# BDA

Big Data Analytics

## Data examples

CERN's Large Hadron Collider(LHR) generates 40 Terabytes/second

All the catalogued books in America's library's 15TB

Bin Laden's death 5106 tweets/second

RFID (radio freq id) tags 30 billion per year

1 billion transistors per person

## Data Sizes

| | | |
|---|---|---|
| Bit (b) | 1 or 0 | binary code 0 or 1 |
| Byte (B) | 8 bits | an English word or number |
| Kilobyte (KB) | 1,000 or $2^{10}$ bytes | One pages of text is 2KB |
| Megabyte (MB) | 1,000KB or $2^{20}$ bytes | Complete works of Shakespeare 5MB or a pop song 4MB |
| Gigabyte (GB) | 1,000MB or $2^{30}$ bytes | Two-hour film is 1-2 GB |
| Terabyte (TB) | 1,000GB or $2^{40}$ bytes | All the books in Americas library 15TB |
| Petabyte (PB) | 1,000TB or $2^{50}$ bytes | Google processes 1PB per hour |
| Exabyte (EB) | 1,000PB or $2^{60}$ bytes | 10 billion copies of the Economist |
| Zettabyte (ZB) | 1,000EB or $2^{70}$ bytes | total amount of info in existence 1.27 ZB |
| Yottabyte (YB) | 1,000 ZB or $2^{80}$ bytes | too big to imagine |

## Characterises of Big data

- Volume – The data sizes (KB, MB, GB, TB, PB, EB, ZB, YB
- Velocity – real time, sparse, batch, interval
- Variety – Test, CSV, Photos, SMS, XML, Video, Unstructured
- Veracity – Uncertainly of Data, biggest challenge keeping data clean

## Data Products

- Twitter – 328 million users, 500 million tweets or 2.1 billion queries per day
- Netflix -- personalised move ratings, put blockbuster out of business
- LinkedIn/Facebook
- Google

## Sexist Job of the Century

- Data Scientist ☺

# BDA

## Data Stages

- Collect
- Clean
- Integrate
- Analysis
- Visualisation

## Statistical Learning

Statistical learning is all about how to estimate f. Why estimate f and how do we estimate f. Use training data and statistical learning method.

Trade-off between prediction accuracy and model interpretability

Supervised vs Unsupervised learning

Regression vs Classification problems

→ 1900 method of least squares (linear regression) – Legendre and Gauss
→ 1936 Linear discriminant analysis – Fisher
→ 1940s logistic regression – various authors
→ 1970s generalized linear models – Nelder Wedderburn
→ 1980s classification and regression trees, cross-validation – Breiman, Friedman, Olshen, Stone
→ 1986 Generalised addition models – Haste and Tibshirani

## Two reasons for estimating f

- Prediction – to produce good estimates for f, variance of e is not too large and get accurate predictions for the response, Y based on a new value X
- Inference – the type of relationship between Y and X's, which predictor affected response, positive or negative relationships

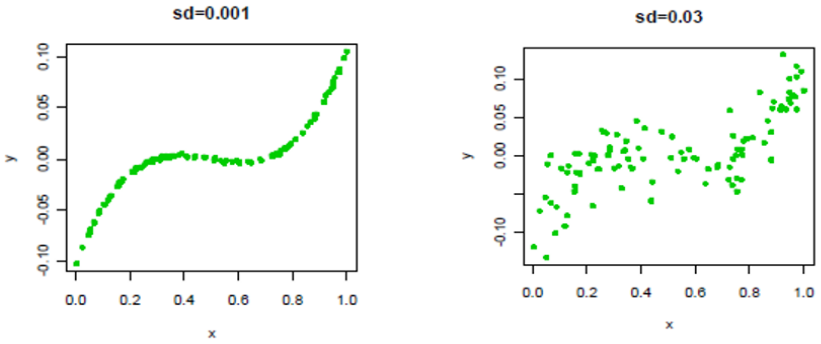
Modelling relationship between Y and X's, where f is an unknown function and e is a random error that cannot be measure and is independent of X

$$Y = f(X) + e$$

Irreducible error no matter how well we estimate f, we cannot reduce the error introduced by e. e has zero mean.

Standard deviation (sd) measures the amount of variation from the average. The more spread out the data is the higher its variance will be.

# BDA




The standard deviation of the e's will estimate the difficulty of estimating f

## Parametric and Non-parametric methods

Parametric methods makes assumptions about the functional form of f (i.e comes up with a model). It reduces the problem of estimating f down to one of the estimating set of parameters. Then fits training data to model to estimate unknown parameters.

Non-parametric methods do not make explicit assumptions about the functional form of f. Advantages accurately fit a wider range of possible shapes of f. Disadvantages a very large number of observations is required to obtain an accurate estimate of f.

Non-linear regression methods are more flexible and can provide more accurate estimates.

## Prediction Accuracy Vs Model Interoperability

If more flexible methods are more realistic why not use them? Two reasons

- Interpretability – a simple method like linear regression produces a model which is much easier to interpret (inference part is better). Often possible to get accurate prediction with a simple instead of a complicated model.
- Overfitting – Even if you are only interested in prediction and not cared about inference there is a potential for overfitting in highly flexible methods.

## Overfitting

- In overfitting a statistical model describes random error or noise instead of underlying relationship
- Overfitting occurs when a model is excessively complex such as having too many parameters relative to the number of observations.
- A model that has been overfit has poor predictive performance, as it overreacts to minor fluctuations in the training data

## Poor estimates

- Non-linear regression methods can also be too flexible and produce poor estimates for f.

# BDA

## Supervised Vs Unsupervised

- Supervised is where both the predictors X's and response/target Y, are observed.
  Examples: Regression (liner and multiple regression)
  Classification (logistic regression, Decision trees, Tree based methods (boosting, random forests), SVM(Support Vector Machines), Ensemble methods
- Unsupervised only the X's are observed. We lack a response variable that can supervise our analysis. Use X's to guess what Y would have been and build model from there.
  Example: Clustering, Dimension reduction

## R Matrix Code

To create a vector

```
> x <- c(1,2,3,4,5)
> x
[1] 1 2 3 4 5
```

To create a sequence

```
> y <- seq(1,10)
> y
 [1]  1  2  3  4  5  6  7  8  9 10
```

or shorthanded

```
> y <- 1:5
> y
[1] 1 2 3 4 5
```

To create a sequence skipping number

```
> odd_numbers <- seq(1, 20, by=2)
> odd_numbers
 [1]  1  3  5  7  9 11 13 15 17 19
```

```
> even_numbers <- seq(0, 20, by=2)
> even_numbers
 [1]  0  2  4  6  8 10 12 14 16 18 20
```

Vector operations give two vectors x and y must be same length for operations

```
> x <- c(2,4,6,8)
> length(x)
[1] 4
> x
[1] 2 4 6 8
```

```
> y <- 1:4
> length(y)
[1] 4
> y
[1] 1 2 3 4
```

Both the same length now to preform x + y and x * y

```
> x + y
[1]  3  6  9 12
```

```
> x * y
[1]  2  8 18 32
```

## Vectors

Create a vector x starting from 5.3 and ending at 8.00 and length is 10

```
> x <- seq(5.3,8,length.out = 10)
> x
 [1] 5.3 5.6 5.9 6.2 6.5 6.8 7.1 7.4 7.7 8.0
```

Create another vector y starting from 3.5 and ending at or less than 7.9 with each term is 0.4 more than the previous one

```
> y <- seq(3.5, 7.9, by=0.4)
> y
 [1] 3.5 3.9 4.3 4.7 5.1 5.5 5.9 6.3 6.7 7.1 7.5 7.9
```

Add x and y gives a warning message because they are different length

```
> x+y
 [1]  8.8  9.5 10.2 10.9 11.6 12.3 13.0 13.7 14.4 15.1 12.8 13.5
Warning message:
In x + y : longer object length is not a multiple of shorter object length
```

Increase all terms in x by 1

```
> x <- x + 1
> x
 [1] 6.3 6.6 6.9 7.2 7.5 7.8 8.1 8.4 8.7 9.0
```

## Matrix

Creating a matrix, order by columns by default

```
> x <- matrix(data = c(1,2,3,4), nrow=2, ncol =2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Matrix ordered by rows

```
> matrix(data = c(1,2,3,4), nrow=2, ncol =2, byrow =TRUE)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Dimension of Matrix

```
> dim(x)
[1] 2 2
```

Indexing data

```
> A <- matrix(1:16, 4,4)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

select one element

```
> A[2,3]
[1] 10
```

Select multiple rows and columns

# BDA

```
> A[1,]
[1]  1  5  9 13

> A[c(2,4), c(1,4)]
     [,1] [,2]
[1,]    2   14
[2,]    4   16
```

```
> A[ ,2]
[1] 5 6 7 8

> A[1:3, ]
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
```

## Indexing Data

With indexing data particularly negative index it removed that row or column not include it

```
> B <- matrix(data = seq(1, 39, by=2), ncol = 5,
+             nrow = 4, byrow = TRUE)
> B
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]   11   13   15   17   19
[3,]   21   23   25   27   29
[4,]   31   33   35   37   39
```

```
> B[c(2,4), c(2,3)]
     [,1] [,2]
[1,]   13   15
[2,]   33   35

> B[-c(1,3), -c(1,4,5)]
     [,1] [,2]
[1,]   13   15
[2,]   33   35
```

## Random normal distribution

Generate a vector of random normal variables n: sample size, default mean=0 and sd=1, each time different (random sampling)

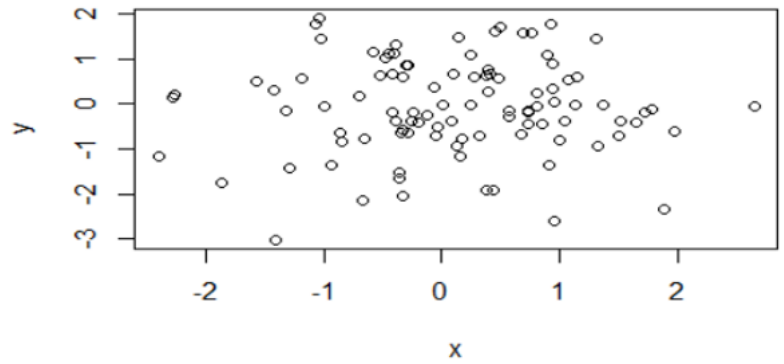
```
> set.seed(1)
> D <- rnorm(50, mean=0, sd=1)
> D
 [1] -0.62645381  0.18364332 -0.83562861  1.59528080  0.32950777
 [6] -0.82046838  0.48742905  0.73832471  0.57578135 -0.30538839
[11]  1.51178117  0.38984324 -0.62124058 -2.21469989  1.12493092
[16] -0.04493361 -0.01619026  0.94383621  0.82122120  0.59390132
[21]  0.91897737  0.78213630  0.07456498 -1.98935170  0.61982575
[26] -0.05612874 -0.15579551 -1.47075238 -0.47815006  0.41794156
[31]  1.35867955 -0.10278773  0.38767161 -0.05380504 -1.37705956
[36] -0.41499456 -0.39428995 -0.05931340  1.10002537  0.76317575
[41] -0.16452360 -0.25336168  0.69696338  0.55666320 -0.68875569
[46] -0.70749516  0.36458196  0.76853292 -0.11234621  0.88110773
```

```
> var(D)
[1] 0.6912159
> sqrt(var(D))
[1] 0.8313939
> sd(D)
[1] 0.8313939
```

## Basic Graphs

To plot variable or functions

```
> x = rnorm(100)
> y = rnorm(100)
> plot(x, y)
```

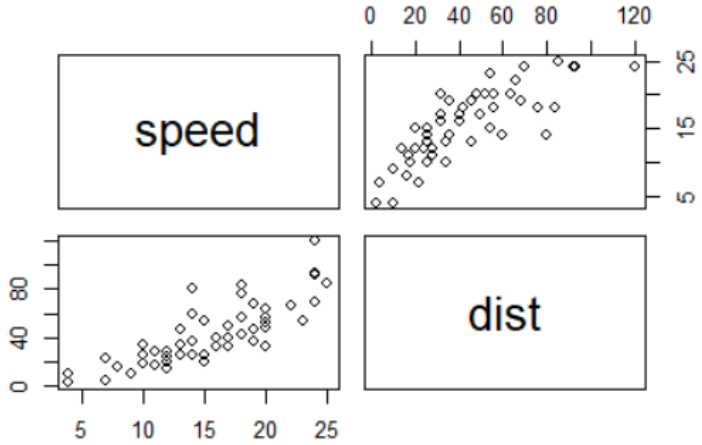


# BDA

## Additional graphical commands

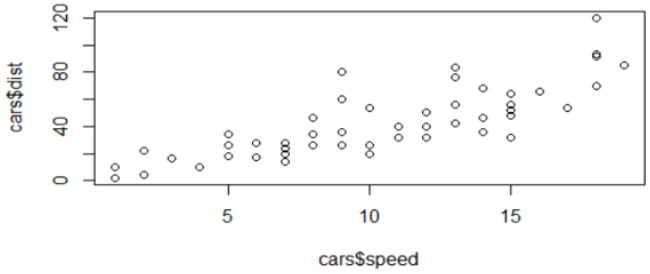
Pairs create a scatter plot matrix

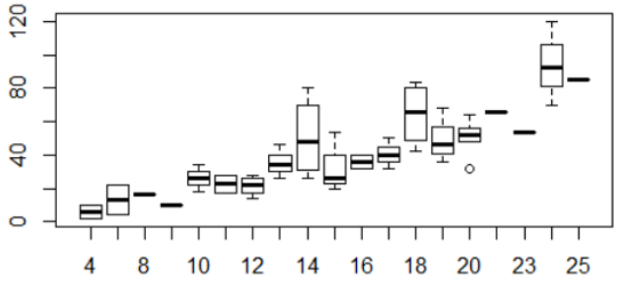
```
> pairs(cars)
```




cylinders variable is stored as a numeric (quantitative) vector, but can be treated as a categorical (qualitative) variable use as.factor() as a result boxplot will automatically produce by the plot function.

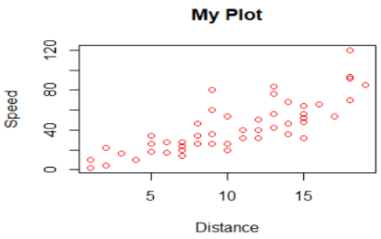
```
> plot(cars$speed, cars$speed)
> cars$speed = as.factor(cars$speed)
> plot(cars$speed, cars$dist)
```




You can also add labels and colour

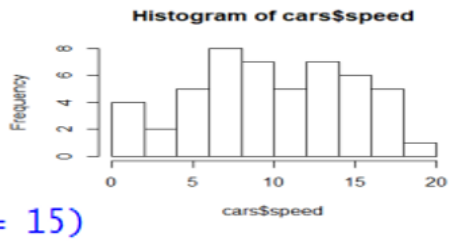
```
> plot(cars$speed, cars$dist, xlab = "Distance",
+      ylab = "Speed", main = "My Plot", col = "red")
```



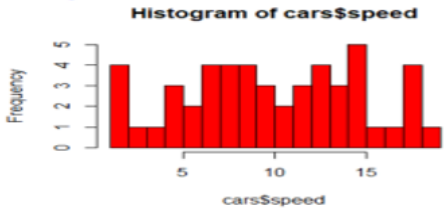
# BDA

## Plot histograms

