Statistics Templates

Go Ito

August 9th, 2019

Contents

Package Installation 3

Generic library to use 3

Useful functions 3

Data Reading 4
- Built in data . . . 4
- .txt file . . . 4
- .csv file . . . 4
- .xls / .xlsx file . . . 5
- .jason file . . . 5
- SQL . . . 5

Exploratory Analysis 6
- Summary . . . 6
- Histogram / Distributions . . . 6
- Boxplots . . . 9
- Plots of two variables . . . 9
- Interaction Plots . . . 9

Data Analysis, Prediction, and Classification 11
- Simple Linear Regression . . . 11
  - Assumption Check . . . 11
  - Statistical Inference . . . 12
  - ANOVA SLR . . . 13
  - Transformation . . . 15
  - Weighted Least Square . . . 17
- Multiple Linear Regression . . . 18
  - Assumptions . . . 18
  - Statistical Inference . . . 20
  - Model Diagnosis : Added Variable Plot . . . 21
  - ANOVA MLR . . . 22
  - Nested Model and Partial F-test . . . 22
  - ANCOVA (Analysis of Covariance) . . . 23
  - Polynomial Regression . . . 25
  - Model Selection Criteria . . . 30
  - Subset Selection: Best Subset Model . . . 30
  - Subset Selection: Stepwise Regression . . . 34
  - Shrinkage: Regularization . . . 39
  - Dimension Reduction: Principle Component Analysis . . . 44
  - Spline . . . 47
  - Generalized Additive Models . . . 50
  - Logistic Regression . . . 52
- Bayes Classifier . . . 56
- K Nearest Neighbor . . . 57
- Linear Descriminant Analysis . . . 60
- Quadratic Discriminant Analysis . . . 66

Support Vector Machine . . . 67
K-Means Clustering . . . 68
Kernelized Clustering . . . 69
EM Type Algorithm . . . 70
Decision Trees . . . 71
Random Forest . . . 74
Neural Network . . . 79
Comparing all classification methods . . . 80

Experimental Design **81**

Computational Techniques **82**
Standarization / Normalization . . . 82
K-Fold Cross Validation . . . 83
Bootstrap . . . 85

R Features **86**
Basics . . . 86
Functions . . . 86
`I()` and `poly()` for polynomial regression . . . 87
Normalization and Standarization . . . 89
`apply()`, `sapply()`, `tapply()`, `lapply()` . . . 89
Package: `dplyr` . . . 89

Package Installation

```
#install.packages("ggplot2")
#install.packages("dplyr")
#install.packages("tidyverse")
#install.packages("readxl")
#install.packages("alr3")
#install.packages("MASS")
#install.packages("ISLR")
#install.packages("class")
#install.packages("caret")
#install.packages("e1071")
#install.packages("leaps")
#install.packages("boot")
#install.packages("crossval")
#install.packages("resample")
#install.packages("glmnet")
#install.packages("pls")
#install.packages("splines")
#install.packages("gam")
#install.packages("akima")
#install.packages("tree")
#install.packages("randomForest")
```

Generic library to use

```
library(ggplot2)
library(dplyr)
library(tidyverse)
library(readxl)
library(MASS)
library(car)
```

Useful functions

```
traindex = function(df, trainn = nrow(df)-nrow(df)/10){
  n = nrow(df)
  return(sample(1:n,trainn, replace=F))
}
```

Data Reading

Built in data

```r
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```r
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
```

.txt file

```r
cleaning = read.table("cleaning.txt", header=T)
head(cleaning)
```

```
  Case Crews Rooms    StdDev
1    1    16    51 12.000463
2    2    10    37  7.927123
3    3    12    37  7.289910
4    4    16    46 12.000463
5    5    16    45 12.000463
6    6     4    11  4.966555
```

.csv file

```r
Heart = read.csv("Heart.csv", header=T)
head(Heart)
```

```
  X Age Sex    ChestPain RestBP Chol Fbs RestECG MaxHR ExAng Oldpeak Slope
1 1  63   1      typical    145  233   1       2   150     0     2.3     3
2 2  67   1 asymptomatic    160  286   0       2   108     1     1.5     2
3 3  67   1 asymptomatic    120  229   0       2   129     1     2.6     2
4 4  37   1   nonanginal    130  250   0       0   187     0     3.5     3
5 5  41   0   nontypical    130  204   0       2   172     0     1.4     1
6 6  56   1   nontypical    120  236   0       0   178     0     0.8     1
  Ca       Thal AHD
1  0      fixed  No
2  3     normal Yes
3  2 reversable Yes
4  0     normal  No
5  0     normal  No
```

```
6  0     normal  No
```

.xls / .xlsx file

```
library(tidyverse)
library(readxl)
```

```
xlsxdata = read_excel("sample.xlsx")
xlsxdata
```

```
# A tibble: 9 x 4
    Age Gender Weight Height
  <dbl> <chr>   <dbl>  <dbl>
1    14 Male       52    150
2    32 Male       82    167
3    54 Female     64    153
4    12 Male       45    147
5    71 Female     52    145
6    56 Female     65    161
7    23 Female     56    162
8    49 Male       92    186
9    30 Male       75    182
```

.jason file

SQL

```
#library(dbplyr)
#library(RSQLite)
#dir.create("data", showWarnings = FALSE)
#download.file(url = "https://ndownloader.figshare.com/files/2292171",destfile = "data/portal_mammals.s
#mammals = DBI::dbConnect(RSQLite::SQLite(), "data/portal_mammals.sqlite")
```

```
#SQLdata = tbl(mammals, sql("SELECT year, species_id, plot_id FROM surveys"))
#head(SQLdata)
```

Exploratory Analysis

Check List:

- General distribution of the object, widely spread? Skewed? Any outliers?
- Check for NA, NaN, or any other missing values equivalent.

Summary

- Check for the basic summary: mean, median, min, max, frequency.
- For frequency, we prefer thatt all category data have sufficient number of data,
- Check for the correlation among variables.

```
summary(iris)
```

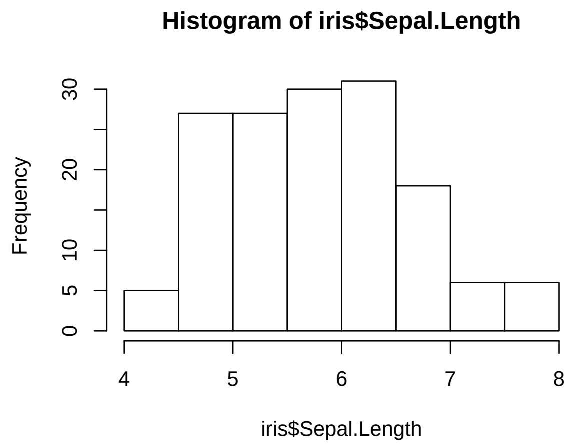
```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

```
cor(iris[,-5])
```

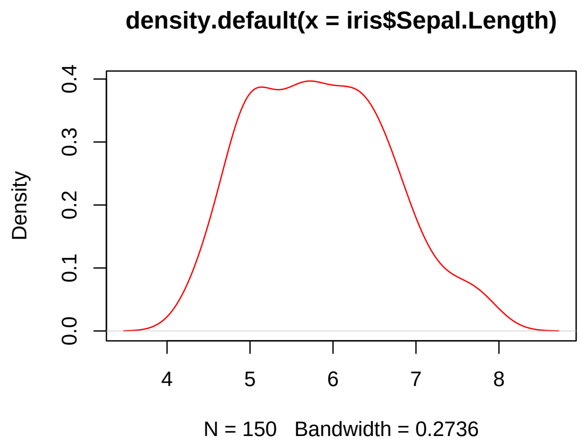
```
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Histogram / Distributions

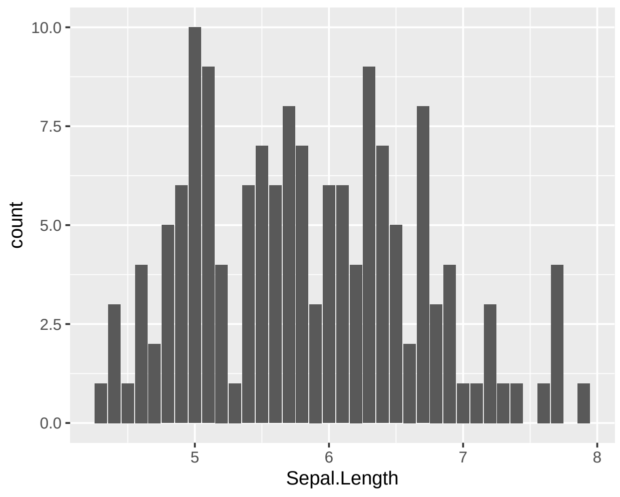
```
# easiest
hist(iris$Sepal.Length)
```



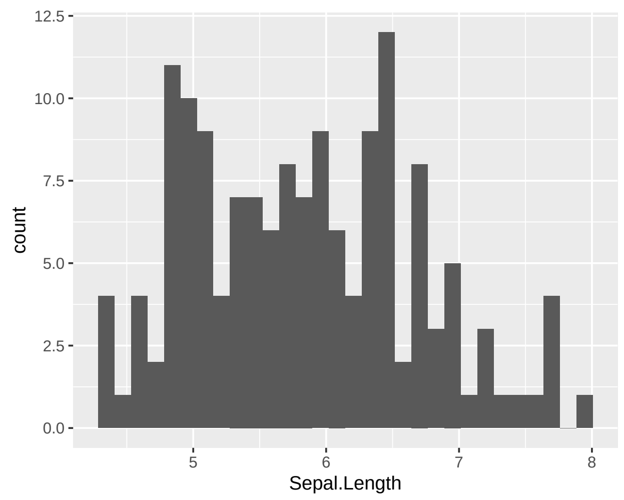
```
#
plot(density(iris$Sepal.Length), col="red")
```



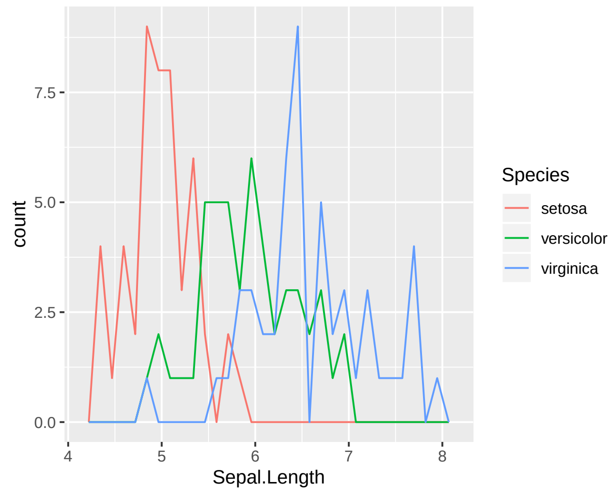
```
# much easier for exploratory analysis
ggplot(data = iris) + geom_bar(mapping = aes(x = Sepal.Length))
```



