wind    Temp   Month     Day
      0       0       0       0       0       0
> summary(airquality_imputed)
     Ozone            Solar.R            Wind             Temp           Month            Day      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.   :5.000   Min.   : 1.0  
 1st Qu.: 20.00   1st Qu.:118.0   1st Qu.: 7.400   1st Qu.:72.00   1st Qu.:6.000   1st Qu.: 8.0  
 Median : 31.00   Median :201.0   Median : 9.700   Median :79.00   Median :7.000   Median :16.0  
 Mean   : 41.57   Mean   :184.9   Mean   : 9.958   Mean   :77.88   Mean   :6.993   Mean   :15.8  
 3rd Qu.: 61.00   3rd Qu.:256.0   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000   Max.   :31.0  
> 
> ########################## Find Highly Correlated Variables ##########################
> cor(airquality,use="complete.obs")
               Ozone     Solar.R        Wind       Temp        Month          Day
Ozone    1.000000000  0.34834169 -0.61249658  0.6985414  0.142885168 -0.005189769
Solar.R  0.348341693  1.00000000 -0.12718345  0.2940876 -0.074066683 -0.057753801
Wind    -0.612496576 -0.12718345  1.00000000 -0.4971897 -0.194495804  0.049871017
Temp     0.698541410  0.29408764 -0.49718972  1.0000000  0.403971709 -0.096545800
Month    0.142885168 -0.07406668 -0.19449580  0.4039717  1.000000000 -0.009001079
Day     -0.005189769 -0.05775380  0.04987102 -0.0965458 -0.009001079  1.000000000
> corrplot(cor(airquality,use="complete.obs"))
> highlyCorDescr <- findCorrelation(cor(airquality,use="complete.obs"), cutoff = .6)
> highlyCorDescr
[1] 4 1
> ########################## Transformation ##########################
```

R Caret Package

```
> ########################## Transformation ##########################
>
> #Scale and Center
> preProc_4<-preProcess(airquality, method = c("scale","center"))
> airquality_scaled<-predict(preProc_4, airquality)
> summary(airquality_scaled) #notice the mean for all values are 0
     Ozone             Solar.R              Wind               Temp              Month
 Min.   :-1.2468   Min.   :-1.9868   Min.   :-2.3439   Min.   :-2.3119   Min.   :-1.407294
 1st Qu.:-0.7315   1st Qu.:-0.7793   1st Qu.:-0.7259   1st Qu.:-0.6215   1st Qu.:-0.701340
 Median :-0.3222   Median : 0.2117   Median :-0.0731   Median : 0.1181   Median : 0.004614
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000
 3rd Qu.: 0.6403   3rd Qu.: 0.8086   3rd Qu.: 0.4378   3rd Qu.: 0.7520   3rd Qu.: 0.710568
 Max.   : 3.8157   Max.   : 1.6441   Max.   : 3.0492   Max.   : 2.0198   Max.   : 1.416522
 NA's   :37        NA's   :7
      Day
 Min.   :-1.67002
 1st Qu.:-0.88035
 Median : 0.02212
 Mean   : 0.00000
 3rd Qu.: 0.81178
 Max.   : 1.71426
```

R Caret Package

```
>
> preProc_5<-preProcess(airquality, method = c("range"))
> airquality_scaled<-predict(preProc_5, airquality)
> summary(airquality_scaled) #notice the min and max values
     Ozone            Solar.R            Wind             Temp            Month             Day        
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.1018   1st Qu.:0.3326   1st Qu.:0.3000   1st Qu.:0.3902   1st Qu.:0.2500   1st Qu.:0.2333  
 Median :0.1826   Median :0.6055   Median :0.4211   Median :0.5610   Median :0.5000   Median :0.5000  
 Mean   :0.2463   Mean   :0.5472   Mean   :0.4346   Mean   :0.5337   Mean   :0.4984   Mean   :0.4935  
 3rd Qu.:0.3728   3rd Qu.:0.7699   3rd Qu.:0.5158   3rd Qu.:0.7073   3rd Qu.:0.7500   3rd Qu.:0.7333  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
 NA's   :37       NA's   :7                                                                           
>
> ######################## Find varibles that are linear combinations of each other ###########3
> airquality_new <-cbind(airquality_imputed, New_Variable=6.5*airquality_imputed$Ozone+2*airquality_i
ed$Wind)
> head(airquality_new) # see the new variable
     Ozone  Solar.R Wind Temp Month Day New_Variable
1 41.00000 190.0000  7.4   67     5   1     281.3000
2 36.00000 118.0000  8.0   72     5   2     250.0000
3 12.00000 149.0000 12.6   74     5   3     103.2000
4 18.00000 313.0000 11.5   62     5   4     140.0000
5 14.96518 206.0734 14.3   56     5   5     125.8736
6 28.00000 207.6060 14.9   66     5   6     211.8000
> findLinearCombos(airquality_new)
$linearCombos
$linearCombos[[1]]
[1] 7 1 3
```

Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- **Data Reduction: NZV Variable Removals**
- Data Reduction: PCA Transformation

Data Reduction

- In some cases, where there are too many variables in the dataset, it might be a good idea to reduce the the number of variables. This is different from variable selection, which is a supervised approach and will be discussed in modelling lecture.

- Variable selection, chooses the variable wich are relevant to a specific modeling task.

- Data reduction, on the other hand, tries to transform variables into smaller set of variables while preserving the predictive power or removing the variables that are completely useless.

Removing Zero Variance (ZV) or Near Zero Variance (NZV) Variables

- If a variable has a constant value for all observations, the variance is zero (RZ) e.g. a gender variable which is “Male” for all records (no predictive information)

- There are also cases that the variance is very close to zero (NZV), in most cases those variables also have very limited contributions.

- Caret has functions nearZeroVar() that can detect such variables.