```
> hist(cars$speed)
```




```
> hist(cars$speed, col =2 , breaks = 15)
```




## Select subsets of data

Use square brackets to put condition on data and view subsets

```
> cars$speed[cars$dist == 10]
[1] 1 4

> cars$speed[cars$dist == 10 & cars$speed == 12]
numeric(0)
```

## Show summary of data

Use summary () on whole data or variables

```
> summary(cars)
     speed           dist
 Min.   : 1.00   Min.   :  2.00
 1st Qu.: 7.00   1st Qu.: 26.00
 Median :10.00   Median : 36.00
 Mean   :10.34   Mean   : 42.98
 3rd Qu.:14.00   3rd Qu.: 56.00
 Max.   :19.00   Max.   :120.00

> summary(cars$speed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    7.00   10.00   10.34   14.00   19.00
```

## Statistics

The purpose of statistics is to develop and apply methodology for extracting useful information from data.

## Scales of Measurement

Scales determine the amount of information contained in the data

- Nominal – no order to the data just name and labels

- Ordinal – order or count the data cannot add or subtract
- Interval – can add and subtract the data but can't multiple or divide
- Ratio – can add, sibtract, multiply and divide has zero value

Qualitative and Quantitative Data

- Type of statistical analysis depend on the classification of data, quantitative or qualitative
- Generally more methods for quantitative
- Qualitative/Categorical = Nominal or Ordinal
- Quantitative = Interval or Ratio

## Descriptive statistics

- Exploring, visualising and summarising data without fitting the data to any models
- Numerical measures
- Tabular and graphical presentation (Histogram, Box plot, Scatter diagram, Freq dist table)
- If values from population then measurement are population parameters
- If values are from a sample, then measurements are sample statistics. A sample statistic is a population parameter.

- Univariate analysis, describing the distribution of a single variable
  - Measures of central tendency (mean, median, mode) $\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$

    Mean is highly influenced by outliers whenever extreme values present so median is a better measurement. Mode rare in continuous data.
  - Measures of spread (variance, standard deviation) $s^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})2}{n-1}$

    variance average of squared difference from the mean. SD is the square root of variance. Sample variance/SD needs to be unbiased divide by n-1. Population just divide by N
  - Measure of dispersion (Range, Quartiles, Interquartile range)

- Bivariate analysis, describing the relationship between a pair of variables
  - Quantitative measures (Correlation and Covariance)
    Difference covariance is dimensional quantity, can't compare. Correlation is dimensionless quality and always between -1 and 1, makes for easier comparison. Sample data divide covariance by n -1.

$$Cov(X, Y) = \sum_{i=1}^{n} \frac{(x - \bar{x})(y - \bar{y})}{n - 1}$$

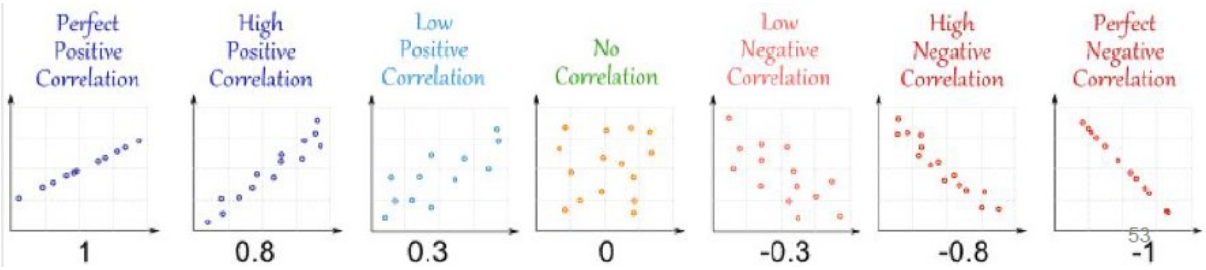
  - Covariance measures how much the movement in one variable predicts the movement in a corresponding variable, the degree of linkage between two

variables. Positive covariance indicates that greater values of one variable are paired with greater values of the other variable. Negative covariance indicates that greater values of one variable are paired with lesser values of another variable.

- Correlation, is a dimensionless quality to facilitate comparison. When two sets of data are strongly linked to together we say they are highly correlated

$$Corr(X,Y) = \sum_{i=1}^{n} \frac{Cov(X,Y)}{s_x s_y}$$

- Correlation is positive when values increase together and negative when one value increase and other decreases.




Remember correlation is a measure of linear association not causation! Just because two variables re highly correlated does not mean that one variable is the cause of the other

R code

```
> x <- seq(1, 20, by=2)
> mean(x)
[1] 10
> var(x)
[1] 36.66667
> sqrt(var(x))
[1] 6.055301
> sd(x)
[1] 6.055301
```

```
> x <- seq(2, 22, by=2)
> y <- seq(1, 21, by=2)
> cov(x,y)
[1] 44
> cor(x,y)
[1] 1
```

**Scale of Measurement**

- Nominal
  Mode
- Ordinal
  Median, Mode
- Interval
  Mode, Median, Mean
  Range, Variance, Standard deviation

# BDA

- Ratio
  Mode, Median, Mean
  Range, Variance, Standard deviation

And many more: geometric mean, harmonic mean, coefficient of variation, and all the other statistical measures

## Tabular and graphical presentation

- **Qualitative**
  Tabular Methods
  Frequency Distribution
  Relative Frequency Distribution
  Percentage Frequency Distribution
  Crosstabulation

  Graphical Methods
  Pie Chart
  Bar Chart

- **Quantitative**

  Tabular Methods
  Frequency Distribution
  Relative Frequency Distribution
  Cumulative Frequency Distribution
  Cumulative Relative Frequency Distribution
  Stem and Leaf plots
  Cross Tabulation

  Graphical Methods
  Dot Plot
  Histogram
  Box Plot
  Scatter Plot
  Ogive (cumulative frequency graph)

## Inferential statistics

- Identification of a suitable model
- Testing either predictions or hypotheses of the model

## Simple Linear Regression

# BDA

- Predicts a quantitative response
- Single predictor value, X just one X!
- Multiple linear regression more than one predictor variable, a few X's

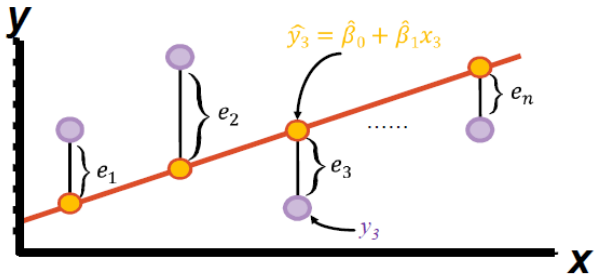
| Linear functions | Non-Linear functions |
|---|---|
| • linear functions are linear with respect to the variables<br>• $f(x) = -0.5x + 2$<br>• $f(x_1, x_2) = 2x_1 + \sqrt{5}x_2 - 0.45x_3$ | • Is where the pridictors X's are no longer linear or to the power of 1<br>• $f(x_1, x_2) = 2x_1^2 - 7x_2^3$<br>• $f(x_1, x_2, x_3) = -6x_1^{\square} + 5x_2^5 - \sqrt{3}x_3^{\square}$ |

$$Y \approx \beta_0 + \beta_1 X$$

- Y is the response variable
- βs are the parameters
- X is the predictor variable

Regressing Y on X

- Use the training data to produce estimates for $\hat{\beta}_0, \hat{\beta}_1$
- Use $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ to predict Y as $\hat{y}$ on the basis X = x
- Plot data using scatter plots
- $y_i$ is the observed value, the real value!
- $e_i = y_i - \hat{y}_i$ is the $i$th residual (residual = observed – predicted)
- Residual sum of squares, $RSS = e_1^2 + e_2^2 + \cdots + e_n^2$
- $RSS = (y_1 - \hat{\beta}_0 + \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 + \hat{\beta}_1 x_2)^2 + \cdots + (y_n - \hat{\beta}_0 + \hat{\beta}_1 x_n)^2$




- The least square lines $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ has the following two properties
  - The sum of the residuals equals 0, aka residual mean = 0
  - The residual sum of squares is minimised
  - The minimised least squares equations to calculate the coefficient estimates

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$\hat{\beta}_o = \bar{y} - \hat{\beta}_1 \bar{x}$$

# BDA

Working example of advert expense Vs Sales

| Ad $x_i$ | Sale $y_i$ | $(x_i - \bar{x})$ | $(y_i - \bar{y})$ | $(x_i - \bar{x})(y_i - \bar{y})$ | $(x_i - \bar{x})(x_i - \bar{x})$ |
|---|---|---|---|---|---|
| 1 | 1 | -2 | -1 | 2 | 4 |
| 2 | 1 | -1 | -1 | 1 | 1 |
| 3 | 2 | 0 | 0 | 0 | 0 |
| 4 | 2 | 1 | 0 | 0 | 1 |
| 5 | 4 | 2 | 2 | 4 | 2 |

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{1+2+3+4+5}{5} = \frac{15}{5} = 3$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} = \frac{1+1+2+2+4}{5} = \frac{10}{5} = 2$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{7}{10} = 0.7$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = 2 - 0.7(3) = -0.1$$

Assessing accuracy of coefficient estimates, three lines

- True relationship Y = f(X) + e, e is the zero-mean random error term
- Population regression line Y = $\beta_0 + \beta_1 X + \varepsilon$, f is approx. by linear function
  $\varepsilon$ independent of X and is a catch all for what we miss with this simple model, reducible error is true relationship is probably not linear. Other variables that cause variation in Y. the best linear approx. to the true relationship between X and Y
- Least squares line $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$ use least square regression estimates which may under or over estimate the population parameters

## Standard Error

The closeness of a single sample mean $\hat{\mu}$ to the population mean μ. The standard error SE the average amount that this estimate $\hat{\mu}$ differs from μ. $\sigma^2$ is variance and $\sigma$ is the standard deviation. The more observations the smaller the SE will be and closer to zero it gets to because population mean is improving.

$$SE(\hat{\mu})^2 = \frac{\sigma^2}{n}$$

Standard error is used to compute confidence intervals and perform hypothesis tests. SE can be used to see how close $\hat{\beta}_0, \hat{\beta}_1$ are to $\beta_0$, $\beta_1$

$$SE(\hat{\beta}_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right]$$

# BDA

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

## Confidence Intervals (CI)

A 95% CI for $\beta_1$ means with 95% probability, the range will contain the true unknown value for $\beta_1$

A 2.5% CI for $\beta_0$ means with 2.5% probability, the range will contain the true unknown value for $\beta_0$

## Hypothesis Tests

The least square hypothesis test if $\beta_1 = 0$ or not

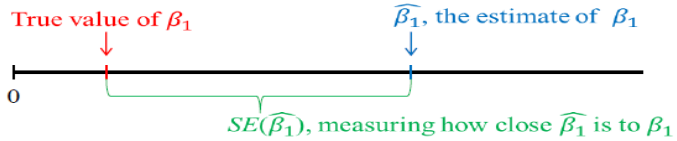
Null hypothesis, there is no relationship between X and Y $H_o: \beta_1 = 0$

Alternative hypothesis, there is a relationship between X and Y $H_a: \beta_1 \neq 0$

Use a t-test to $\hat{\beta}_1$ on estimate $\beta_1$

## t-value

How far $\hat{\beta}_1$ the estimate $\beta_1$is from 0 depends on the accuracy of $\hat{\beta}_1$ aka the standard error of $\beta_1$. High t-value means more possible X and Y are related or the number of standard deviations away from 0. Therefor is $SE(\hat{\beta}_1)$ is small may be strong evidence that $\beta_1 \neq 0$ hence a relationship otherwise if $SE(\hat{\beta}_1)$ is large may be strong evidence that $\beta_1 = 0$hence a no relationship




$$t = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)}$$

## p-value

Understanding the p-value is used to check if the hypothesis is accepted or rejected, use a level of significance $\alpha = 0.05$. If $p \leq \alpha$ reject the null hypothesis and accept the alternative. If $p > \alpha$ accept the null hypothesis. If p=value is small we can assume a relationship.

***Regression coefficients***

| | Coefficient | Std Err | t-value | p-value |
|---|---|---|---|---|
| Constant | 7.0326 | 0.4578 | 15.3603 | 0.0000 |
| TV | 0.0475 | 0.0027 | 17.6676 | 0.0000 |

$\hat{\beta}_1$ $SE(\hat{\beta}_1)$ t-value p-value

# BDA

## Residual Standard Error (RSE)

Least square line could never fit population model perfectly because of error term ε.

Population regression line: $Y = \beta_0 + \beta_1 X + \varepsilon$

Least squares line: $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$

RSE is the estimate of the standard deviation of ε. It quantifies the average amount that response, Y will deviate from population regression line. It measures the extent to which the model fits the data. So, it measures the lack of fit and is in the same units as Y but not always clear if a good fit.

## Measure of Fit: $R^2$

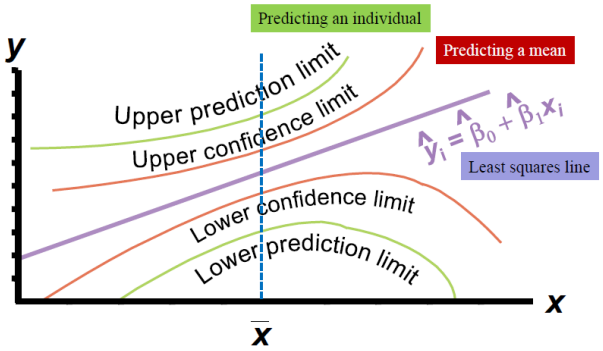
Measures the extent to which the model fits the data. Explains the some of the variation in Y by the variation in X's. It tell the proportion of variance that can be explained by X

$$R^2 = 1 - \frac{RSS}{\sum (Y_i - \bar{Y})^2} \approx 1 - \frac{\text{Ending Variance}}{\text{Starting Variance}}$$

The starting variance it the variability of the response before regression is performed. The ending variance is the amount of variability that is left unexplained after preforming regression. $R^2$ is always between 0 and 1. Zero means no variance has been explained and one means it has all been explain, prefect fit to data. Also $R^2 = \text{Cor}(X,Y)^2$ both measure the relationship between X and Y

## Confidence and Prediction Intervals

Prediction intervals (PI) is an estimate of an interval in which future observations for a particular individual will fall with a certain probability given what is already observed. PI are normally bigger than CI




Simple linear regression is a supervised learning method, prediction and parameterised method with dependent variable Y and independent variables X.

# BDA

## R Simple Linear Regression Code

To fit a simple linear regression model use lm(), It gives the least squares line parameter estimates

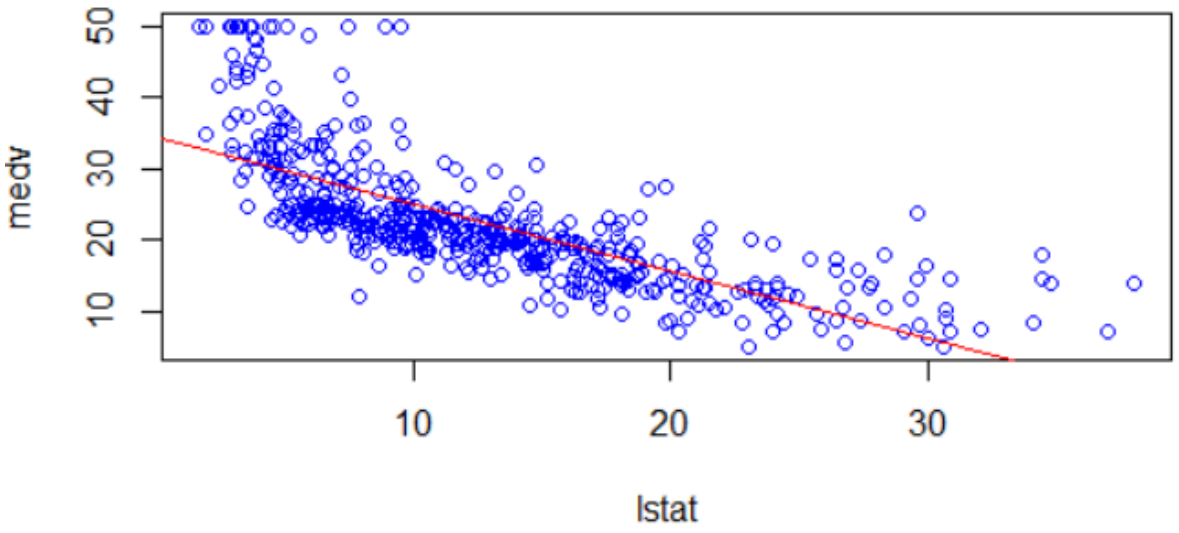
```
> library(MASS)
> library(ISLR)
> lm.fit = lm(medv ~ lstat, data = Boston)
> lm.fit