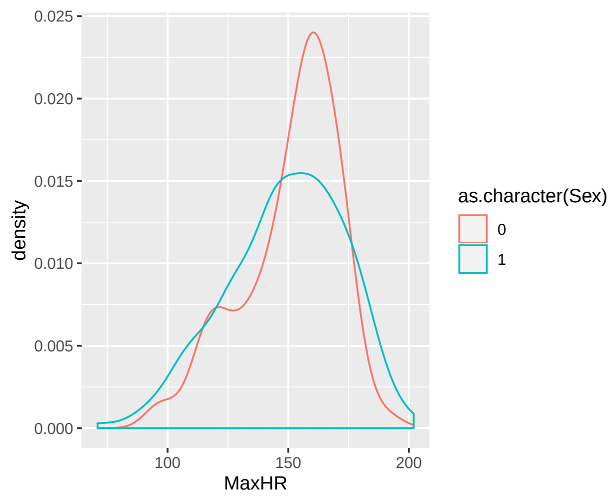
```r
ggplot(data=iris) + geom_histogram(mapping = aes(x=Sepal.Length))
```



```r
ggplot(data=iris, mapping = aes(x=Sepal.Length, colour = Species)) + geom_freqpoly()
```



```r
ggplot(Heart) + geom_density(mapping=aes(x=MaxHR, color=as.character(Sex)))
```

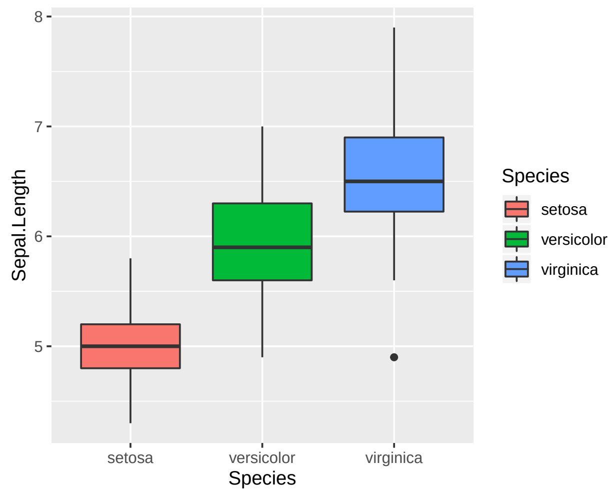


- Which values are most common among which cateogry?
- Which values are rare, or odd? Could it be an outlier, or mis-interpreted?

- Any unusual patters? Can you explain it?
- Why setosa tends to have smaller Sepal.Length than virginica?

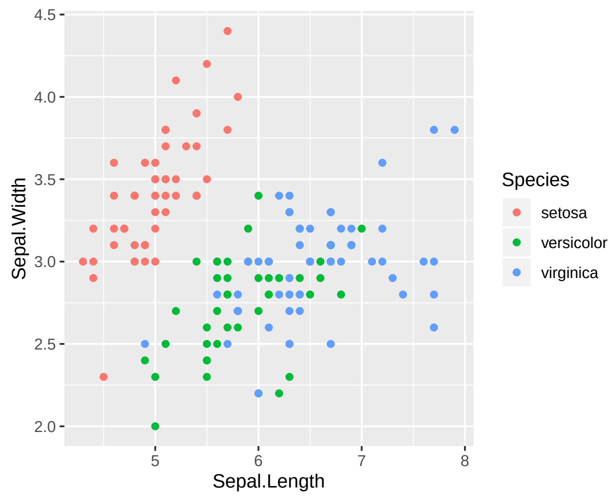
Boxplots

```r
ggplot(data = iris) + geom_boxplot(mapping = aes(x=Species, y=Sepal.Length, fill=Species))
```



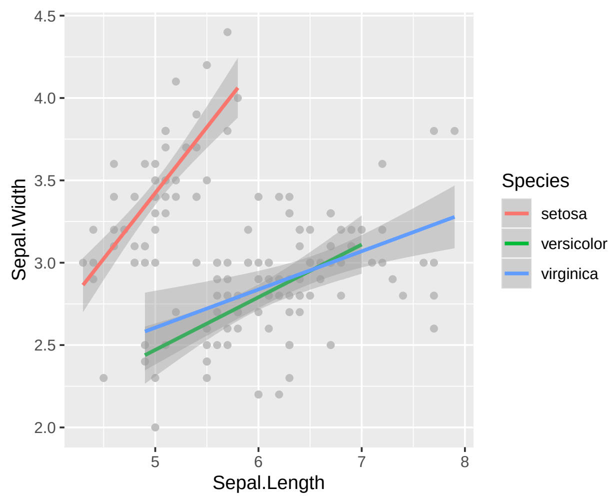
Plots of two variables

```r
ggplot(data=iris) + geom_point(aes(x=Sepal.Length, y=Sepal.Width, colour=Species))
```



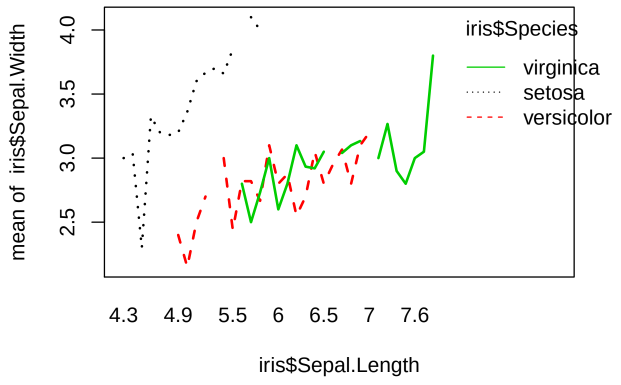
Interaction Plots

```r
ggplot(data=iris) + aes(x = Sepal.Length, y = Sepal.Width, colour = Species) +
  geom_point(color = "grey") +
  geom_smooth(method = "lm")
```



- Often used in time series

```
interaction.plot(x.factor = iris$Sepal.Length, trace.factor = iris$Species, response = iris$Sepal.Width
```



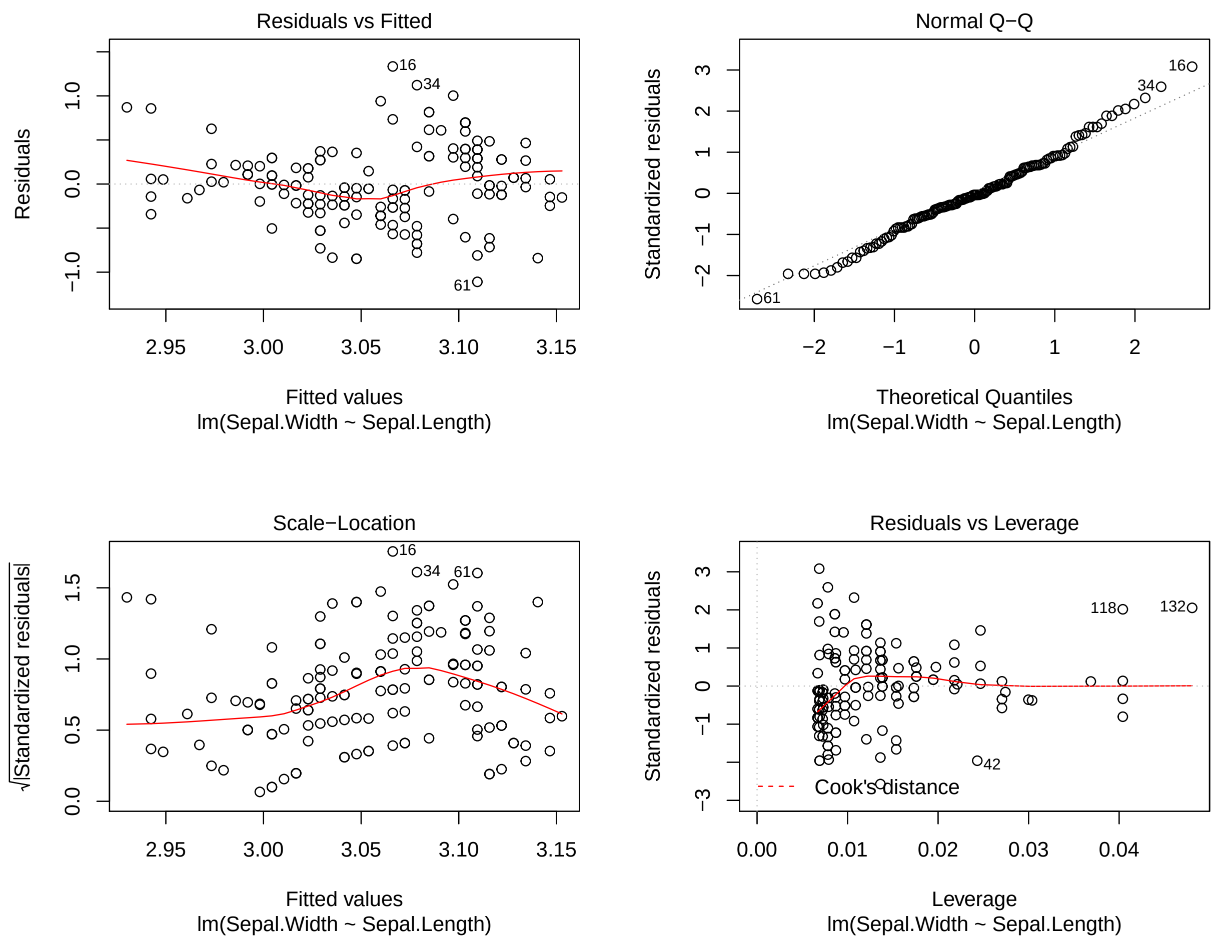
Data Analysis, Prediction, and Classification

Simple Linear Regression

Assumption Check

- 1. Average is 0?
- 2. Standarized residual (more informative when leverage points exist because errors can show const var while residuals don't / how many estimated std deviations any point away from the fitted regression model / if outside -2 to 2, outlier), check for constant variance primarily here!
- 3. Normality (straight line) holds? For each x, see if corresponding y follow normal distributions where mean is fitted line.
- 4. Any outliers, leverage points(influential to fitted model / how predicted y change if removed / bigger than $4/n$), outside Cook's distance ($D_i = \frac{r_i^2}{2} * \frac{h_i i}{1-h_i i}$ / $D_i > 4/(n-2)$)?
- If any of the assumptions violated, any further inferences are invalidated.

```
model1 = lm(Sepal.Width~Sepal.Length, data=iris)
plot(model1)
```