```
library(caret)
mtcars_2<-
cbind(mtcars,
new_variable_1=1,
new_variable_2=c('Reliable'))
summary(mtcars_2)
nzv <- nearZeroVar(mtcars_2)
nzv
mtcars_filtered <- mtcars_2[, -nzv]
summary(mtcars_filtered)
```

Removing Zero Variance (ZV) or Near Zero Variance (NZV) Variables

```
> summary(mtcars_2)
      mpg             cyl             disp             hp             drat             wt       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760   Min.   :1.513  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080   1st Qu.:2.581  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695   Median :3.325  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597   Mean   :3.217  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920   3rd Qu.:3.610  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930   Max.   :5.424  
      qsec             vs               am              gear            carb       new_variable_1
 Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000   Min.   :1.000   Min.   :1     
 1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1     
 Median :17.71   Median :0.0000   Median :0.0000   Median :4.000   Median :2.000   Median :1     
 Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688   Mean   :2.812   Mean   :1     
 3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:1     
 Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000   Max.   :8.000   Max.   :1     
  new_variable_2
 Reliable:32    
```

```
> nzv <- nearZeroVar(mtcars_2)
> nzv
[1] 12 13
```

Agenda

- Subsetting and Filtering Data
- Understanding Data
- Data Preprocessing and Quality
- Data Quality: Missing Values
- Data Quality: Outliers
- Data Transformation: Normalization
- Data Transformation: Logarithmic Transformation
- Data Transformation: Dummy Variables
- Data Reduction: NZV Variable Removals
- **Data Reduction: PCA Transformation**

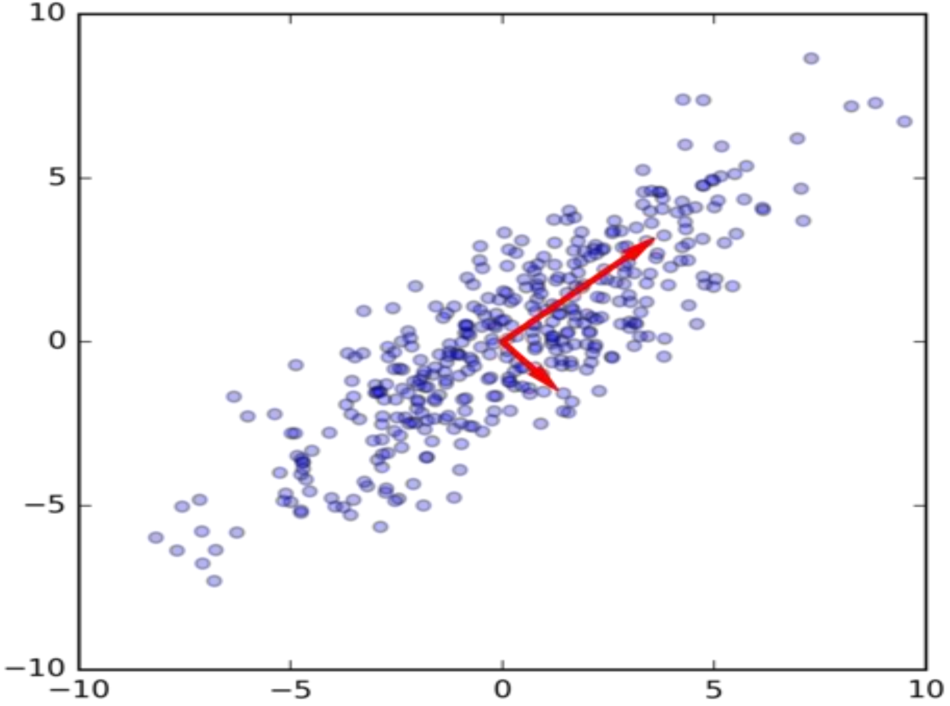
Principle Component Analysis (PCA)

- Objective: find the 'best' low dimension space that conveys maximum useful information

- We wish to explain/summarize the underlying variance-covariance structure of a large set of variables through a few linear combinations of these variables

- The math behind PCA is complex and beyond this course. Click here to get a high level idea of how PCA works. And here is a gentle math introduction of PCA (just in case!) .

PCA Visual Example



Principle Component Analysis (PCA)

- You have to select sufficient components to assure that high percentage of data variability is captured (i.e. minimum information loss)

- The problem with PCA is that new variables can not be interpreted (i.e. do not have any business meaning) since they are linear combination of original variables.

- You can use caret preprocess() method to compute PCs. See example next slide.

PCA Example: From 113 Variables to 57 Variables (and only 5% information loss)