Call:
lm(formula = medv ~ lstat, data = Boston)

Coefficients:
(Intercept)        lstat
      34.55        -0.95
```

To obtain the confidence interval for the coefficient estimates

```
> confint(lm.fit)
                2.5 %     97.5 %
(Intercept) 33.448457 35.6592247
lstat       -1.026148 -0.8739505

> summary(lm.fit)

Call:
lm(formula = medv ~ lstat, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max
-15.168  -3.990  -1.318   2.034  24.500

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 34.55384    0.56263   61.41   <2e-16
lstat       -0.95005    0.03873  -24.53   <2e-16

(Intercept) ***
lstat       ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.216 on 504 degrees of freedom
Multiple R-squared:  0.5441,    Adjusted R-squared:  0.5432
F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16

> predict(lm.fit, data.frame(lstat=(c(5,10,15))), interval = "confidence")
       fit      lwr      upr
1 29.80359 29.00741 30.59978
2 25.05335 24.47413 25.63256
3 20.30310 19.73159 20.87461
> predict(lm.fit, data.frame(lstat=(c(5,10,15))), interval = "prediction")
       fit       lwr      upr
1 29.80359 17.565675 42.04151
2 25.05335 12.827626 37.27907
3 20.30310  8.077742 32.52846
```

```
plot(Boston$lstat, Boston$medv, xlab= "lstat", ylab ="medv", col="blue")
abline(lm.fit, col = "red")
```

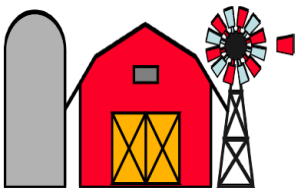
# BDA




You're an economist for the county cooperative. You gather the following data:

| Fertilizer (lb.) | Yield (lb.) |
|---|---|
| 4 | 3.0 |
| 6 | 5.5 |
| 10 | 6.5 |
| 12 | 9.0 |

Find the **least squares line** relating crop yield and fertilizer.




```
> x = c(4,6,10,12)
> y = c(3,5.5,6.5,9)
>
> lm.fit = lm(y ~ x)
> lm.fit

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
       0.80         0.65
```

How to interpret the least squares estimates in this example. $\hat{\beta}_1$ =0.65 implies that the estimate mean crop yeild increases by 65 lbs for each additional lb of fertilizer.

```
> predict(lm.fit, newdata = data.frame(x = c(2.5,5.5,8.5)))
    1     2     3
2.425 4.375 6.325
```

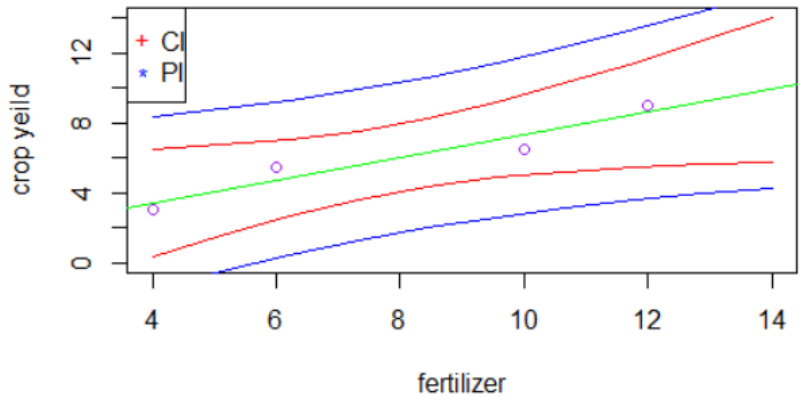
The prediction of the yield when 2.5, 5.5 and 8.5 lb of fertilizer are used gives crop yields of 2.425, 4.375 and 6.325 lb

```
> confint(lm.fit,  level = 0.95)
                  2.5 %   97.5 %
(Intercept) -4.43440296 6.034403
x            0.04151302 1.258487
```

We estimate with 95% confidence that the interval from 0.4lb to 1.25 lb encloses the mean increase of $\hat{\beta}_0$in the crop yield per additional lb in fertilizer

# BDA

```
newdata = data.frame(x=seq(4,14,length.out = 10))
predictconf = predict(lm.fit, newdata, interval  ="confidence", level = 0.95)
predictpred = predict(lm.fit, newdata, interval = "prediction", level = 0.95)

plot(x,y, xlab = 'fertilizer', ylab = 'crop yeild', col ="purple", xlim = c(4,14), ylim =c(0,14))
abline(lm.fit, col = "green")
lines(newdata$x, predictconf[, "lwr"], col="red", pch ="+")
lines(newdata$x, predictconf[, "upr"], col="red", pch ="+")
lines(newdata$x, predictpred[, "lwr"], col="blue", pch ="*")
lines(newdata$x, predictpred[, "upr"], col="blue", pch ="*")
legend("topleft", pch=c("+","*"), col= c("red", "blue"), legend = c("CI", "PI"))
```

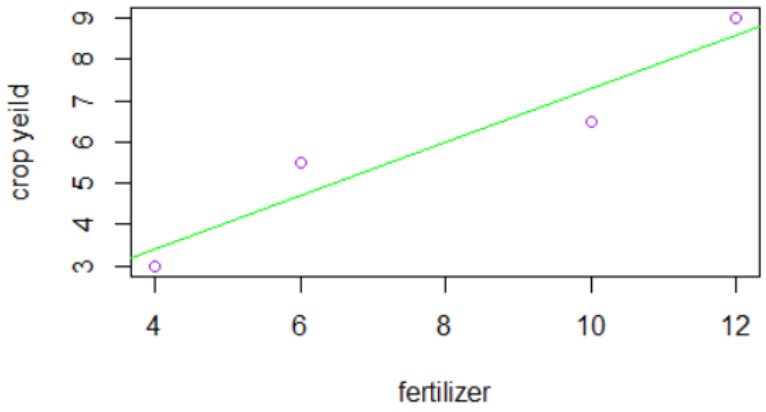



```
> summary(lm.fit)

Call:
lm(formula = y ~ x)

Residuals:
   1    2    3    4
-0.4  0.8 -0.8  0.4

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8000     1.2166   0.658   0.5784
x             0.6500     0.1414   4.596   0.0442 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8944 on 2 degrees of freedom
Multiple R-squared:  0.9135,    Adjusted R-squared:  0.8703

F-statistic: 21.12 on 1 and 2 DF,  p-value: 0.04422
```

- The t-value and p-value for $\hat{\beta}_1$ is 4.596 and 0.0442 respectively, a large t-value and small p-value means there is a relationship between fertiliser and crop yield.
- RSE the residual standard error is 0.8944 this implies that the any prediction in crop yield base on fertilizer would be off by 0.89 lb on average when using least squares.
- The R squared value is the adjusted R-squared is 0.8703 which means that 87.03% of the time the variation in crop yield is explained by the variation in fertilizer.

```
plot(x,y, xlab = 'fertilizer', ylab = 'crop yeild', col ="purple")
abline(lm.fit, col = "green")
```

# BDA




## Logistic Regression

Regression is a quantitative response and classification is a qualitative response. Classification means predicting a categorical response

### Classification techniques

- Logistic regression
- Linear discriminant
- K nearest neighbours
- Generalised addictive models
- Tree-based methods
- Support vector machines

### Simple and multiple logistic regression

### Simple regression

Cannot use a regression line on categorical response because it will predict the wrong value for y. Instead of trying to predict Y lets predict P(Y=1) so output is between 0 and 1. The solution is using a function that gives outputs of 0 and 1, that is logistic function that gives us logistic regression. Regardless of the value of X

$$P(X) = P(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

$$P(X)(1 + e^{\beta_0+\beta_1 X}) = e^{\beta_0+\beta_1 X}$$

$$e^{\beta_0+\beta_1 X} = \frac{P(X)}{1 - P(X)}$$

Making the odds take any value between 0 and infinity, odds are t ally used in horse races since they relate more to the betting strategy. The logistic regression model has a logit (aka log-odds) that is linear in X.

Odds

$$\beta_0 + \beta_1 X = \log(\frac{P(X)}{1 - P(X)})$$

logit

# BDA

Logistic regression is very similar to linear regression, except instead of using least square coefficient estimate we use a method called the maximum likelihood. Formally we maximise the likelihood function.

$$l(\beta_{0,}\, \beta_1) = \prod_{i:y=1} P(X) \prod_{i:y=0} 1 - P(X)$$

Interpreting what $\beta_1$means is not very easy in logistic regression, simply because we are predicting P(Y) and not Y. We know if $\beta_1 = 0$ there is no relationship between our X and Y. We still want to preform hypothesis testing on our $\beta_0$and $\beta_1$ to see if they significantly differ from zero. Use Z instead of a T test doesn't change the way we interpret the p-value!

$$t = \frac{\hat{\beta}_1}{se(\hat{\beta}_1)}$$

Making a prediction given values for $\hat{\beta}_0$and $\hat{\beta}_1$and an X value use the below predicted probability

$$\hat{P}(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

## Multiple logistic regression

Can fit multiple logistic just like linear regression

$$P(X) = \frac{e^{\beta_0+\beta_1 x_1+ \ldots+\beta_n x_n}}{1 + e^{\beta_0+\beta_1 x_1+ \ldots+\beta_n x_n}}$$

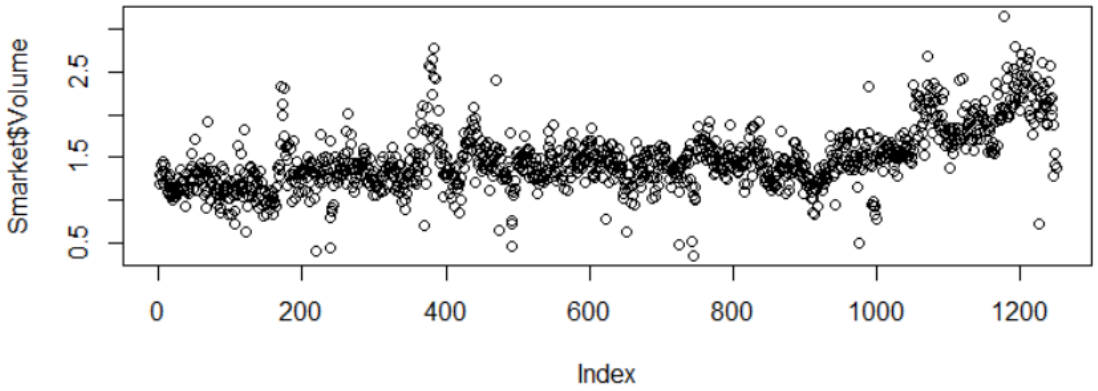
### R Logistic Regression Code

The **cor()** function produces a matrix that contains all the pairwise correlations amount the predictor in the data set. Values must be numeric