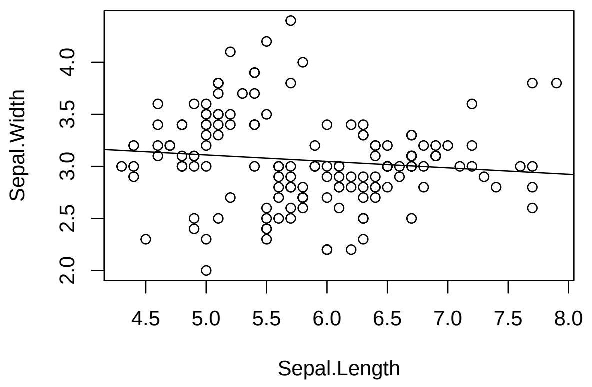
```
head(lm.influence(model1)$hat)
```

```
          1           2           3           4           5           6 
0.012074844 0.015376584 0.019461346 0.021797361 0.013627836 0.008590398 
```

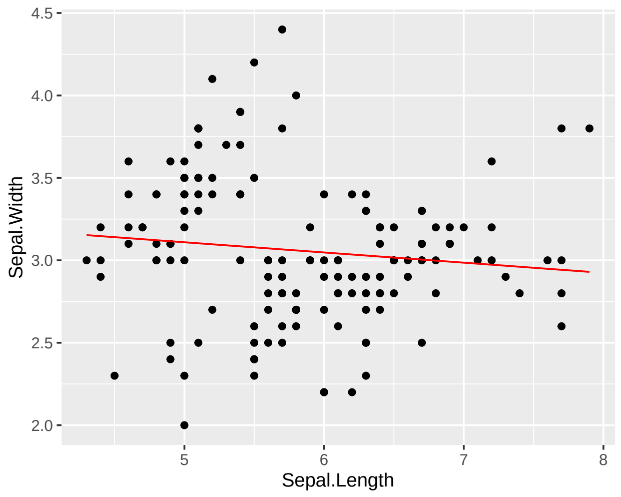
Statistical Inference

- Shape of the plots, linear? quadratic? exponential?

```
plot(Sepal.Width~Sepal.Length, data=iris);abline(model1)
```



```
# Equivalently this, but much more complicated
predicted1 = data.frame(Sepal.Length=iris$Sepal.Length, predicted = predict(model1, iris))
ggplot(data=iris) + geom_point(aes(x=Sepal.Length,y=Sepal.Width))+ geom_line(color='red',data = predict
```



- Coefficients significant? P-value? Standard Error?
- Positive? Negative? Its strength?
- "One unit increase in X results in β_1 much increase in Y"
- $\hat{\beta}_1 = r * \frac{S_Y}{S_X} = \frac{\sum^n (X-\bar{X})(Y-\bar{Y})}{\sum^n (Y-\bar{Y})^2}$
- $\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$
- $e \sim N(0, \sigma^2)$
- $S^2 = \frac{\sum^n (Y-\hat{Y})^2}{n-2}$
- $se(\hat{\beta}_0)^2 = S^2(\frac{1}{n} + \frac{(x^*-\bar{x})^2}{\sum^n (x-\bar{x})^2})$
- $se(\hat{\beta}_1)^2 = \frac{S^2}{\sum^n (X-\bar{X})^2}$

- $SSE = \sum^n (y - \hat{y})^2$
- $SSR = \sum^n (\hat{y} - \bar{y})^2$
- $SST = \sum^n (y - \bar{y})^2$
- $R^2 = 1 - \frac{SSE}{SST}$
- $F = \frac{SST - SSE/1}{SST/n-2} = t^2 = (\frac{\hat{\beta_1}}{se(\hat{\beta_1})})^2$

```r
summary(model1)
```

```

Call:
lm(formula = Sepal.Width ~ Sepal.Length, data = iris)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.1095 -0.2454 -0.0167  0.2763  1.3338 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3.41895    0.25356   13.48   <2e-16 ***
Sepal.Length -0.06188    0.04297   -1.44    0.152    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4343 on 148 degrees of freedom
Multiple R-squared:  0.01382,	Adjusted R-squared:  0.007159 
F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

```r
var(iris$Sepal.Length)
```

```
[1] 0.6856935
```

- Confidence Interval: $\hat{y} \pm t_{n-2} * S\sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum^n (x - \bar{x})^2}}$
- CI ex) $\hat{y} \pm 2.06 * 0.4343\sqrt{\frac{1}{150} + \frac{(5.0 - 5.843333)^2}{149 * 0.6856935}}$
- Prediction Interval: $\hat{y} \pm t_{n-2} * S\sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum^n (x - \bar{x})^2}}$
- PI ex) $\hat{y} \pm 2.06 * 0.4343\sqrt{1 + \frac{1}{150} + \frac{(5.0 - 5.843333)^2}{149 * 0.6856935}}$

ANOVA SLR

Variation	df	SS	MS	F
Regression	1	SSR	SSR/1	SSR/(SSE/(n-2))
Residual/Error	n-2	SSE	SSE/n-2	
Total	n-1	SST		

```r
anova(model1)
```

```
Analysis of Variance Table

Response: Sepal.Width
              Df  Sum Sq Mean Sq F value Pr(>F)
```

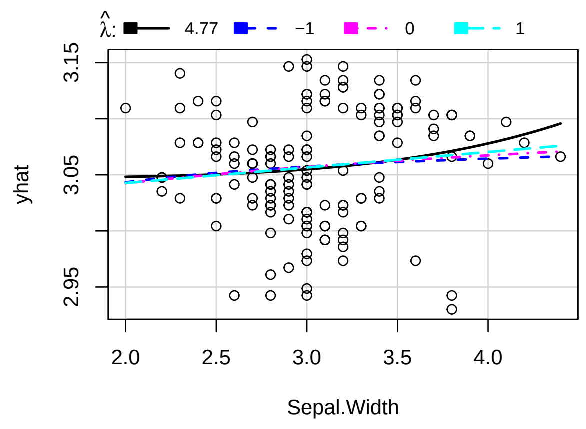
```
Sepal.Length   1  0.3913 0.39128  2.0744 0.1519
Residuals    148 27.9157 0.18862
```

Transformation

Inverse response transformation

- $g^{-1}(y) = \beta_0 + \beta_1 x + \epsilon$, e.g. $g(y) = exp(y), g^{-1}(y) = log(y)$, $g(y) = y^{\lambda}, g^{-1}(y) = y^{1/\lambda}$, make sure response only!
- Pick λ that has the lowest RSS/SSE.

```
library(alr3)
inverseResponsePlot(model1,key=TRUE)
```



```
     lambda       RSS
1  4.768992 0.3829388
2 -1.000000 0.3887806
3  0.000000 0.3873421
4  1.000000 0.3858685
```

Box-cox transformation

- Try to make vairables close to normally distributed. For SLR, maximize likelihood = minimize $SSE(\lambda) = \sum(y^{\lambda} - \hat{\beta}_0 - \hat{\beta}_1 x)^2$. Don't assume normality of x.
- Pick the λ = `Rounded Pwr`

```
library(MASS)
model1bc = powerTransform(model1)
summary(model1bc)
```

```
bcPower Transformation to Normality
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1    0.3429           1      -0.5634       1.2492

Likelihood ratio test that transformation parameter is equal to 0
 (log transformation)
                             LRT df    pval
LR test, lambda = (0) 0.5506328  1 0.45806

Likelihood ratio test that no transformation is needed
                           LRT df    pval
LR test, lambda = (1) 2.01204  1 0.15606
```

Log transformation

- Take logarithm on response or predictors or both. $\log(y_2/y_1) = \beta_1 * \log(x_2/x_1)$. "One percentage change in X results in β_1 percentage change in Y".
- Transform variables according to `Rounded Pwr`, e.g. `Y1=log(Y1)`, `Y2=Y2^1`.
- `Rounded Pwr = 0` means log transform.
- e.g. if `Rounded Pwr = 0.5` for `Y3`, `Y3=sqrt(Y3)`

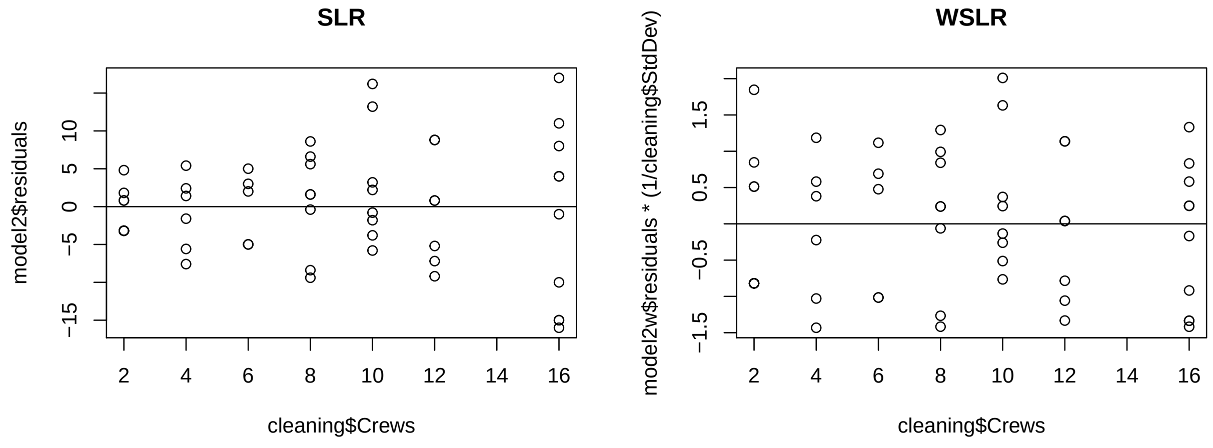
```
model1pt = powerTransform(cbind(iris$Sepal.Length,iris$Sepal.Width)~1)
summary(model1pt)
```

```
bcPower Transformations to Multinormality
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1   -0.1978           0      -1.2981       0.9026
Y2    0.3461           1      -0.5584       1.2506

Likelihood ratio test that transformation parameters are equal to 0
 (all log transformations)
                              LRT df    pval
LR test, lambda = (0 0) 0.6864385  2 0.70948

Likelihood ratio test that no transformations are needed
                             LRT df     pval
LR test, lambda = (1 1) 6.588234  2 0.037101
```

Weighted Least Square

- When constant variance is violated, assigning a reasonable weigth to each variance could fix the problem.
- Assign inversely proportional weights to the corresponding variances.
- $SSE = \sum^n (Y - (\hat{\beta}_0 + \hat{\beta}_1 X))^2$
- $WSSE = \sum^n w_i (Y - (\hat{\beta}_0 + \hat{\beta}_1 X))^2$ with $\epsilon \sim N(0, \sigma^2/w_i)$
- Then, $var(\sqrt{w_i}\epsilon_i) = \sigma^2$
- The weights are assumed to known, so the estimated weights are used. Thus, this method works when the weights can be estimated precisely relative to one another.
- Sensitive to outliers, and possibly increase the influence of them.