```
> library(caret)
> library(corrplot)
> #install.packages('AppliedPredictiveModeling')
> library(AppliedPredictiveModeling)
> data(segmentationOriginal)
> segData <- subset(segmentationOriginal, Case == "Train")
> # Now remove the columns
> segData <- segData[, -(1:3)]
> nzv <- nearZeroVar(segData)
> segData <- segData[, -nzv]
>
> ncol(segData) # segData has 113 variables (columns)
[1] 113
> tranformation <- preProcess(segData, method = c( "center", "scale", "pca"))
> tranformation
Created from 1009 samples and 113 variables

Pre-processing:
  - centered (113)
  - ignored (0)
  - principal component signal extraction (113)
  - scaled (113)

PCA needed 57 components to capture 95 percent of the variance
> segData_transformed <- predict(tranformation, segData)
> ncol(segData_transformed) #transformed data has 57 variables (columns)
[1] 57
```

Multiple transformations at the same time

Data Timeliness

- Make sure the data that you are including into your model is still relevant.

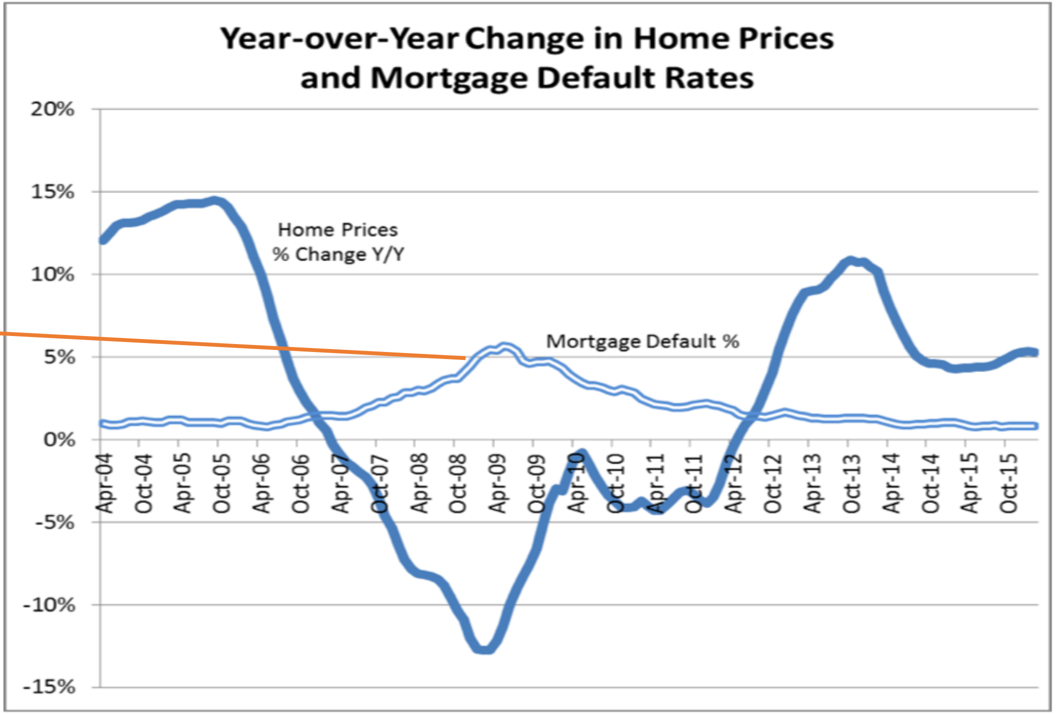
- Note that the purpose of building a model is to use it for scoring future observations. Therefore, if your past data does not hold any resemblance to your future data that you are going to use for scoring, the model will be inaccurate.

- Such statistical drifts can happen due to changes in the business environment and are very common.

- Make sure that you check the statistical properties (e.g. mean of different variables) over different time intervals to assure consistency.

Data Timeliness: Example

Models build using data prior to recessions would be very inaccurate



Source: S&P/Case-Shiller National Home Price Index; S&P/Experian Consumer Credit First Mortgage Default Index


ANY
QUESTIONS
?