```
> library(ISLR)
> cor(Smarket[,-9])
             Year         Lag1         Lag2         Lag3         Lag4         Lag5       Volume        Today
Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718  0.029787995  0.53900647  0.030095229
Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911 -0.005674606  0.04090991 -0.026155045
Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533 -0.003557949 -0.04338321 -0.010250033
Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036 -0.018808338 -0.04182369 -0.002447647
Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000 -0.027083641 -0.04841425 -0.006899527
Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641  1.000000000 -0.02200231 -0.034860083
Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246 -0.022002315  1.00000000  0.014591823
Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527 -0.034860083  0.01459182  1.000000000
```

```
> plot(Smarket$Volume)
```

# BDA




To fit a logistics regression model use glm(), but must state in the argument family = binomial.

```
> glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
  Volume, data = Smarket, family = binomial)

> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.446  -1.203   1.065   1.145   1.326

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

# BDA

Making a prediction from logistic function first calculate the prediction probabilities of glm model use type = response to tell R to give probabilities. Then create a vector of whatever you are trying to predict then use the probability to assign success and compare to original data in a confusion matrix.

```
> glm.probs = predict(glm.fit, type = "response")
> glm.pred <- rep("Down", 1250)
> glm.pred[glm.probs>0.5]="Up"
> table(glm.pred, Smarket$Direction)

glm.pred Down  Up
    Down  145 141
    Up    457 507
```

## R Train and Test Sets

To do this divide data up into train and test sets by creating vectors….

## Assessing Model Accuracy

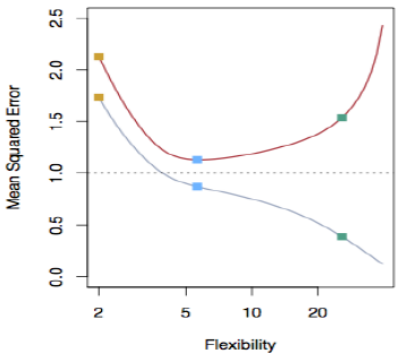
- Measuring the Quality of Fit
  Training Data Builds >> Model >> Test Data Predict
  Regression problem use Residual Sum of Squares (RSS), $\hat{y}_i$is the prediction from the model built on the training data.

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  More common measure of accuracy is the mean squared error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}RSS$$

  Method is chosen to make MSE small on training data, for linear regression MSE is minimised on the least square line this doesn't guarantee the smallest test MSE!
  Note more flexible a method the lower the training MSE will be (explain training date very well) but may make the test MSE higher.




RIGHT
RED: Test MSE
Grey: Training MSE
Dashed: Minimum possibl
MSE (irreducible error)

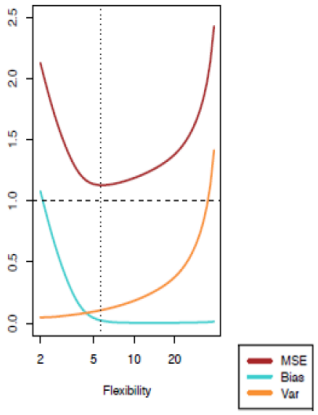
- The Bias-Variance Trade – Off
  - This governs the choice of statistical learning methods.
  - Bias refers to the error introduced by modelling a real life problem by a much simpler problem. Eg linear regression assumes a linear relationship between X and Y but unlikely in real life, the relationship is exactly linear, some bias present
  - The more flexible/complex a method the less bias it will have.

# BDA

- Variance refers to how much the estimate for f would change if you had a different training data set.
- The more flexible a method the more variance.

$$Expected\ Test\ MSE = Bias^2 + Var + \sigma^2$$

- $\sigma^2$ is the Irreducible error.
- The trade off is to find a method which both the variance and squared bias are low, this is very important and a recurring theme in this course.




- The Classification setting
  - Regression use MSE
  - Classification use error rate in confusion matrix

## R MSE on training data set

```
> lm.fit <- lm(mpg ~ horsepower, data = Auto)
> summary(lm.fit)

Call:
lm(formula = mpg ~ horsepower, data = Auto)

Residuals:
     Min       1Q   Median       3Q      Max
-13.5710  -3.2592  -0.3435   2.7630  16.9240

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 39.935861   0.717499   55.66   <2e-16 ***
horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.906 on 390 degrees of freedom
Multiple R-squared:  0.6059,    Adjusted R-squared:  0.6049
F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16

> y_pred <- predict(lm.fit, Auto)
> mse <- mean((Auto$mpg-y_pred)^2)
> mse
[1] 23.94366
```

## Confusion matrix

- In a confusion matrix, elements on the diagonal of the matrix tables represent correctly predicted, while off-diagonal elements represent misclassified.

| Actual Class \ Predicted Class | Class = Yes | Class = No |
|---|---|---|
| Class = Yes | 1 | 2 |
| Class = No | 4 | 3 |

Accuracy = (1+3)/10=0.4
Error rate = (4+2)/10=0.6

## R error rate

```
> glm.fit <- glm(default ~ income + balance,
+                data = Default, family = "binomial")
> summary(glm.fit)

Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8


> summary(default)
  No  Yes
9667  333
> dim(Default)
[1] 10000     4
> probs <- predict(glm.fit, Default, type = "response")
> y_pred = rep("No",10000)
> y_pred[pred>0.5] = "Yes"
> table(y_pred, Default$default)

y_pred   No  Yes
   No  9645  254
   Yes   22   79
> mean(Default$default==y_pred)
[1] 0.9724
```

## Cross Validation CV

Estimate the test error rate by holding out a subset and applying method to held out observations.

### CV on Regression problems

- Validation set approach
  randomly split data in training and test sets
  Advantages:
  - Simple
  - Easy to understand

  Disadvantages:
  - Validation MSE can be highly variable
  - Only a subset of observations are used to fit the model, stats methods perform worse in fewer observations.

# BDA

- Leave-one-out cross validation
  - Tries to address the disadvantages of validation set

  Advantages:
  - Less bias, using all the data
  - Less Variable MSE, always the same only remove 1 obs

  Disadvantages:
  - Computationally intense, have to fit m models!
- K-fold cross validation (Bias-Variance Trade off)

  Advantages:
  - Less computationally intense
  - Just as at stable as LOOCV, k-fold special case of k-fold, k=n
  - K=5 or 10 are magical neither high bias or variance

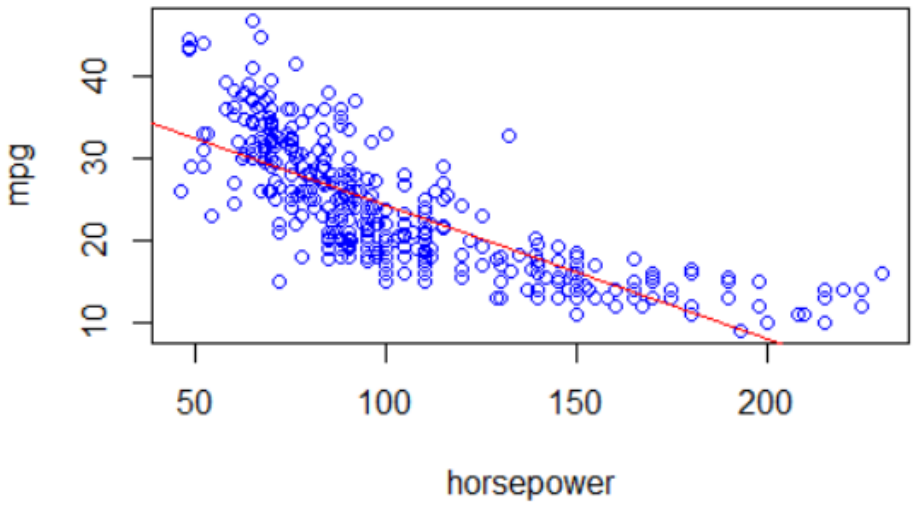
  Disadvantages:
  - Higher variance than LOOCV
  - Slightly higher bias
-

R validation set linear example

```
> set.seed(1)
> dim(Auto)
[1] 392   9
> train <- sample(392, 196)
> lm.fit.train = lm(mpg ~ horsepower, data = Auto, subset = train)
> summary(lm.fit)

Call:
lm(formula = mpg ~ horsepower, data = Auto, subset = train)

Residuals:
    Min      1Q  Median      3Q     Max
-13.698  -3.085  -0.216   2.680  16.770

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 40.340377   1.002269   40.25   <2e-16 ***
horsepower  -0.161701   0.008809  -18.36   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.692 on 194 degrees of freedom
Multiple R-squared:  0.6346,    Adjusted R-squared:  0.6327
F-statistic: 336.9 on 1 and 194 DF,  p-value: < 2.2e-16

> summary(Auto$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   17.00   22.75   23.45   29.00   46.60
> y_pred = predict(lm.fit.train, data = Auto)
> # validation mse
> mean((Auto$mpg - y_pred)[-train]^2)
[1] 94.36907
```

```
> plot(Auto$horsepower, Auto$mpg, xlab = "horsepower", ylab = "mpg", col = "blue")
> abline(lm.fit.train, col = "red")
```

# BDA




There appear to be evidence that the relationship between mpg and horsepower is non-linear suggest trying a quadratic model using R command poly(x, 2)

R validation set non-linear example

```
> dim(Auto)
[1] 392   9
> set.seed(2)
> train = sample(392, 196)
> glm.fit2.train = glm( mpg ~ poly(horsepower, 2), data = Auto, subset = train)
> summary(glm.fit2.train)

Call:
glm(formula = mpg ~ poly(horsepower, 2), data = Auto, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-15.018   -2.211    0.025    2.292   14.492

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            23.663      0.317  74.639  < 2e-16 ***
poly(horsepower, 2)1 -122.624      5.860 -20.927  < 2e-16 ***
poly(horsepower, 2)2   43.380      5.666   7.657 8.95e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 19.49466)

    Null deviance: 13014.2  on 195  degrees of freedom
Residual deviance:  3762.5  on 193  degrees of freedom
AIC: 1143.3

Number of Fisher Scoring iterations: 2
```

```
> probs = predict(glm.fit2.train, Auto)
> summary(Auto$mpg)#continue variable
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   17.00   22.75   23.45   29.00   46.60
> #Validation MSE
> mean((Auto$mpg - y_pred)[-train]^2)
[1] 87.58933
```

The non-liner model has a smaller validation MSE (test error)

Leave-one-out Cross Validation – LOOCV

1. Split the data set of size n into: Training data (size n-1) and Validation data (size 1)

2. Fit the model using the training data
3. Validate model using the validation data
4. Repeat this process n times
5. The MSE for the model is computed as follows:

$$CV_n = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

R LOOCV example

```
> glm.fit = glm(mpg ~ horsepower, data = Auto)
> library(boot)
> cv.err  =cv.glm(Auto, glm.fit)
> cv.err$delta #(raw Cv est, adjusted CV est)
[1] 24.23151 24.23114
```

k-fold cross validation

1. Divide the data into k different parts (k=5 or k=10)
2. Remove first part and fit the model on remaining k-1 parts
3. Compute MSE on first part
4. Repeat this process k-different times taking a different part each time
5. Avaergaing the k different MSE get an estimated validation (test) error rate

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

R LOOCV example

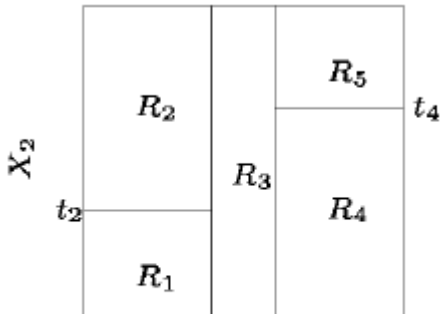
```
> glm.fit = glm(mpg ~ horsepower, data = Auto)
> glm.fit = glm(mpg ~ horsepower, data = Auto)
> library(boot)
> cv.err = cv.glm(Auto, glm.fit, K=10)
> cv.err$delta # raw cv est, adjusted cv est
[1] 24.08327 24.07592
```

Decision Trees

- A flow chart like tree structure

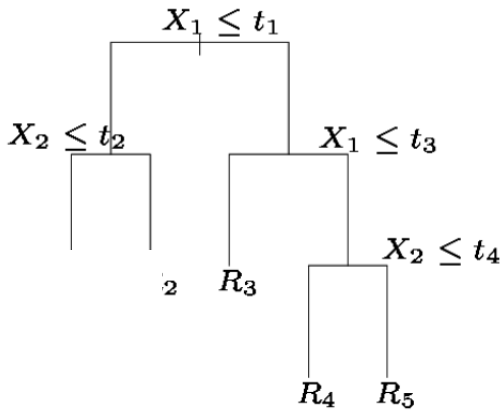
# BDA

- Internal node denotes a test on an attribute
- Branch represents an outcome of test
- Least nodes present class labels or distribution
- Upside down




## Regression Trees

- Predicting a quantitative response
- First split one of X variables into two regions
- Mark regions after all splitting
- Use partitions to present a tree structure.
- Provides a simple way to explain model to a non-expert.