```
model2 = lm(Rooms~Crews, data=cleaning)
model2w = lm(Rooms~Crews, weights=1/StdDev^2, data=cleaning)

plot(cleaning$Crews, model2$residuals, main="SLR");abline(h=0);plot(cleaning$Crews, model2w$residuals*(
```

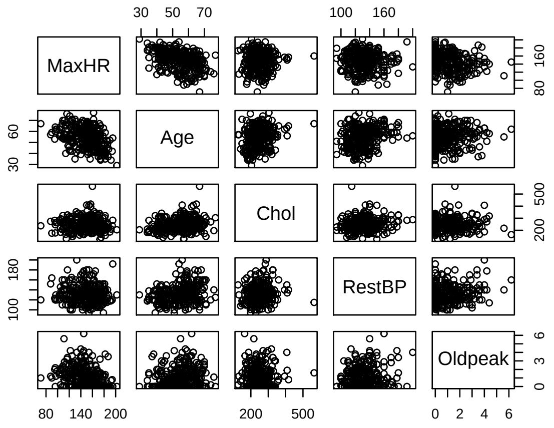


Multiple Linear Regression

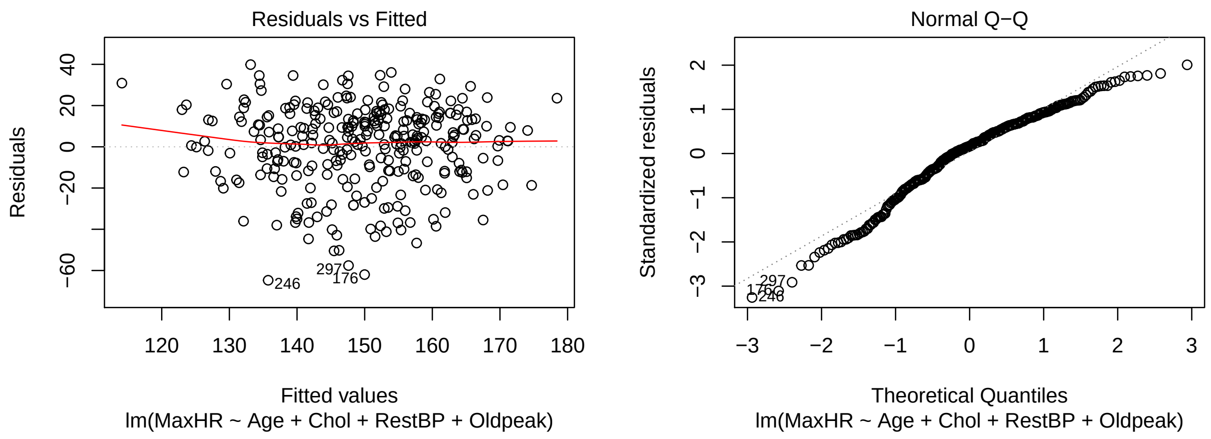
Assumptions

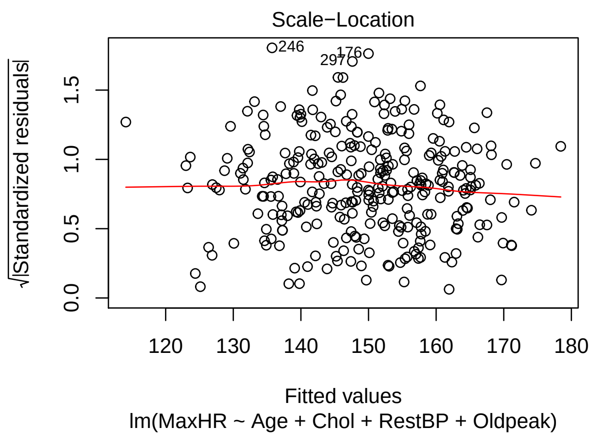
- The response variable and predicotrs are linearly related.
- Error terms are normally distributed.
- Error terms have a constant variance.
- Check the outliers, leverage points, and influential points.
- Check if the predictors are highly correlated (multicolliniearity).
- Check the diagonal elements of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. If $h_{ii} > 2 * \bar{h} = 2 * \frac{p+1}{n}$, the point is consider to be a leverage points for MLR.
- Standarized residual $r_i = \frac{\hat{e_i}}{S\sqrt{1-h_{ii}}}$ where $S^2 = \frac{SSE}{n-(p+1)}$. If r_i is outside $(-2, 2)$, it's considered to be an outlier.
- Likewise, check for Cook's distance D_i. If greater than $4/(n-2)$, the point is an influential point for MLR.
- The diagnosis plots show if the entire model is valid.

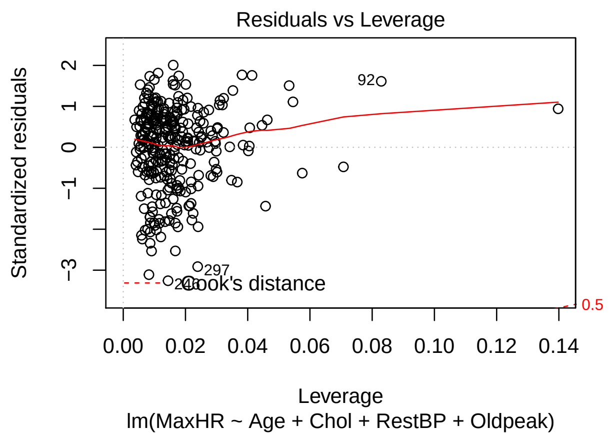
```r
m = lm(MaxHR~Age+Chol+RestBP+Oldpeak, data=Heart)
pairs(MaxHR~Age+Chol+RestBP+Oldpeak, data=Heart)
```



```r
plot(m)
```







```r
stdres1 = rstandard(m)
lev1 = hatvalues(m)
cookd1 = cooks.distance(m)

# outliers
which(abs(stdres1)>2)
```

```
 24  38  80  88 115 172 176 224 229 237 245 246 253 297
 24  38  80  88 115 172 176 224 229 237 245 246 253 297
```

```r
# leverage ( hii > 2*(p+1)/n)
which(lev1 > 2*(4+1)/nrow(Heart))
```

```
  4  49  69  92 114 122 124 127 153 162 182 184 189 192 202 212 232 274
  4  49  69  92 114 122 124 127 153 162 182 184 189 192 202 212 232 274
286
286
```

```r
# influential points (Di > 4/(n-2))
which(cookd1 > 4/(nrow(Heart)-2))
```

```
  4  30  92 122 152 153 162 176 189 212 232 245 246 266 297
  4  30  92 122 152 153 162 176 189 212 232 245 246 266 297
```

- When strong correlations exist among the predictor variables, the following issues may arise:
- 1. F-test results will be highly significan, when very few predictors are significant.
- 2. Some of the coefficients in the model show the opposite sign than expected.
- Variance Inflation Factor (VIF) : $\frac{1}{1-R_j^2}$, where R_j^2 denote the value of R^2 obtained from the regression of x_j on the other$x$'s. Note that $var(\beta_j) = \frac{1}{1-R_j^2} * \frac{\sigma^2}{(n-1)S_{x_j}^2}$. If $VIF_j > 5$, then $\hat{\beta}_j$ is poorly estimated due to multicollinearity.
- There are several ways to handle multicollinearity. One way is to delete the redundant predictors (highly correlated predictors).
- Another way to handle multicollenearity is to make the dataset uncorrelated i.e. linearly independend. The method is called Principle Component Analysis.

```r
library(car)
vif(m)
```

```
     Age     Chol   RestBP  Oldpeak 
1.153910 1.051659 1.116389 1.064148 
```

Statistical Inference

- $\mathbf{Y} = \mathbf{X}\beta + \epsilon$
- $SSE = (\mathbf{Y} - \mathbf{X}\hat{\beta})^T(\mathbf{Y} - \mathbf{X}\hat{\beta}) = ||\mathbf{Y} - \mathbf{X}\hat{\beta}||^2$
- $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\mathbf{Y}$
- $\mathbf{X} = (1, X_1, X_2, \ldots, X_p)^T$
- $S^2 = \frac{SSE}{n-p-1}$
- $T_i = \frac{\hat{\beta}_i}{se(\hat{\beta}_i)} \sim t_{n-p-1}$ for $H_0 : \beta_i = 0$
- $R^2 = 1 - \frac{SSE}{SST}$, but always increase as p increase.
- $R^2_{adj} = 1 - \frac{SSE/(n-p-1)}{SST/(n-1)}$
- $F = \frac{(SST-SSE)/p}{SSE/(n-p-1)} \sim F_{p,(n-p-1)}$ for $H_0 : \beta_1 = \cdots = \beta_p = 0$

```
summary(m)
```

```

Call:
lm(formula = MaxHR ~ Age + Chol + RestBP + Oldpeak, data = Heart)

Residuals:
    Min      1Q  Median      3Q     Max 
-64.737 -11.889   3.353  13.875  39.861 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 181.31908   10.60124  17.104  < 2e-16 ***
Age          -0.96548    0.13690  -7.052 1.23e-11 ***
Chol          0.03343    0.02281   1.465   0.1439    
RestBP        0.14069    0.06915   2.034   0.0428 *  
Oldpeak      -5.70035    1.02343  -5.570 5.70e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.02 on 298 degrees of freedom
Multiple R-squared:  0.2443,	Adjusted R-squared:  0.2342 
F-statistic: 24.09 on 4 and 298 DF,  p-value: < 2.2e-16
```

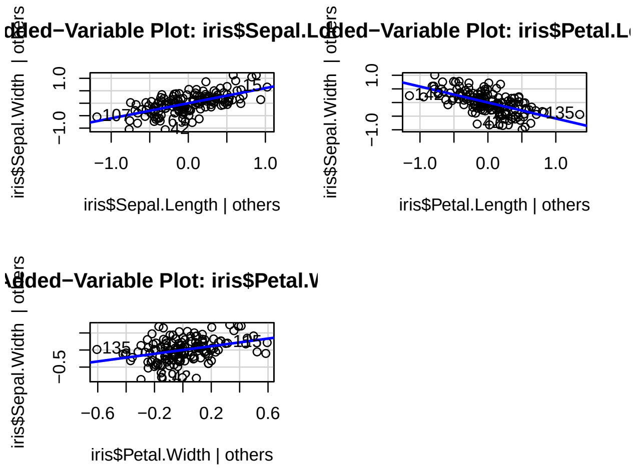
```
# or
Y = Heart$MaxHR
X = cbind(1,Heart$Age, Heart$Chol, Heart$RestBP, Heart$Oldpeak)
beta = solve(t(X)%*%X)%*%t(X)%*%Y;beta
```

```
             [,1]
[1,] 181.31907819
[2,]  -0.96547761
[3,]   0.03343059
[4,]   0.14068501
[5,]  -5.70035136
```

Model Diagnosis : Added Variable Plot

- Added variable plot enable us to visually assess the effect of each predictors, having adjusted for the effects of other predictors.
- In stead of the model $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, consider $\mathbf{Y} = \mathbf{X}\beta + \mathbf{Z}\alpha + \epsilon$.
- $\mathbf{Z} = \mathbf{X}\delta + \epsilon$ and $n \times 1$ vector. If the model with $\mathbf{Z}$ fits better to the data, then the added variable plot should produce points randomly scattered around a line through thte origin with slope $\hat{\alpha}$