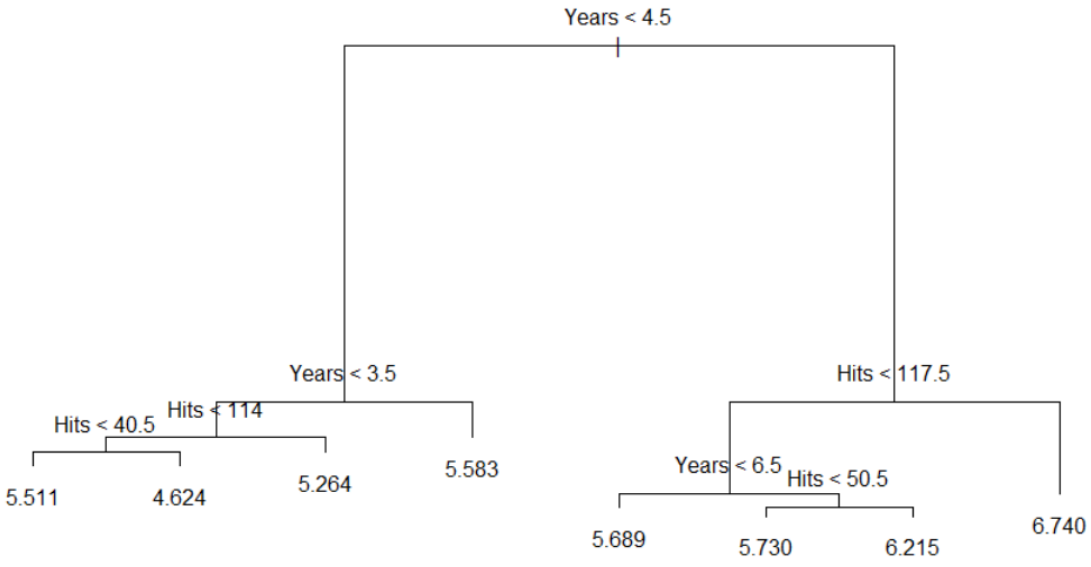



## R regression tree example

```
> library(ISLR)
> dim(Hitters)
[1] 263  20
> Hitters = na.omit(Hitters)
> tree.hitters = tree(log(Salary)~Years+Hits, Hitters)
> summary(tree.hitters)

Regression tree:
tree(formula = log(Salary) ~ Years + Hits, data = Hitters)
Number of terminal nodes:  8
Residual mean deviance:  0.2708 = 69.06 / 255
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.2400 -0.2980 -0.0365  0.0000  0.3233  2.1520
> plot(tree.hitters)
> text(tree.hitters, pretty = 0)#pretty=0 makes category names qualitative
```

Tree structure

Tree structure




## Tree structure

- The regions to use are the best prediction simply the average of all the responses from the training data that fell in a region.
- Split by the variables that results in the lowest MSE on the training data aka smallest MSE.
- Stopping criteria process continues until region has too few observations.

## Tree Pruning

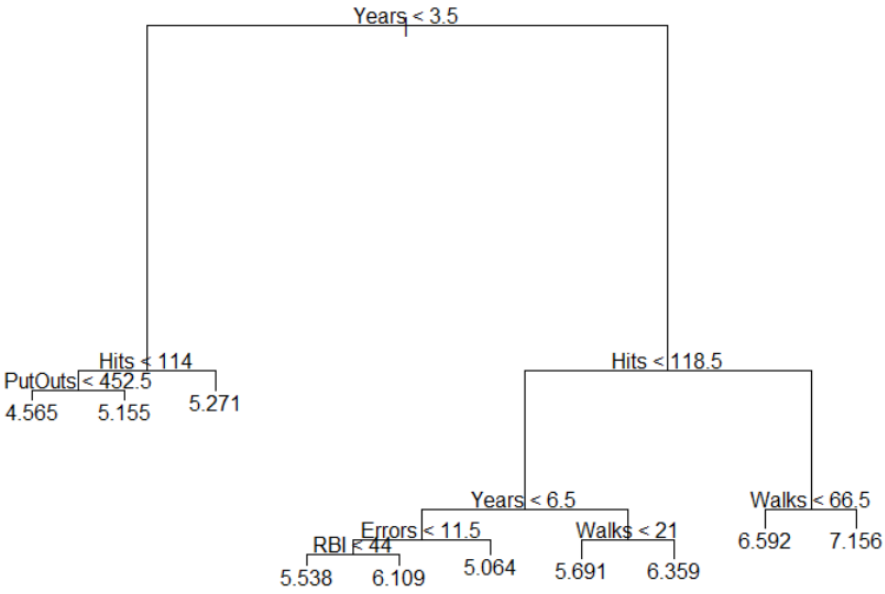
- A large tree with many nodes may tend to over fit the training data.
  Large tree: lower bias, higher variance, worse interpretation
  Small tree: higher bias, lower variance, better interpretation
- Improve accuracy by pruning the tree, aka cutting off some terminal nodes
- Use cross validation to see which tree has the lowest error rate

## R Regression Tree Train example

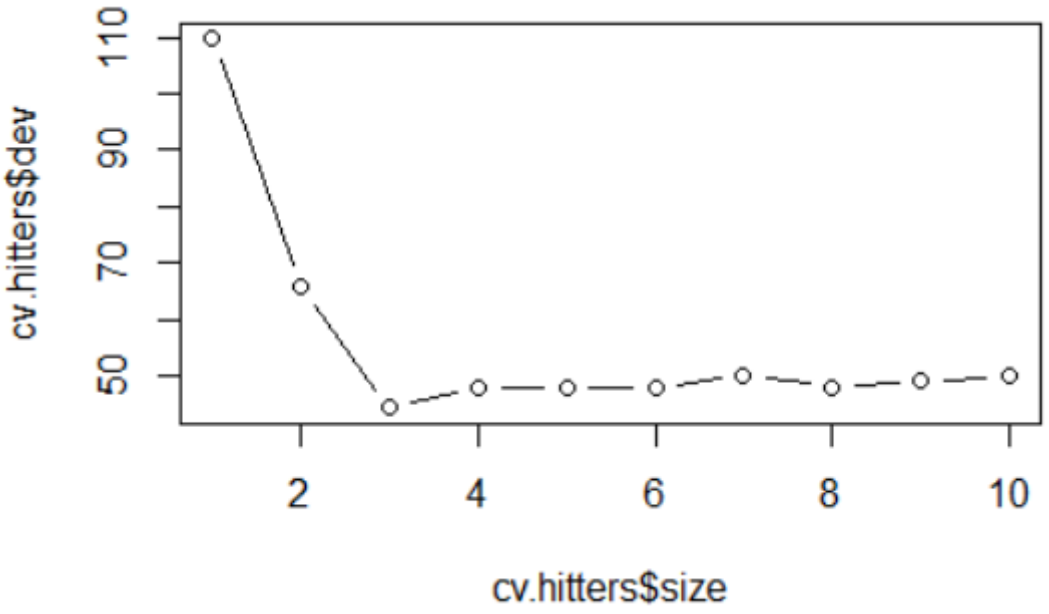
- Fitting
- Plotting
- Pruning
- Estimating error rate

```
> library(ISLR)
> Hitters <- na.omit(Hitters)
> dim(Hitters)
[1] 263  20
> set.seed(2)
> train = sample(263, 132)
> tree.hitters.train = tree(log(Salary) ~ Hits+Runs+RBI+Walks+Years+PutOuts
+                           +AtBat+Assists+Errors, Hitters, subset = train)
> plot(tree.hitters.train)
> text(tree.hitters.train, pretty=0)
```

# BDA

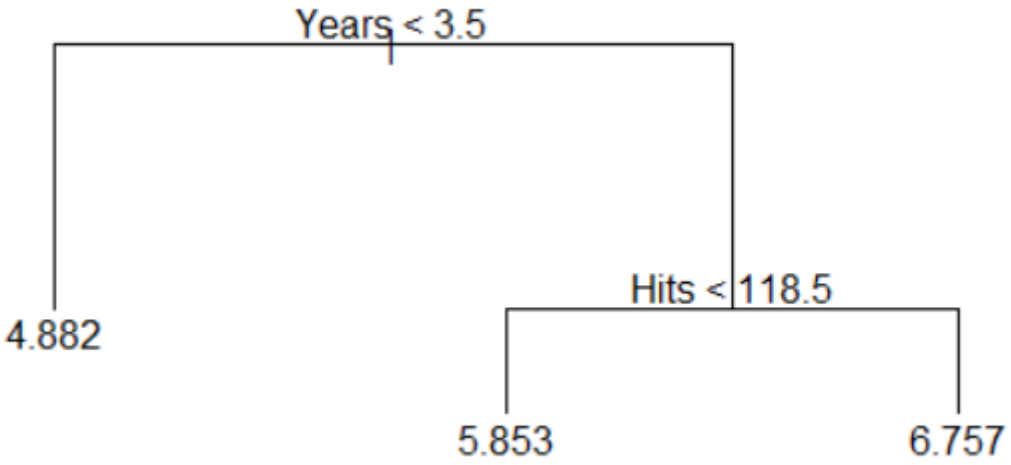



```
> summary(tree.hitters.train)

Regression tree:
tree(formula = log(Salary) ~ Hits + Runs + RBI + Walks + Years +
    PutOuts + AtBat + Assists + Errors, data = Hitters, subset = train)
Variables actually used in tree construction:
[1] "Years"   "Hits"    "PutOuts" "Errors"  "RBI"     "Walks"
Number of terminal nodes:  10
Residual mean deviance:  0.173 = 21.11 / 122
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.15500 -0.24840 -0.02918  0.00000  0.24830  1.66900

> cv.hitters = cv.tree(tree.hitters.train)
> plot(cv.hitters$size, cv.hitters$dev, type='b')
> cv.hitters$dev # deviance, RSS, summed over all nodes
 [1]  49.76245  48.98684  48.01159  49.90206  47.91326  47.91326
 [7]  47.91326  44.18955  65.78981 109.70949
```




Cross validation indicated above that the minimum MSE is when the tree size is 3, so prune tree at three clusters.

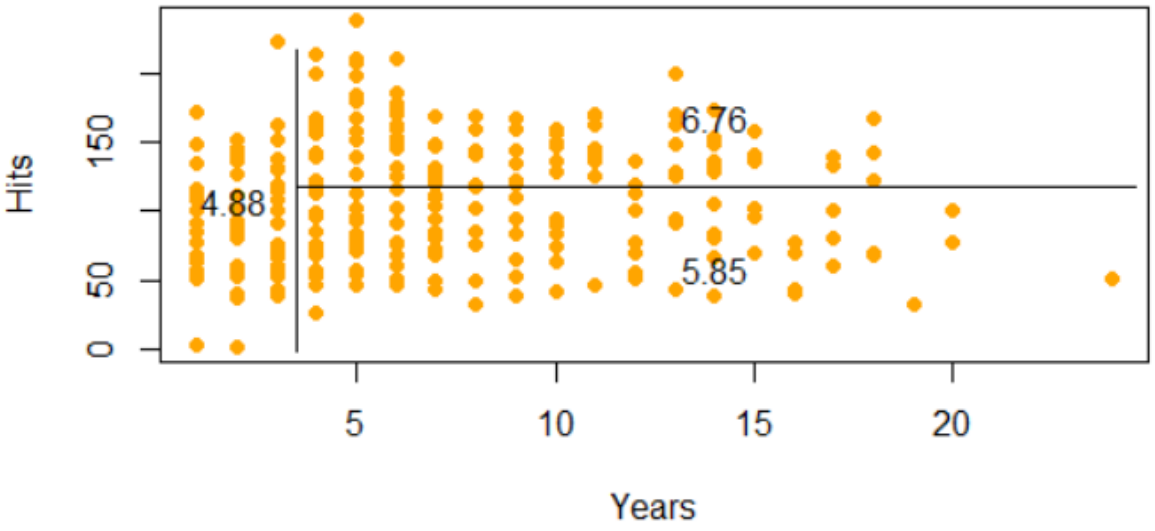
```
> prune.hitters=prune.tree(tree.hitters.train, best=3)
> plot(prune.hitters)
> text(prune.hitters,pretty=0)
```




Visualisation only works for one or two predictors

```
plot(Hitters$Years, Hitters$Hits, col="orange", pch=16,
     xlab="Years", ylab="Hits")
partition.tree(prune.hitters, ordvars = c("Years", "Hits"), add=TRUE)
```

# BDA




Use pruned tree to make predictions

```
> yhat = predict(prune.hitters, newdata=Hitters[-train,])
> Hitters.test = log(Hitters[-train, "Salary"])
> mean((yhat-Hitters.test)^2)#Test MSE pruned
[1] 0.4445136
>
> yhat.unpruned = predict(tree.hitters.train, newdata= Hitters[-train,])
> mean((yhat.unpruned-Hitters.test)^2)#Test MSE unprunes
[1] 0.3746249
```

The MSE for the pruned tree is higher than the unpruned tree, MSE is like variance, the square root of MSE gives the same units like SD. It shows not always the case that pruning is beneficial, unpruned tree has less bias.

## Classification Trees

- When you predict a qualitative response, not a continuous Y a categorical Y, ex a response of YES or NO.
- Each region will predict most common category among the training data within that region.
- The tree is grown in same way as decision trees but MSE doesn't make sense, other measurements like the gini index and cross-entopy but easiest one to think about is the minimise the error rate.

## Confusion matrix

- Elements on the diagonal of the matrix tables represent correctly predicted, while off-diagonal elements represent misclassified.

| Actual Class | Predicted Class: Class = 1 | Predicted Class: Class = 0 |
|---|---|---|
| Class = 1 | $f_{11}$ | $f_{10}$ |
| Class = 0 | $f_{01}$ | $f_{00}$ |

$$\text{Accuracy} = \frac{\#\text{ correct predictions}}{\text{total }\#\text{ of predictions}} = \frac{f_{11}+f_{00}}{f_{11}+f_{10}+f_{01}+f_{00}}$$

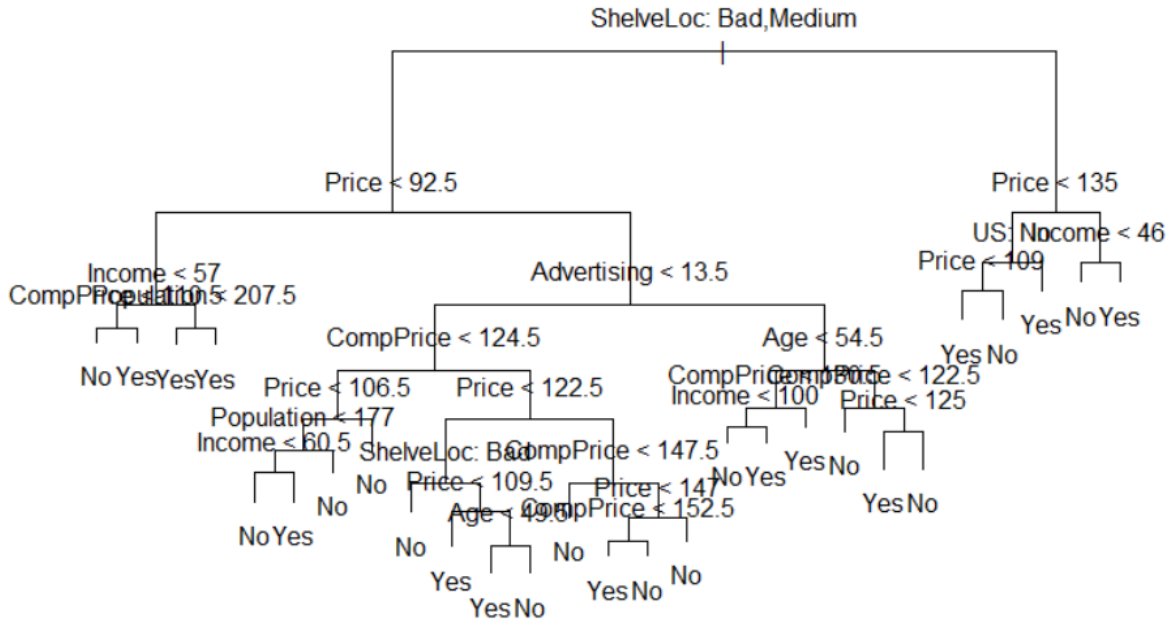
$$\text{Error rate} = \frac{\#\text{ wrong predictions}}{\text{total }\#\text{ of predictions}} = \frac{f_{10}+f_{01}}{f_{11}+f_{10}+f_{01}+f_{00}}$$

# BDA

```
library(tree)
High =ifelse(Carseats$Sales<8, "No", "Yes")
Carseats = data.frame(Carseats,High)
tree.carseats = tree(High~.-Sales, Carseats)
summary(tree.carseats)
```

```
> summary(tree.carseats)

Classification tree:
tree(formula = High ~ . - Sales, data = Carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
[6] "Advertising" "Age"         "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

```
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```




R train and test classification tree

```
library(tree)
High =ifelse(Carseats$Sales<8, "No", "Yes")
Carseats = data.frame(Carseats,High)
tree.carseats = tree(High~.-Sales, Carseats)
summary(tree.carseats)
```

```
> summary(tree.carseats.train)

Classification tree:
tree(formula = High ~ . - Sales, data = Carseats, subset = train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "CompPrice"   "Advertising" "Income"
[6] "US"          "Population"  "Age"
Number of terminal nodes:  20
Residual mean deviance:  0.336 = 60.47 / 180
Misclassification error rate: 0.075 = 15 / 200
```

# BDA

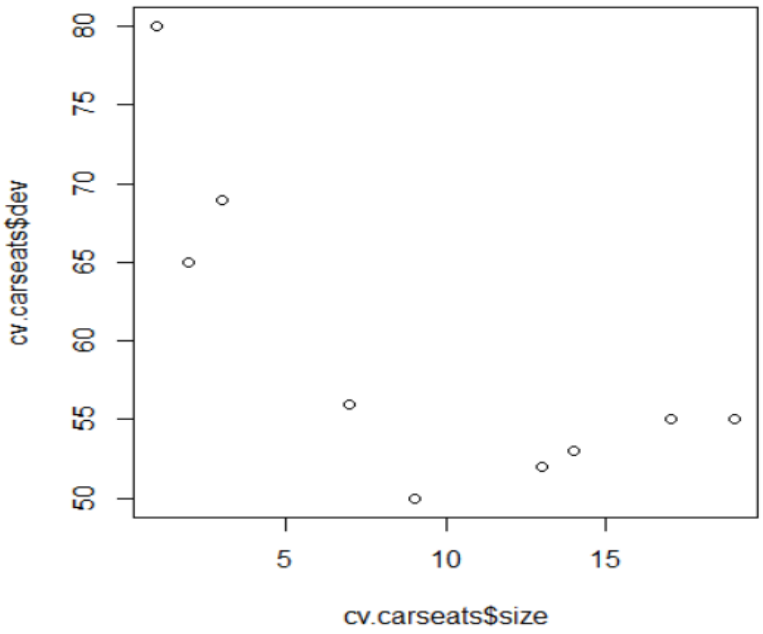
Test error rate

```
> carseats.test = Carseats[-train,]
> tree.pred.test = predict(tree.carseats.train, carseats.test, type="class")
> High.test = High[-train]
> table(tree.pred.test, High.test)
              High.test
tree.pred.test No Yes
           No  82  23
           Yes 44  51
> #Test Error Rate
> 23+44/200
[1] 23.22
```

Training error

```
> High.train = High[train]
> Carseats.train = Carseats[train,]
> tree.pred.train = predict(tree.carseats.train, Carseats.train, type="class")
> table(tree.pred.train, High.train)
               High.train
tree.pred.train  No Yes
            No  100   5
            Yes  10  85
> #Train error rate
> 12+5/200
[1] 12.025
```

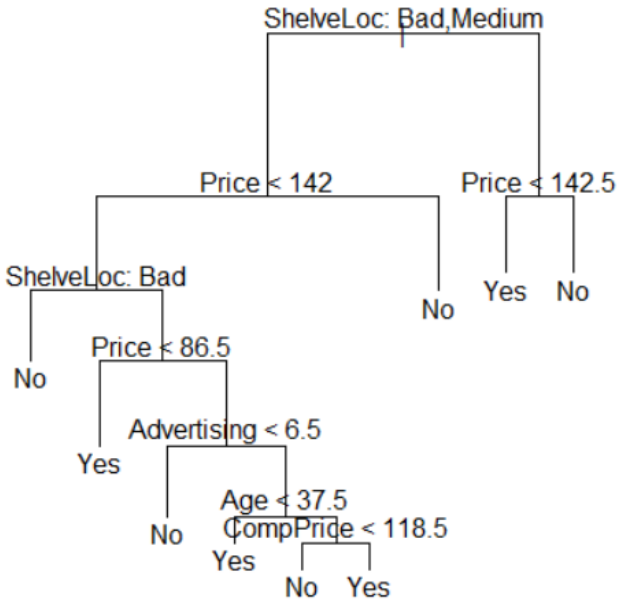
```
> set.seed(3)
> cv.carseats=cv.tree(tree.carseats.train, FUN=prune.misclass)
> plot(cv.carseats$size, cv.carseats$dev)
> cv.carseats
$size
[1] 19 17 14 13  9  7  3  2  1

$dev
[1] 55 55 53 52 50 56 69 65 80
```




```
> prune.carseats = prune.misclass(tree.carseats.train, best =9)
> plot(prune.carseats)
> text(prune.carseats, pretty =0)
```

Cross validation indicated above that the minimum RSE is when the tree size is 9, so prune tree at three clusters.

```
prune.carseats = prune.misclass(tree.carseats.train, best =9)
plot(prune.carseats)
text(prune.carseats, pretty =0)
```




```
> tree.pred = predict(prune.carseats, carseats.test, type="class")
> table(tree.pred, High.test)
         High.test
tree.pred No Yes
      No  94  24
      Yes 22  60
> #Test error rate
> (22+24)/200
[1] 0.23
```

This pruned test error rate is much better than unpruned test error rate. Pruning improved interpretability and classification accuracy. If you increase the value of best, it lowers the classification accuracy.

```
> prune.carseats2 = prune.misclass(tree.carseats.train, best= 15)
> tree.pred2 = predict(prune.carseats2, carseats.test, type="class")
> table(tree.pred2, High.test)
          High.test
tree.pred2 No Yes
       No  86  22
       Yes 30  62
> # New test error rate
> (30+22)/200
[1] 0.26
```

Decision Tree Summary

Tree construction

- at start all training example are at root

- Partition is recursively based on selected attributes

Tree pruning

- Identity and remove branches that reflect noise or outliers

Used on regression and classification problems

Decision tree perform better than linear regression when relationship between predictors is non-linear. If the relationship between the predictors and response is linear, then classical linear regression will perform better.

## Advantage and disadvantages of trees

### Pros:

- Trees can be plotted graphically
- easily interpreted by non- expert
- They work fine on classification and regression problems

### Cons

- Tress don't have same prediction accuracy as some more complicated approaches
- High variance

By aggregating tress can improve prediction performance using methods like bagging, random forests and boosting

Decision trees suffer from high variance, if we split training data into 2 parts and fit decision tress on both parts, results are very different

We like to have models with low variance and since tress are high variance we solve this by using bagging (bootstrap aggregating)

Random sampling without replacement

- Member can only be chosen one, cannot be chosen again

Random sampling with replacement

- Population is replaced everything a member is chosen
- Member can be chosen more than once
- Probability of being chosen same for all members

Resampling observed dataset by sampling with replacement, distinct test are usual there to obtain a measure of variability.

## Bagging

By using plenty of training datasets and averaging results it reduces the variance. How it works

- Generate B different bootstrapped training datasets

- Train the statistical learning method on each of the B training datasets and obtain prediction

Regression (average all predictions from all B trees) Classification (majority vote among all B trees)

## Bagging – Regression

Average the resulting predictions from B regression tress of B bootstrap training datasets. Note not the pruned tress, each tree high variance and low bias but averaging reduces the variance.

```
library(MASS)
library(randomForest)
set.seed(6)
train <- sample(nrow(Boston), nrow(Boston)/2)
bag.boston <- randomForest(medv ~ ., data = Boston,
                           subset = train, mtry = 13,
                           importance = TRUE)
```

```
> bag.boston

Call:
 randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance =
TRUE,      subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

          Mean of squared residuals: 14.82061
                    % Var explained: 82.43
```

Importance means importance of each predictor is accessed

mtry means all predictor should be considered for each split of tree and indicates bagging in random Forest

By default it generates 500 tress

```
boston.test <- Boston[-train, "medv"]
yhat.bag <- predict(bag.boston, newdata = Boston[-train,])
mean((yhat.bag - boston.test)^2)
```

```
> mean((yhat.bag - boston.test)^2)
[1] 12.09593
```

If you reduce the number of tress from default of 500 to 25 it increases the MSE!!!

```
bag.boston2 <- randomForest(medv ~ ., data = Boston,
                           subset = train, mtry = 13,
                           ntree =25,
                           importance = TRUE)

yhat.bag2 <- predict(bag.boston2, newdata= Boston[-train,])
mean((yhat.bag2-boston.test)^2)
```

```
> mean((yhat.bag2-boston.test)^2)
[1] 12.22046
```

# BDA

## Bagging – Classification

Use most common occurring prediction or average probabilities and then predict to the class with the highest probability from B regression tress of B bootstrap training datasets. Note not the pruned tress, each tree high variance and low bias but averaging reduces the variance.

```
library(ISLR)
library(randomForest)
High = ifelse(Carseats$Sales<=8, "No", "Yes")
Carseats <- data.frame(Carseats, High)
set.seed(2)
train = sample(1:nrow(Carseats), 200)
Carseats.test <- Carseats[-train,]
High.test = Carseats[-train, "High"]
bag.carseats <- randomForest(High ~ . -Sales, Carseats,
                             subset = train, mtry = 10)
yhat.carseats <- predict(bag.carseats, newdata=Carseats.test)
table(yhat.carseats, High.test)
```

```
> table(yhat.carseats, High.test)
             High.test
yhat.carseats No Yes
          No  95  17
          Yes 21  67
> (21+17)/200
[1] 0.19
```