```
library(car)
m = lm(iris$Sepal.Width~iris$Sepal.Length+iris$Petal.Length+iris$Petal.Width)
par(mfrow=c(2,2))
avPlot(m,variable=iris$Sepal.Length, ask=F)
avPlot(m,variable=iris$Petal.Length, ask=F)
avPlot(m,variable=iris$Petal.Width, ask=F)
par(mfrow=c(1,1))
```



ANOVA MLR

Variation	df	SS	MS	F
Regression	p	SSR	SSR/p	(SST-SSE)/p/(SSE/(n-2))
Residual/Error	n-p-1	SSE	SSE/n-p-1	
Total	n-1	SST		

```
anova(model2)
```

```
Analysis of Variance Table

Response: Rooms
          Df  Sum Sq Mean Sq F value    Pr(>F)
Crews      1 16429.7 16429.7  305.27 < 2.2e-16 ***
Residuals 51  2744.8    53.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nested Model and Partial F-test

$$H_o : \beta_1 = \cdots = \beta_k = 0 \quad \text{(k < p i.e. reduced model)}$$
$$\text{vs.}$$
$$H_a : \beta_1 = \cdots = \beta_p = 0 \quad \text{(i.e. full model)}$$

- $F = \frac{SSE_{reduced} - SSE_{full}/(df_{reduced} - df_{full})}{SSE_{full}/df_{full}} = \frac{SSE_{reduced} - SSE_{full}/k}{SSE_{full}/df_{full}}$
- Although here the term "reduced" and "full" are used, make sure that "full" could already be a reduced model i.e. $n - p - 1 \geq df_{full}$.

```
model2reduced = lm(MaxHR~Age+RestBP+Oldpeak, data=Heart)
anova(model2,model2reduced)
```

```
Analysis of Variance Table

Response: Rooms
          Df  Sum Sq Mean Sq F value    Pr(>F)
Crews      1 16429.7 16429.7  305.27 < 2.2e-16 ***
Residuals 51  2744.8    53.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

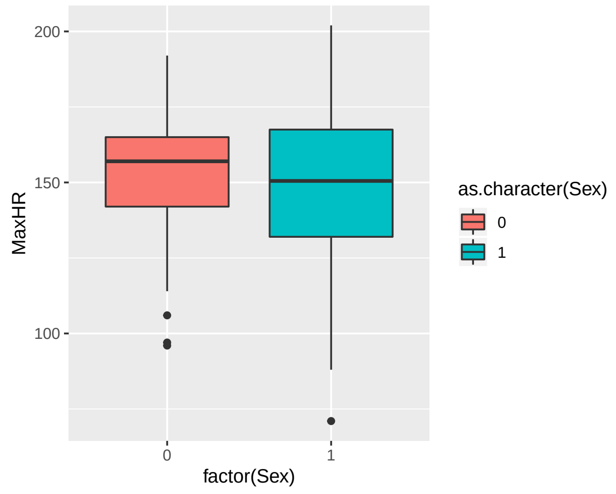
- Here, F-stats is not significant i.e. failed to reject null, reduced model is better.

ANCOVA (Analysis of Covariance)

- Suppose we have a categorical variable with K levels. ANCOVA allows a categorical variable to be included in a linear model. Technically speaking, ANOVA is a sub-technique of ANCOVA where we have a hidden categorical variable with only 1 level.
- There could be multiple categorical variable. If that's the case, the number of levels would be $K_1 \times K_2$, and might require more sample size.
- Use variable selections (e.g. partial F-test) to reduce the number of predictors.
- $Y = \beta_0 + \beta_1 x + \beta_2 d + \beta_3 (d \times x) + \epsilon$ if $d \in \{0, 1\}$
- $SSB = \sum_{i=1}^{k} \sum_{j=1}^{n_k} (\bar{Y}_i - \bar{\bar{Y}})^2$, similar to SSE
- $SSW = \sum_{i=1}^{k} \sum_{j=1}^{n_k} (Y_{ij} - \bar{Y}_i)^2$, similar to SSR
- $SST = \sum_{i=1}^{k} \sum_{j=1}^{n_k} (Y_{ij} - \bar{\bar{Y}})^2$
- $H_o : \mu_1 = \mu_2 = \cdots = \mu_K$ vs. H_a : at least one of the group means is different

Variation	df	SS	MS	F
Between	k-1	SSB	SSB/(k-1)	SSB/(k-1)/(SSW/(n-k))
Within	n-k	SSW	SSW/n-k	
Total	n-1	SST		

```r
ggplot(data = Heart) + geom_boxplot(mapping = aes(x=factor(Sex), y=MaxHR, fill=as.character(Sex)))
```

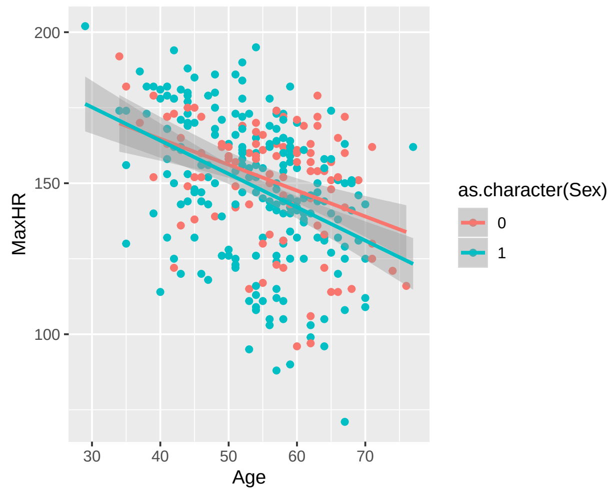


```r
model3 = lm(MaxHR~Age*Sex, data=Heart)
# Same as lm(MaxHR~Age+Sex+Age:Sex)
summary(model3)
```

```

Call:
lm(formula = MaxHR ~ Age * Sex, data = Heart)

Residuals:
    Min      1Q  Median      3Q     Max 
-63.319 -13.192   3.475  15.630  46.337 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) 198.9724    12.9278  15.391  < 2e-16 ***
Age          -0.8569     0.2288  -3.745 0.000217 ***
Sex           9.2725    15.7855   0.587 0.557373
Age:Sex      -0.2465     0.2827  -0.872 0.383945
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.01 on 299 degrees of freedom
Multiple R-squared:  0.1649,	Adjusted R-squared:  0.1565
F-statistic: 19.67 on 3 and 299 DF,  p-value: 1.147e-11
```

```
ggplot(data=Heart) + aes(x=Age, y=MaxHR,colour=as.character(Sex))+ geom_point() + geom_smooth(method="l
```



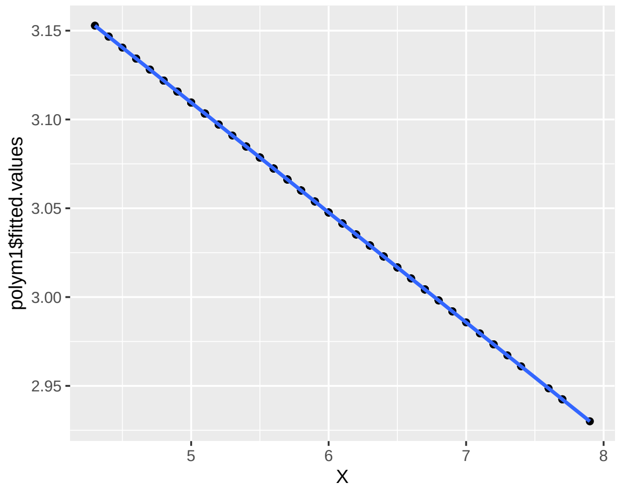
Polynomial Regression

- $Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_h x^h + \epsilon$
- Pick the degree that gives the lowest MSE and higest R square adjusted.
- Be careful of over-fitting.

```r
Y = iris$Sepal.Width
X = iris$Sepal.Length
polym1 = lm(Y~X)
summary(polym1)
```

```

Call:
lm(formula = Y ~ X)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.1095 -0.2454 -0.0167  0.2763  1.3338 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.41895    0.25356   13.48   <2e-16 ***
X           -0.06188    0.04297   -1.44    0.152    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4343 on 148 degrees of freedom
Multiple R-squared:  0.01382,	Adjusted R-squared:  0.007159 
F-statistic: 2.074 on 1 and 148 DF,  p-value: 0.1519
```

```r
qplot(X,polym1$fitted.values, geom=c("point","smooth"))
```

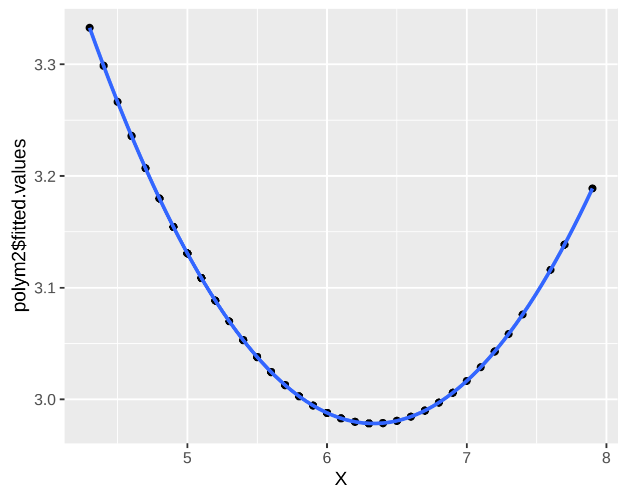


```r
polym2 = lm(Y~X+I(X^2))
summary(polym2)
```

```

Call:
lm(formula = Y ~ X + I(X^2))

Residuals:
     Min       1Q   Median       3Q      Max 
```

```
-1.13070 -0.26310 -0.02446  0.25728  1.38725

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  6.41584    1.58499   4.048 8.33e-05 ***
X           -1.08556    0.53625  -2.024   0.0447 *  
I(X^2)       0.08571    0.04476   1.915   0.0574 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4304 on 147 degrees of freedom
Multiple R-squared:  0.03783,	Adjusted R-squared:  0.02474 
F-statistic:  2.89 on 2 and 147 DF,  p-value: 0.05877
```

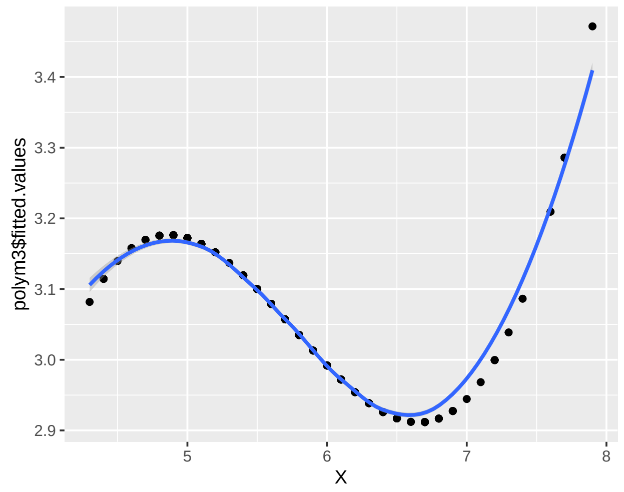
```r
qplot(X,polym2$fitted.values, geom=c("point","smooth"))
```



```r
polym3 = lm(Y~X+I(X^2)+I(X^3))
summary(polym3)
```

```

Call:
lm(formula = Y ~ X + I(X^2) + I(X^3))

Residuals:
     Min       1Q   Median       3Q      Max 
-1.17219 -0.23769 -0.00581  0.27359  1.34285 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)  
(Intercept) -13.25795    9.98420  -1.328   0.1863  
X             8.93524    5.05021   1.769   0.0789 .
I(X^2)       -1.58995    0.84097  -1.891   0.0607 .
I(X^3)        0.09202    0.04612   1.995   0.0479 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4261 on 146 degrees of freedom
Multiple R-squared:  0.06337,	Adjusted R-squared:  0.04412 
F-statistic: 3.292 on 3 and 146 DF,  p-value: 0.02239
```

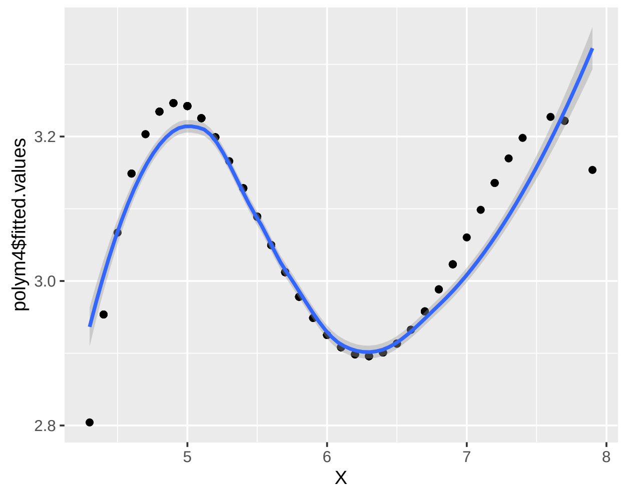
```
qplot(X,polym3$fitted.values, geom=c("point","smooth"))
```



```
polym4 = lm(Y~X+I(X^2)+I(X^3)+I(X^4))
summary(polym4)
```

```
Call:
lm(formula = Y ~ X + I(X^2) + I(X^3) + I(X^4))

Residuals:
     Min       1Q   Median       3Q      Max 
-1.24217 -0.23267  0.00282  0.25123  1.38777 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)  
(Intercept) -142.56391   63.82905  -2.234   0.0270 *
X             97.48827   43.47380   2.242   0.0264 *
I(X^2)       -24.06445   10.99196  -2.189   0.0302 *
I(X^3)         2.59737    1.22267   2.124   0.0353 *
I(X^4)        -0.10351    0.05048  -2.051   0.0421 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4215 on 145 degrees of freedom
Multiple R-squared:  0.08976,   Adjusted R-squared:  0.06465 
F-statistic: 3.575 on 4 and 145 DF,  p-value: 0.008209
```

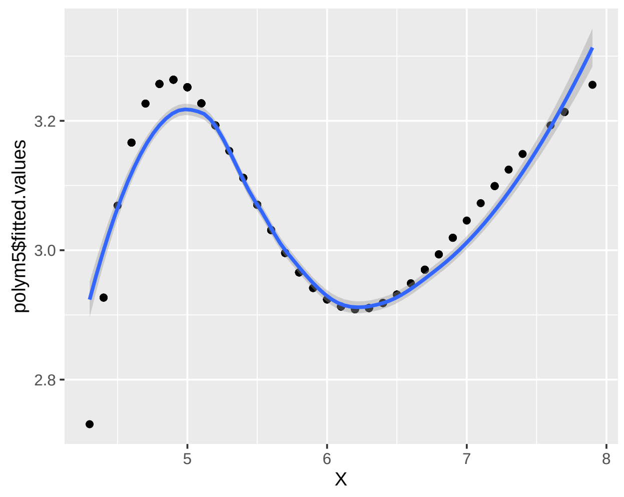
```
qplot(X,polym4$fitted.values, geom=c("point","smooth"))
```



```r
polym5 = lm(Y~X+I(X^2)+I(X^3)+I(X^4)+I(X^5))
summary(polym5)
```

```
Call:
lm(formula = Y ~ X + I(X^2) + I(X^3) + I(X^4) + I(X^5))

Residuals:
     Min       1Q   Median       3Q      Max 
-1.25194 -0.22446 -0.00225  0.25581  1.40416 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -369.84591  416.99626  -0.887    0.377
X            292.79809  356.76489   0.821    0.413
I(X^2)       -90.55757  121.05337  -0.748    0.456
I(X^3)        13.80872   20.36289   0.678    0.499
I(X^4)        -1.03981    1.69825  -0.612    0.541
I(X^5)         0.03099    0.05618   0.552    0.582

Residual standard error: 0.4226 on 144 degrees of freedom
Multiple R-squared:  0.09168,	Adjusted R-squared:  0.06014 
F-statistic: 2.907 on 5 and 144 DF,  p-value: 0.01568
```

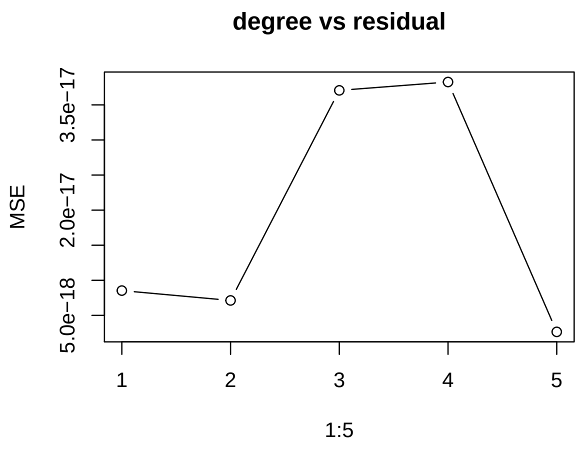
```r
qplot(X,polym5$fitted.values, geom=c("point","smooth"))
```



```
#MSEs
polyMSE = c(sqrt(sum(polym1$residuals)^2)/148,sqrt(sum(polym2$residuals)^2)/147,sqrt(sum(polym3$residual
polyMSE
```

```
[1] 8.532964e-18 7.127707e-18 3.707080e-17 3.825963e-17 2.650272e-18
```

```
plot(1:5, polyMSE, type="b", main="degree vs residual", ylab="MSE")
```



Model Selection Criteria

- When multicollinearity among the predictor variables are observed, model selection is one way to resolve it.
- Make sure to check all the assumptions are met before starting model selection.
- Goodness of fit criteria:

1. Adjusted R-square: $R^2_{adj} = 1 - \frac{SSE/(n-p-1)}{SST/(n-1)} = 1 - \frac{(1-R^2)(n-1)}{(n-p-1)}$. We don't use R-square because it automatically increase as the number of predictors increase.
2. Akaike Information Criterion (AIC): Smaller the better. Reward for a good fit + penalty for complexity. $AIC = n * \log(\frac{SSE}{n}) + 2p$
3. AIC corrected: Greater penality. Smaller the better. $AIC_C = AIC + \frac{2p(p+2)(p+3)}{n-p-1}$
4. Bayes Information Criteria (BIC): Greater penalty than AIC when $\log(n) > 2$, thus favors simpler model than AIC. As the sample size n increase, the probability that BIC choose the correct model becomes 1. For smaller n, BIC choose too simple model hence biased. $BIC = n * \log(\frac{SSE}{n}) + \log(n) * p$
5. Mean Square Error: When test data is given, MSE is the most reliable measurement for choosing the best model. Combine with CV.

Subset Selection: Best Subset Model

- For p predictors, we have $\sum_{k=1}^{p} \binom{p}{k}$ possible subset models.

1. Start with $k = p$ i.e. full model, fit the model.
2. $k = p - 1$. Fit all $\binom{p}{p-1}$ models, keep the winner among $\binom{p}{p-1}$ with higest R^2. Here it's R^2 because we are comparing the models with same number of predictors.
3. $k = p - 2$, keep the winner.
4. Repeat until $k = 1$.
5. Choose the best model among all winners. Use the criteria other than R^2.

R: Best Subset Model

```r
library(leaps)
X = cbind(iris$Sepal.Length, iris$Petal.Length, iris$Petal.Width)
b = regsubsets(X, iris$Sepal.Width)
bs = summary(b);bs
```

```
Subset selection object
3 Variables  (and intercept)
  Forced in Forced out
a     FALSE      FALSE
b     FALSE      FALSE
c     FALSE      FALSE
1 subsets of each size up to 3
Selection Algorithm: exhaustive
         a   b   c  
1  ( 1 ) " " "*" " "
2  ( 1 ) "*" "*" " "
3  ( 1 ) "*" "*" "*"
```

```r
om1 = lm(Sepal.Width~Petal.Length,data=iris)
om2 = lm(Sepal.Width~Sepal.Length+Petal.Length,data=iris)
om3 = lm(Sepal.Width~Sepal.Length+Petal.Length+Petal.Width,data=iris)

n = nrow(iris)

p=1
AIC1 = extractAIC(om1,k=2)[2]
AICc1 = extractAIC(om1,k=2)[2] + 2*(p+2)*(p+3)/(n-p-1)
BIC1 = extractAIC(om1, k=log(n))[2]

p=2
AIC2 = extractAIC(om1,k=2)[2]
AICc2 = extractAIC(om1,k=2)[2] + 2*(p+2)*(p+3)/(n-p-1)
BIC2 = extractAIC(om1, k=log(n))[2]

p=3
AIC3 = extractAIC(om1,k=2)[2]
AICc3 = extractAIC(om1,k=2)[2] + 2*(p+2)*(p+3)/(n-p-1)
BIC3 = extractAIC(om1, k=log(n))[2]