Bayes error rate is the lowest possible error rate

## Out-of-Bag Error Estimate

- Straightforward way to estimate the test error of bagged model
- Not need cross validation or validation set approach
- Since bootstrapping involves random subsets of observations build the training data set and the remaining non-selected can be the test data.
- Each tress uses 2/3 of observations and the remaining 1/3 is the out-of-bag (OOB) observations
- Estimate test error using OOB observations is called the OOB error rate
- When number of tress B is sufficient large, OOB error is equivalent to LOOCV error.
- OOB error is very convenient performing bagging on large data sets for which CV would be computationally expensive.
- R code prints out of bagging model error rate an estimate for OOB

## Variable Importance

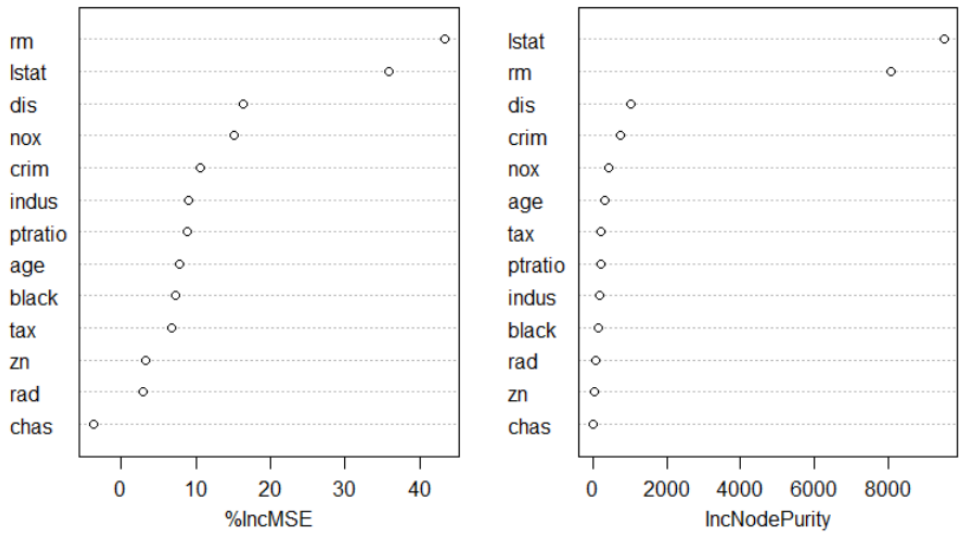
- Bagging improves the accuracy but hard to interpret the model since we have hundreds of trees
- Can get summary of importance of each predictor using Relative Influence Plots

## Relative Influence Plots

- score for each variable
- score represent the decrease in MSE when splitting on that variable
- low score indicate viable is not important

- large score more influence the variable has

```
> importance(bag.boston)
          %IncMSE IncNodePurity
crim    10.574877     737.74377
zn       3.337518      60.61291
indus    9.090736     186.13475
chas    -3.655690      26.78005
nox     15.206971     431.42294
rm      43.342730    8064.64920
age      7.894996     344.46581
dis     16.409667    1016.96534
rad      2.958084      72.62944
tax      6.763824     241.50639
ptratio  8.814332     221.61017
black    7.269107     163.05818
lstat   35.817890    9482.19610
> varImpPlot(bag.boston)
```




## Random Forests

- Very efficient statistical learning method
- Builds on idea of bagging but improved as de-correlates the trees

Build several decision tresses on bootstrapped training sample but each time you split trees using random sample of m predictors from the full set of p predictors.

Usually m = sqrt(p) for random forests

- Using a selection of predictors stops very strong predictors in the data set will always be used for first split.
- Using all parameters will mean all bagged trees look the same and so all predictors for bagged will be correlated!
- Averaging highly correlated quantifies doesn't lead to large variance reduction and so random forest "de-correlates" the bagged tress leading to more reduction in variance
- Build a random forest in the same way as a bagged model but will let predictors aka **mtry** in R.

# BDA

```
library(MASS)
library(randomForest)
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/2)
rf.boston <- randomForest(medv ~ . , data = Boston,
                          subset=train, mtry = 6,
                          importance = TRUE)
yhat.rf = predict(rf.boston, newdata=Boston[-train,])
boston.test = Boston[-train, "medv"]
mean((yhat.rf-boston.test)^2)
```

```
> mean((yhat.rf-boston.test)^2)
[1] 11.53109
```

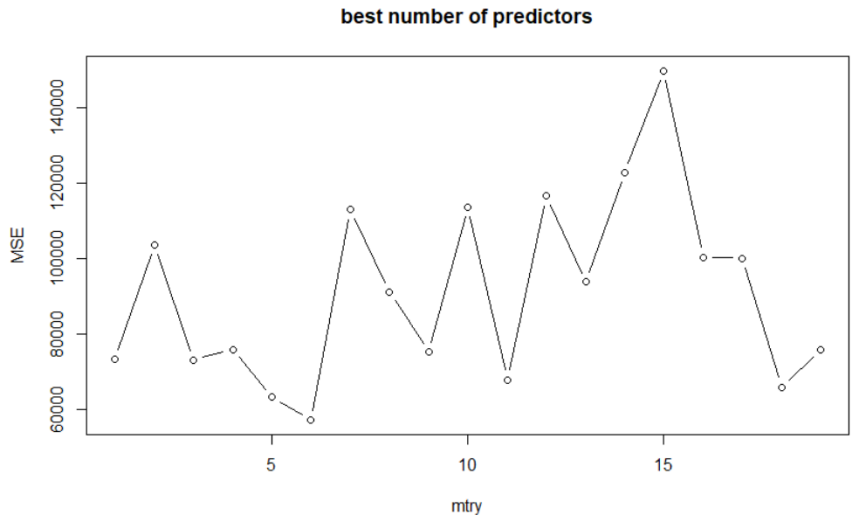
Notice this MSE is smaller than the bagged model

```
> mean((yhat.bag - boston.test)^2)
[1] 12.09593
```

Tree Summary

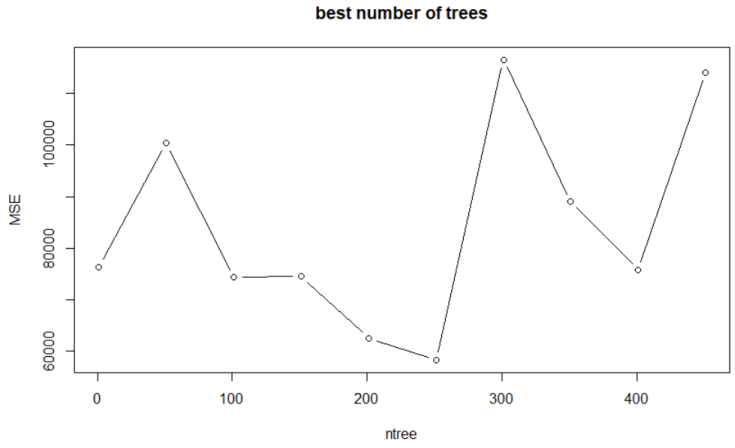
- Decision tree: Validation set approach, only uses half the data to build model and has high variance
- Bagging close to LOOCV, uses all observation with random samples
- Random Forest close K-fold CV, decorrelates training sets, reduces variance, special case of bagging like LOOCV is a special case of K-fold CV

```
MSE = rep(1:19)

for (i in 1:19){
set.seed(i)
train <- sample(1:nrow(Hitters), nrow(Hitters)/2 )
bag.hitters <- randomForest(Salary ~ ., data = Hitters,
                            subset = train, mrty = i,
                            importance = TRUE)
test.bagged <- Hitters[-train, "Salary"]
yhat.bag <- predict(bag.hitters, newdata = Hitters[-train,])
MSE[i] <- mean((yhat.bag-test.bagged)^2)
}

plot(rep(1:19), MSE, xlab="mtry", ylab="MSE", type="b",
     main="best number of predictors")
```



# BDA

```
x <-seq(1,500, by= 50)
MSE2 = rep(1:10)

for (i in 1:10){
  set.seed(i)
  train <- sample(1:nrow(Hitters), nrow(Hitters)/2 )
  for (j in x) {
  bag.hitters <- randomForest(Salary ~ ., data = Hitters,
                              subset = train, mrty = 19, ntree=j,
                              importance = TRUE)
  test.bagged <- Hitters[-train, "Salary"]
  yhat.bag <- predict(bag.hitters, newdata = Hitters[-train,])
  MSE2[i] <- mean((yhat.bag-test.bagged)^2)
}}

plot(x, MSE2, xlab="ntree", ylab="MSE",type="b",
     main="best number of trees")
```




## SVM – Support Vector Machines

One of the best “out of box” classifiers

## Classifiers so far

Logistic regression

Decision trees

Bagging and random forests

## Maximal Margin Classifier

Given two predictors, two-class classification can be linearly separable or linearly inseparable

- linearly separable
  Find a straight line that gives the biggest separation between the classes
  basic idea of maximal margin classifier
- non-linear
  support vector classifiers are fairly easy however it only allows for linear decision boundary so may not be all that powerful but support vector machines can

# BDA

## Maximal Margin Line

- Margin = the minimum (perpendicular distance from all the observation to the separation line.
- Maximal margin line, the line for which the margin is largest
- Use maximal margin line to classify test observation (classification of a point depends on which side a margin falls on)

## Support Vectors

- Support the maximal margin line
- If theses support vector are moved, then so will the margin line
- The maximal margin line depends only on support vectors

## Maximal Margin Classifier

- Works well on two predictors, a line
- Three predictors you need a plane that produces the largest separation between classes.
- More than three dimensions hard to visualise a plane but still exists called hyper-planes
- Looking for maximal margin hyper-planes as maximal classifiers

## Reasons why Maximal Margins Classifiers are not ideal

- Maximal margin hyperplanes may not exist (Linearity inseparable classes) and so not possible to find a hyper-plane that perfectly separates two classes, always will be at least two points on wrong side
- Even if the maximal hyperplanes exist they are extremely sensitive to a change in a single observation, easy to overfit

## Support Vector Classifiers SVC

- SVCs are based on a hyperplane that does not perfectly separate the two classes in the interest of
    - greater robustness of individual observations
    - better classification of most of the training observations
- Soft margin allow some observations to be on incorrect side of the margin or even the incorrect side of the hyperplane.

### Cost of Violation

- Cost is a tuning parameter and is generally chosen via cross validation
- Cost is important it determines the extent to which the model underfits or overfits the data

### Large Cost

- Margin will be narrow

- Few support vectors involved in determining the hyperplane
- Classifier highly fits to the data
- Low bias and high variance

### Small Cost

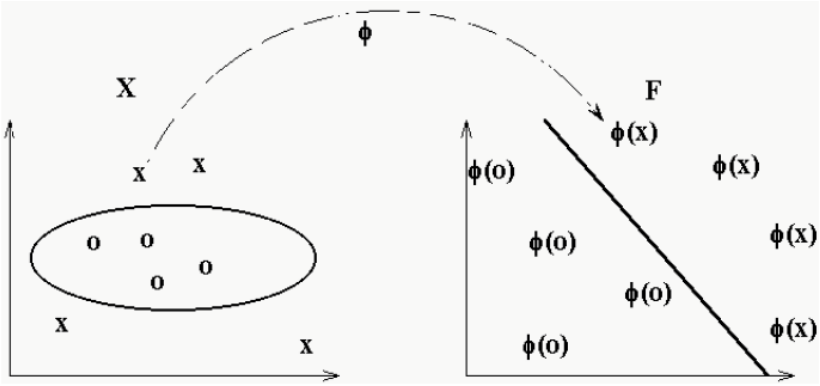
- The margin will be wide
- Many support vectors involved in determining the hyperplane
- Classifier fits the data less hard
- High bias and low variance

### R code for SVC

```
library(e1071)
x = matrix(rnorm(20*2), ncol=2)
y = sample(c(-1,1), 20, rep=TRUE)
dat = data.frame(x,y=as.factor(y))
train = sample(20,10)
svm.fit = svm(y ~ ., data=dat, kernel ="linear",cost= 10,
              scale = FALSE)
summary(svm.fit)
#checking for best cost using tune()
tune.out = tune(svm, y ~ ., data=dat, kernel ="linear",
                ranges=list(cost=c(0.01, 0.1, 1, 10)))
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
# plot values and predicting and using confusion table
plot(x,y)
ypred = predict(bestmod, dat)
table(predicted=ypred, test=dat$y)
```

### Support Vector Machine

SVM maps data into high-dimensional feature space, including non-linear features, then uses linear classifiers




In the original feature space:
Polynomial boundary

In the high-dimensional feature space:
Linear boundary

# BDA

- Transform the original feature (Quadratic function) in the high-dimensional space (linear function).
- Use the transformation or basis to obtain optimal non-linear separating hyperplane.
- It's quite complicated the basis approach so instead we choose a kernel function which takes place of basis
- Common kernel functions
    - Linear
    - Polynomial
    - Radial Basis Function
    - Sigmoid
- SVM produce a non-linear decision boundary and lower test error rate than SVC

## R code for SVC

```
library(e1071)
set.seed(1)
x = matrix(rnorm(200*2), ncol=2)
x[1:100,] = x[1:100,] + 2
x[101:150,] = x[101:150,] -2
y = c(rep(1,150),rep(2,50))
dat = data.frame(x=x, y=as.factor(y))
set.seed(2)
train = sample(200,100)
svm.fit = svm(y ~ . , data = dat, cost = 10, kernal = "linear")
svm.fit2 = svm( y ~ . , data = dat, cost = 10, kernal = "polynomial",
                degree = 3)
svm.fit3 = svm( y ~ . , data = dat, cost = 10, kernal = "radial",
               gamma = 1)
summary(svm.fit)
summary(svm.fit2)
summary(svm.fit3)
```

## SVM linear = SVC

```
> summary(svm.fit)

Call:
svm(formula = y ~ ., data = dat, cost = 10,
    kernal = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10
      gamma:  0.5

Number of Support Vectors:  52

 ( 26 26 )


Number of Classes:  2

Levels:
 1 2
```