AIC = c(AIC1,AIC2,AIC3)
AICc = c(AICc1, AICc2, AICc3)
BIC = c(BIC1, BIC2, BIC3)

data.frame(Radj2 = bs$adjr2,AIC,AICc,BIC)
```

```
      Radj2       AIC      AICc       BIC
1 0.1780444 -276.5497 -276.3876 -270.5285
2 0.4490193 -276.5497 -276.2776 -270.5285
3 0.5142264 -276.5497 -276.1388 -270.5285
```

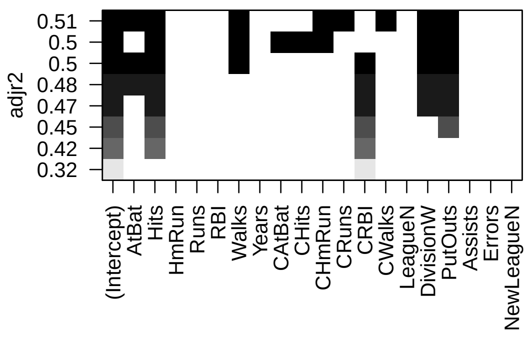
R: Best Subset Model (Alternative)

- Black represents that the variables are selected. Pick the variables that highlighted on the very top.

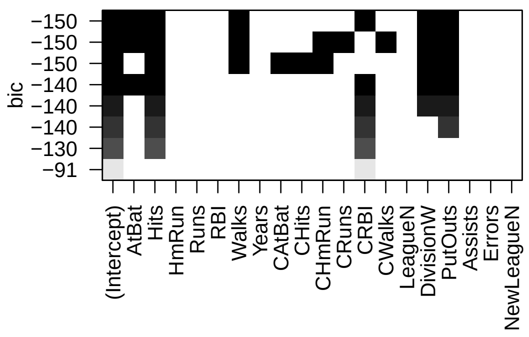
```r
library(ISLR)
Hitters=na.omit(Hitters)
regfit.full=regsubsets(Salary~.,Hitters)
summary(regfit.full)
```

```
Subset selection object
Call: regsubsets.formula(Salary ~ ., Hitters)
19 Variables  (and intercept)
           Forced in Forced out
AtBat          FALSE      FALSE
Hits           FALSE      FALSE
HmRun          FALSE      FALSE
Runs           FALSE      FALSE
RBI            FALSE      FALSE
Walks          FALSE      FALSE
Years          FALSE      FALSE
CAtBat         FALSE      FALSE
CHits          FALSE      FALSE
CHmRun         FALSE      FALSE
CRuns          FALSE      FALSE
CRBI           FALSE      FALSE
CWalks         FALSE      FALSE
LeagueN        FALSE      FALSE
DivisionW      FALSE      FALSE
PutOuts        FALSE      FALSE
Assists        FALSE      FALSE
Errors         FALSE      FALSE
NewLeagueN     FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "  
2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "  
6  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "  
7  ( 1 ) " "   "*"  " "   " "  " " "*"   " "   "*"    "*"   "*"    " "  
8  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   "*"    "*"  
         CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
1  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "       
2  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "       
3  ( 1 ) "*"  " "    " "     " "       "*"     " "     " "    " "       
4  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "       
5  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "       
6  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "       
7  ( 1 ) " "  " "    " "     "*"       "*"     " "     " "    " "       
8  ( 1 ) " "  "*"    " "     "*"       "*"     " "     " "    " "       
```

```r
par(mfrow=c(1,1))
plot(regfit.full,scale="adjr2")
```



```r
plot(regfit.full,scale="bic")
```



Subset Selection: Stepwise Regression

Forward Stepwise

1. Start with null model (intercept only)
2. Fit model with $k = 1$. Choose the best among p models based on R^2.
3. Add another vairbale to the previous model, keep the best model.
4. Repeat until $k = p$.
5. Choose the best among the p candidate models using AIC, BIC etc…

R: Forward AIC

```r
m = lm(Sepal.Width~1, data=iris)

forwardAIC = step(m, scope = list(lower=~1,
upper=~Sepal.Length+Petal.Length+Petal.Width,data),
direction = "forward", data=iris)
```

```
Start:  AIC=-248.13
Sepal.Width ~ 1

               Df Sum of Sq    RSS     AIC
+ Petal.Length  1    5.1960 23.111 -276.55
+ Petal.Width   1    3.7945 24.512 -267.72
+ Sepal.Length  1    0.3913 27.916 -248.22
<none>                      28.307 -248.13

Step:  AIC=-276.55
Sepal.Width ~ Petal.Length

               Df Sum of Sq    RSS     AIC
+ Sepal.Length  1    7.7237 15.387 -335.56
+ Petal.Width   1    0.8363 22.275 -280.08
<none>                      23.111 -276.55

Step:  AIC=-335.56
Sepal.Width ~ Petal.Length + Sepal.Length

              Df Sum of Sq    RSS     AIC
+ Petal.Width  1    1.9133 13.474 -353.48
<none>                     15.387 -335.56

Step:  AIC=-353.48
Sepal.Width ~ Petal.Length + Sepal.Length + Petal.Width
```

R: Forward BIC

```r
forwardBIC = step(m, scope = list(lower=~1,
upper=~Sepal.Length+Petal.Length+Petal.Width,data),
direction = "forward", data=iris, k=log(n))
```

```
Start:  AIC=-245.12
Sepal.Width ~ 1

               Df Sum of Sq    RSS     AIC
+ Petal.Length  1    5.1960 23.111 -270.53
+ Petal.Width   1    3.7945 24.512 -261.70
<none>                      28.307 -245.12
+ Sepal.Length  1    0.3913 27.916 -242.20

Step:  AIC=-270.53
Sepal.Width ~ Petal.Length

               Df Sum of Sq    RSS     AIC
+ Sepal.Length  1    7.7237 15.387 -326.53
+ Petal.Width   1    0.8363 22.275 -271.05
<none>                      23.111 -270.53

Step:  AIC=-326.53
Sepal.Width ~ Petal.Length + Sepal.Length

              Df Sum of Sq    RSS     AIC
+ Petal.Width  1    1.9133 13.474 -341.44
<none>                     15.387 -326.53

Step:  AIC=-341.44
Sepal.Width ~ Petal.Length + Sepal.Length + Petal.Width
```

Backward Stepwise

1. Start with full model.
2. Fit all models with $k = p - 1$, pick the best model.
3. Reduce another vriable from 2., pick the best model.
4. Repeat until $k = 1$.
5. Choose the best among the p candidate models using AIC, BIC etc…

R: Backward AIC

```r
m = lm(mpg~disp+hp+drat+wt+qsec,data=mtcars)
backAIC = step(m,direction = "backward", data=mtcars)
```

```
Start:  AIC=65.47
mpg ~ disp + hp + drat + wt + qsec

       Df Sum of Sq    RSS    AIC
- disp  1     3.974 174.10 64.205
<none>              170.13 65.466
- hp    1    11.886 182.01 65.627
- qsec  1    12.708 182.84 65.772
- drat  1    15.506 185.63 66.258
- wt    1    81.394 251.52 75.978

Step:  AIC=64.21
mpg ~ hp + drat + wt + qsec

       Df Sum of Sq    RSS    AIC
- hp    1     9.418 183.52 63.891
- qsec  1     9.578 183.68 63.919
<none>              174.10 64.205
- drat  1    11.956 186.06 64.331
- wt    1   113.882 287.99 78.310

Step:  AIC=63.89
mpg ~ drat + wt + qsec

       Df Sum of Sq    RSS    AIC
<none>              183.52 63.891
- drat  1    11.942 195.46 63.908
- qsec  1    85.720 269.24 74.156
- wt    1   275.686 459.21 91.241
```

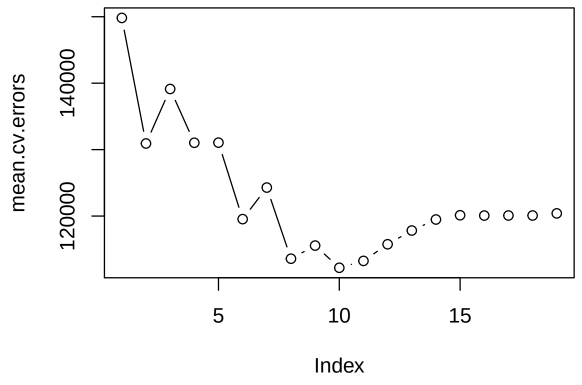
R: Backward BIC

```r
backBIC = step(m, direction="backward", data=mtcars, k=log(n))
```

```
Start:  AIC=83.53
mpg ~ disp + hp + drat + wt + qsec

       Df Sum of Sq    RSS    AIC
- disp  1     3.974 174.10 79.258
- hp    1    11.886 182.01 80.681
- qsec  1    12.708 182.84 80.825
- drat  1    15.506 185.63 81.311
<none>              170.13 83.530
- wt    1    81.394 251.52 91.031

Step:  AIC=79.26
mpg ~ hp + drat + wt + qsec

       Df Sum of Sq    RSS    AIC
- hp    1     9.418 183.52 75.934
- qsec  1     9.578 183.68 75.962
- drat  1    11.956 186.06 76.373
<none>              174.10 79.258
- wt    1   113.882 287.99 90.352

Step:  AIC=75.93
mpg ~ drat + wt + qsec

       Df Sum of Sq    RSS     AIC
- drat  1    11.942 195.46  72.940
<none>              183.52  75.934
- qsec  1    85.720 269.24  83.188
- wt    1   275.686 459.21 100.272

Step:  AIC=72.94
mpg ~ wt + qsec

       Df Sum of Sq    RSS     AIC
<none>              195.46  72.940
- qsec  1     82.86 278.32  79.239
- wt    1    733.19 928.66 117.797
```

R: Forward and Backward Selection (Alternative)

```r
regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="backward")

# Choosing best number of predictors w/ CV, based on MSE.

k=10
set.seed(1)
folds=sample(1:k,nrow(Hitters),replace=TRUE)
cv.errors=matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))

predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}


for(j in 1:k){
  best.fit=regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=19)
  for(i in 1:19){
    pred=predict(best.fit,Hitters[folds==j,],id=i)
    cv.errors[j,i]=mean((Hitters$Salary[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
plot(mean.cv.errors,type='b')
```