## SVM for polynomial decision boundary

```
> summary(svm.fit2)

Call:
svm(formula = y ~ ., data = dat, kernal = "polynomial",
    cost = 10, degree = 3)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10
      gamma:  0.5

Number of Support Vectors:  52

 ( 26 26 )


Number of Classes:  2

Levels:
 1 2
```

## SVM for radial decision boundary

```
> summary(svm.fit3)

Call:
svm(formula = y ~ ., data = dat, cost = 10,
    kernal = "radial", gamma = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10
      gamma:  1

Number of Support Vectors:  52

 ( 26 26 )


Number of Classes:  2

Levels:
 1 2
```

the 3 outputs from the R code for

- SVM linear = SVC
- SVM for polynomial decision boundary
- SVM for radial decision boundary
- Train and test sets can also be used to calculate the misclassification rate from a confusion matrix.
- Can also use tune() to check the best gamma value

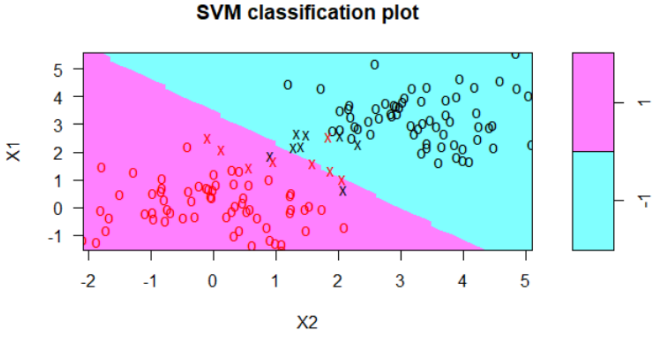
# BDA

```
set.seed(3)
tune.rad = tune(svm, y ~ . ,data = dat[train,], kernel = "radial",
                ranges = list(cost =c(0.01, 0.1, 1, 10)),
                gamma = c(0.1, 1, 2, 3, 4))
summary(tune.rad)
tune.rad$best.model

yhat = predict(tune.rad$best.model, data = dat[-train,])
table(predicted=yhat, observed=dat[train,"y"])
#misscalsifed rate
(8+3)/100
#11%
```

```
> summary(tune.rad)

Error estimation of 'svm' using 10-fold cross validation: 0.29

> tune.rad$best.model

Call:
best.tune(method = svm, train.x = y ~ ., data = dat[train, ],
    ranges = list(cost = c(0.01, 0.1, 1, 10)), kernel = "radial",
    gamma = c(0.1, 1, 2, 3, 4))


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10
      gamma:  0.1 1 2 3 4

Number of Support Vectors:  38

> table(predicted=yhat, observed=dat[train,"y"])
         observed
predicted  1  2
        1 65  3
        2  8 24
> #misscalsifed rate
> (8+3)/100
[1] 0.11
```

Plotting SVM

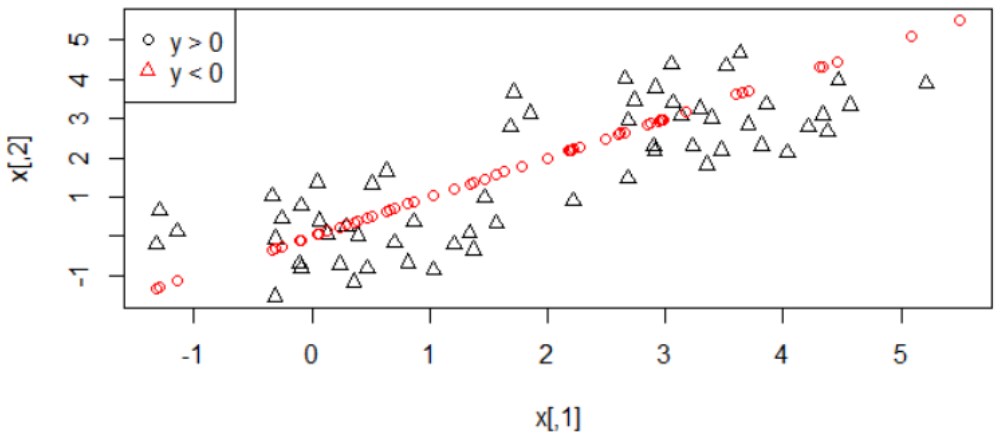
```
> plot(svm.fit, dat[train,])
```



# BDA

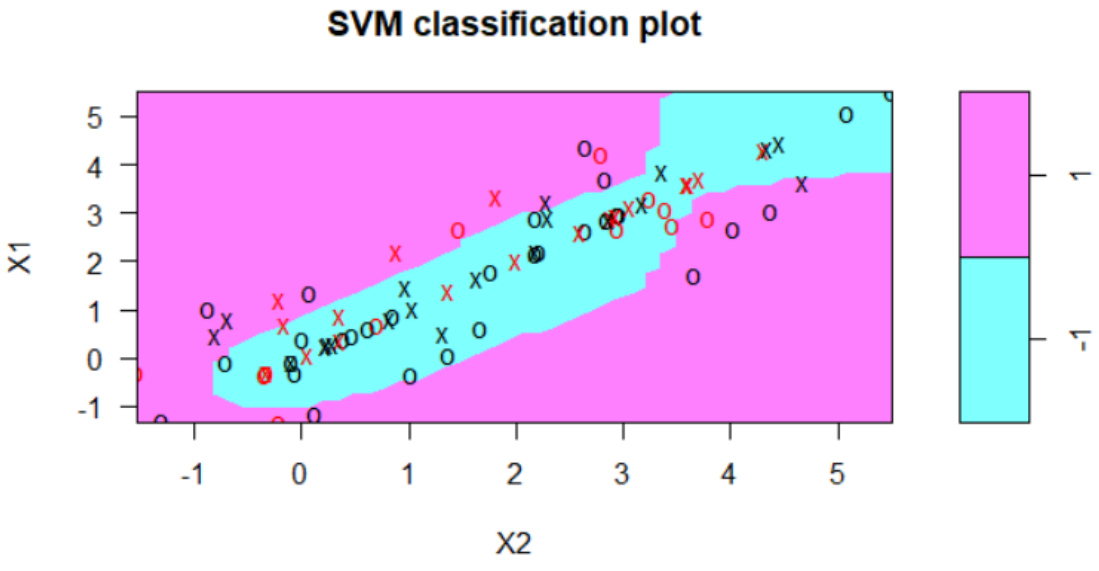
Generating non-linearly separable data

```
set.seed(300)
x <- matrix(rnorm(30*2, mean=0, sd=1), nrow=30, ncol=2)
set.seed(300)
z <- matrix(rnorm(30*2, mean=3, sd=1), nrow=30, ncol=2)
x1 <- matrix(rnorm(30, mean=0, sd=1) + 3, nrow=30, ncol=2)
set.seed(300)
z1 <- matrix(rnorm(30, mean=3, sd=1) - 3, nrow=30, ncol=2)
x = rbind(x, z, x1, z1)
y <- c(rep(1, 60), rep(-1, 60))
dat = data.frame(x,y=as.factor(y))

plot(x, col=ifelse(y>0, 1, 2), pch =ifelse(y<0, 1, 2) )
legend("topleft", c("y > 0", "y < 0"), col=c(1,2), pch = 1:2)
```




Fitting a SVM radial kernel with gamma 1 and cost 1to the non-linearly separable data

```
set.seed(300)
train <- sample(120, 120*70)
library(e1071)
svm.fit = svm(y ~ . , data = dat[train,], cost = 1, kernel = "radial", gamma = 1)
summary(svm.fit)
plot(svm.fit, dat[train,])
```



# BDA

Computing the test error rate

```
x.test = x[-train]
y.test = y[-train]
yhat <- predict(svm.fit, x.test)
table(yhat, y.test)
mean(yhat!=y.test)
```

```
> table(yhat, y.test)
    y.test
yhat -1  1
  -1 16  9
  1   2 12
> mean(yhat!=y.test)
[1] 0.2820513
```

## Unsupervised learning

Unsupervised learning hard to assess and more subjective

- Clustering
  - K-means
  - Hierarchical clustering
- Hidden Markov Models
- Feature extraction for dimension reduction

### Clustering

- Techniques for finding subgroups or cluster in a dataset
- Good clustering the observation within group are similar
- Groups are very different

### K-Means Clustering

- Must specify the desired number of **K** groups
- Algorithm assigns each observation to exactly one group, groups are mutually exclusive
- Randomly partition data into K clusters and calculate cluster centroid (mean of observations)
- Objective to minimise the within-cluster-variation, have obs as similar as possible
- This is done by minimising the pairwise squared Euclidean distances between each observation and each cluster centroid
- Assign each observation to the closet centroid using Euclidean distance
- Local optimums rather than global optimums, important to run algorithm multiple times with random starting points to get a good solution.

```
set.seed(1)
kmn <- kmeans(x, 5, nstart= 20)
```

# BDA

```
> kmn$cluster
 [1] 2 2 1 5 4 2 1 5 2 1 2 1 3 5 4 4 2 3 5 5 4 4 1 5 3 5 1 2 1 1
> kmn$totss
[1] 38.67767
> kmn$tot.withinss
[1] 7.323341
> kmn$totss - kmn$tot.withinss
[1] 31.35433
> kmn$betweenss
[1] 31.35433
```

## Hierarchical Clustering

- Don't need to state how many clusters
- Two types of algorithms, Agglomerative (bottom-up) Divisive(top-down)

## Dendrogram

- Agglomerative algorithm
- Each observation as a separate cluster (n clusters)
- Calculate a measure of dissimilarity between all points/clusters
- Fuse two clusters together so there are n-1 clusters
- Fuse next two clusters together so there are n-1 clusters
- Continue until there is only 1 cluster
- To choose the clusters by drawing a line on genogram
- Is a tree where each node is a group and each leaf is an observation

## Linkage

Defines the dissimilarity between clusters

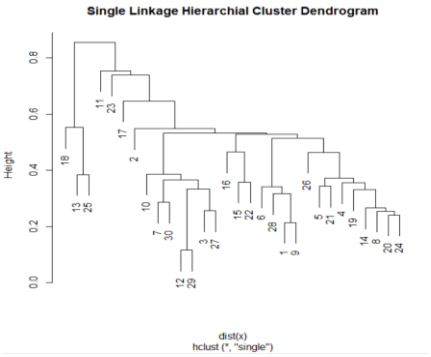
- Single Linkage the dissimilarity between two clusters is the smallest dissimilarity between two observations in opposite clusters (closest pair)
- Complete Linkage the dissimilarity between two clusters is the largest dissimilarity between two observations in opposite clusters (furthest pair)
- Average Linkage the dissimilarity between two clusters is the average over all points in opposite clusters
- Centroid Linkage the dissimilarity between two clusters is the distance between cluster centroids

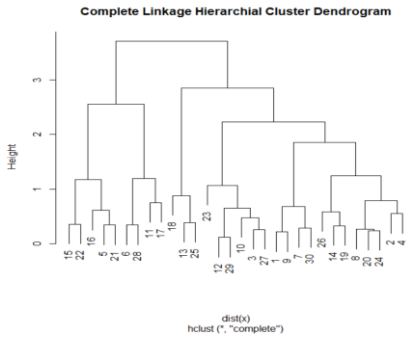
## Shortcomings

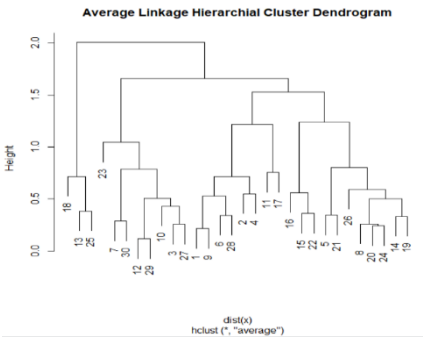
- Single linkage suffers from chaining, clusters too spread out and not compact enough
- Complete linkage suffers from crowding, clusters are compact but not far enough apart. It can break large cluster, but small clusters are merged into big ones
- Average linkage tries to find a balance, clusters compact and far apart but hard to interpret
- Centroid Linkage produces dendrogram with inversions, parent lower than child

```
hclust.single = hclust(dist(x), method="single")
hclust.complete = hclust(dist(x), method = "complete")
hclust.average = hclust(dist(x), method = "average")
```

# BDA




```
plot(hclust.single, main="Single Linkage Hierarchial Cluster Dendrogram")
plot(hclust.complete, main="Complete Linkage Hierarchial Cluster Dendrogram")
plot(hclust.average, main="Average Linkage Hierarchial Cluster Dendrogram")
```

Correlation based distance could be used instead. Cuts matrix into triangle and return correlation

## Scaling Variables

If independent variables are measured on different scales, might want to scale the variable to have zero mean and standard deviation one. In this way each variable will in effect be given equal importance in the hierarchal/k-means clustering.

Standardized variables have mean zero and standard deviation one

```
scale(x)
hclust.single = hclust(dist(x), method="single")
```

This can be done by subtracting the variable mean from each variable value and divided my standard deviation of that variable

## PCA – Principal Component Analysis

Dimension reduction and Unsupervised learning

Could plot two-dimensional scatter plots all day but not possible to see all feature and most likely most are uninformative and only small fraction

- PCA is a technique to simplify datasets and use components to understand the data
- Can be used to reduce number of features/dimensions
- A tool for data visualisation, use biplot

1. Uses covariance between the variables in a covariance matrix
2. Compute set of pairs of eigenvector and eigenvalues
3. Sort them the eigenvector in descending order
4. The first eigenvector is the first principal component etc
5. PVE (portion of variance explained) by each component = PC Var/Total Variance
6. Use smallest number of PC, look for elbow in PVE and Number of PCs

How many PCs to use, always use the smallest number or eyeballing the scree plot (PC Vs PVE by each component) and look for an elbow