```r
reg.best=regsubsets(Salary~.,data=Hitters, nvmax=19);coef(reg.best,11)
```

```
 (Intercept)        AtBat         Hits        Walks       CAtBat 
 135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914 
       CRuns         CRBI       CWalks      LeagueN    DivisionW 
   1.4553310    0.7852528   -0.8228559   43.1116152 -111.1460252 
     PutOuts      Assists 
   0.2894087    0.2688277 
```

Shrinkage: Regularization

- Fit a model involving all p predictors, but the estimated coefficients are shrunken toward zero. This shirknkage has the effect of reducing variance and can also perfrom variable selection.
- For both shrinkage methods below, it is recommended to use CV to choose the best tuning parameter.

Ridge Regression

- Instead of minimizing loss function $SSE = (Y - X\hat{\beta})^T(Y - X\hat{\beta})$, minimize the loss function with L2 penalty term:
- Minimize $(Y - X\hat{\beta})^T(Y - X\hat{\beta}) + \lambda\hat{\beta}^T\hat{\beta} = SSE + \lambda||\hat{\beta}||_2^2$ where $\lambda \geq 0$
- The coefficients generated by Ridge regression tends to be similar in value (absolute value-wise) as the tuning parameter λ increases. This is because the penalty term uses Euclidean distance, or L2 norm. When the tuning parameter is too large, all coefficients goes down to 0.
- Ridge regression coefficiet estimate can change substantially when multiplying a given predictor by a constant, due to L2 norm part. Thus, it is best to apply Ridge regression after standarizing the predictors using the formula: $\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}}$

LASSO regression

- Minimize $(Y - X\hat{\beta})^T(Y - X\hat{\beta}) + \lambda\mathbf{1}^T|\hat{\beta}| = SSE + \lambda||\hat{\beta}||_1$ where $\lambda \geq 0$
- The coefficients generated by LASSO regression tends to be reduced down to 0 as the tuning parameter increases. This is because the penalty term uses Manhattan distance, or L1 norm. Thus, this shrinkage method can perform as variable selection.
- We say that the lasso yeilds sparse models i.e. the models that involve only a subset of variables, or contains many zeros.

Elastic Net

- The combination of LASSO and Ridge.
- Minimize $SSE + \lambda(\frac{(1-\alpha)}{2}||\hat{\beta}||_2^2 + \alpha||\hat{\beta}||_1)$ where $\lambda \geq 0$, $\alpha \in (0, 1)$

R: Shrinkage

- Ridge: $\alpha = 0$
- LASSO: $\alpha = 1$
- Elastic Net: $\alpha \in (0, 1)$
- Make sure to use matrix / vector for the inputs. Sparse matrix from `library(Matrix)` is supported.

R: Ridge Regression

```
library(glmnet)
library(ISLR) # For Hitters dataset
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary


set.seed(1)
train=traindex(x)
temp = 1:nrow(x)
test=temp[-train]
y.test=y[test]
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x[train,],y[train], alpha=0,lambda=grid, thresh=1e-12)

# lambda = 4
ridge.pred=predict(ridge.mod,s=4,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
[1] 165161.7
```

```
# lambda = 0
ridge.pred=predict(ridge.mod,s=0,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
[1] 163360
```

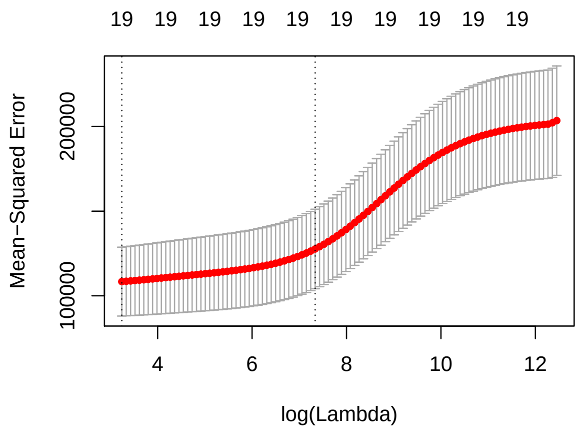
```
# When lambda = 0, same as lm
lm(y~x, subset=train)
```

```

Call:
lm(formula = y ~ x, subset = train)

Coefficients:
(Intercept)       xAtBat        xHits       xHmRun        xRuns
   219.6948      -2.1459       7.4588       0.3843      -2.5243
       xRBI       xWalks       xYears      xCAtBat       xCHits
     0.6735       6.6994     -11.9238      -0.0865       0.1849
    xCHmRun       xCRuns        xCRBI      xCWalks     xLeagueN
    -0.2475       0.9189       0.9029      -0.9902      68.1354
 xDivisionW     xPutOuts     xAssists      xErrors  xNewLeagueN
  -145.4021       0.3320       0.3830      -5.3149     -55.1098
```

```
ridge.mod=glmnet(x[train,],y[train], alpha=0,lambda=0, thresh=1e-12)
predict(ridge.mod,s=0,exact=T,type="coefficients")
```

```
20 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  219.69034188
AtBat         -2.14582772
Hits           7.45829955
HmRun          0.38382694
Runs          -2.52386914
RBI            0.67358027
Walks          6.69923354
Years        -11.92178980
CAtBat        -0.08654625
CHits          0.18518313
CHmRun        -0.24717482
CRuns          0.91871110
CRBI           0.90278683
CWalks        -0.99010232
LeagueN       68.13853859
DivisionW   -145.40170447
PutOuts        0.33202384
Assists        0.38306254
Errors        -5.31509262
NewLeagueN   -55.11307800
```

```
# Cross Validation to choose lambda
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
```



```
bestlam=cv.out$lambda.min
bestlam
```

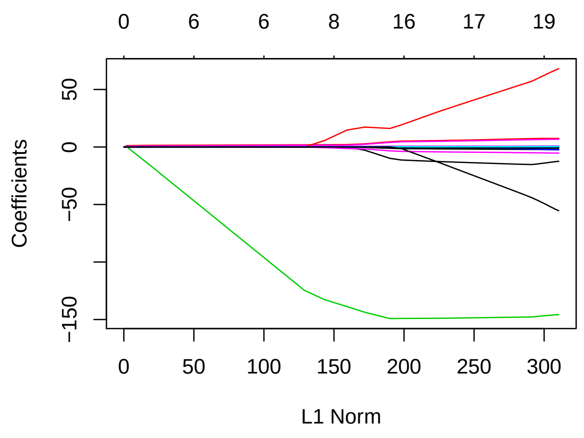
```
[1] 25.58362
```

```
ridge.mod=glmnet(x[train,],y[train], alpha=0,lambda=bestlam, thresh=1e-12)
ridge.pred=predict(ridge.mod, newx=x[test,])
mean((ridge.pred-y.test)^2)
```

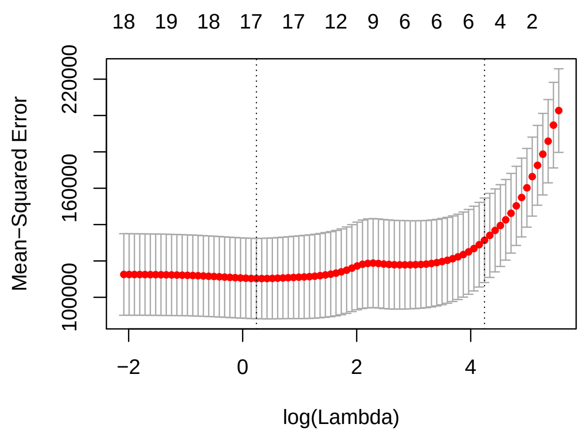
```
[1] 149900.7
```

R: LASSO Regression

```r
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```



```r
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```



```r
bestlam=cv.out$lambda.min
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=bestlam)
lasso.pred=predict(lasso.mod,newx=x[test,])
mean((lasso.pred-y.test)^2)
```

```
[1] 163253
```

```r
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,];lasso.coef[lasso.coef!=0]
```

```
  (Intercept)         AtBat          Hits         HmRun          Runs
 149.54920606   -1.88397299    6.55952778    0.88197054   -0.59100530
        Walks         Years        CAtBat        CHmRun         CRuns
   5.43552977   -8.55446820   -0.04530057    0.34497437    1.00059582
         CRBI        CWalks       LeagueN     DivisionW       PutOuts
   0.51183549   -0.69430903   42.45335427 -117.37199676    0.28122999
```

```
     Assists        Errors    NewLeagueN 
  0.26777558   -2.71817199   -6.60210398 
```

Dimension Reduction: Principle Component Analysis

- Project the p predictors into M-dimensional subspace, where M < p. This is achieved by computing M different linear combinations of the variables. Then, M projections are used as predictor to fit a linear regression model by least squares.
- Before applying PCA, normalize the dataset.
- The first principle component is the linear combination of the variables with the largest variance so that the dataset is easily distinguishable e.g. LDA or grouping would perform well.
- The second principle component is also the linear combination of the vairables with the largest variance, subject to being uncorrelated with the first one i.e. perpendicular / independent of the first one, and third PC, forth PC, and so on.
- Drawback: the directions / new dimensions identified by PCA, i.e. the way linear combinations are created, is unsupervised way since the response Y is not used to help determine the PCA directions. Thus, there is no gurantee that the directions that best explain the predictors will also be the best directions to use for predicting the response.
- Regression on data with PCA applied is called principle component regression.
- Mathematically, we have the following optimization problem. Solving this using the method of Lagrange multiplier, it reveals that the PCA coefficients of the linear combinations are the eigenvectors of the variance-covariance matrix of a dataset with the eigenvalues equal to the Lagrange multipliers.
- A dataset with all PC's applied has a variance-covariance matrix whose non-diagonal entires are zero. That is, the all variables are uncorrelated.

$$\min \quad \mathbf{W}^T\mathbf{\Sigma}\mathbf{W} s.t. \mathbf{W}^T\mathbf{W} = 1$$

R: PCA

```
mtcars_temp = subset(mtcars,select=c(mpg,wt, disp, qsec))
mtcars_c = scale(mtcars_temp) # Normalize
out_pca = princomp(mtcars_c) # PCA

summary(out_pca)
```

```
Importance of components:
                         Comp.1    Comp.2    Comp.3     Comp.4
Standard deviation     1.681763 0.9178832 0.3865919 0.23390037
Proportion of Variance 0.729891 0.2174218 0.0385686 0.01411855
Cumulative Proportion  0.729891 0.9473129 0.9858814 1.00000000
```

```
W = out_pca$loadings;W # Linear combinations
```

```

Loadings:
     Comp.1 Comp.2 Comp.3 Comp.4
mpg   0.555         0.762  0.327
wt   -0.538 -0.376  0.105  0.747
disp -0.561         0.639 -0.523
qsec  0.297 -0.922        -0.247

               Comp.1 Comp.2 Comp.3 Comp.4
SS loadings      1.00   1.00   1.00   1.00
Proportion Var   0.25   0.25   0.25   0.25
Cumulative Var   0.25   0.50   0.75   1.00
```

```
# Dimension Reduction to PC1
X = as.matrix(mtcars_c)
XW = X%*%W ; XW[,1]
```

```
          Mazda RX4       Mazda RX4 Wag          Datsun 710
          0.5012988           0.4542238           1.4243651
     Hornet 4 Drive   Hornet Sportabout             Valiant
          0.2627126          -0.9732059           0.1029417
         Duster 360           Merc 240D            Merc 230
         -1.6460261           1.1495838           1.5324653
           Merc 280           Merc 280C          Merc 450SE
          0.1561807           0.1268289          -1.0870481
         Merc 450SL         Merc 450SLC  Cadillac Fleetwood
         -0.7841288          -0.9386639          -3.0797248
Lincoln Continental   Chrysler Imperial            Fiat 128
         -3.1475747          -2.6838810           2.6502919
        Honda Civic      Toyota Corolla       Toyota Corona
          2.6431662           3.0948242           1.4026790
   Dodge Challenger         AMC Javelin          Camaro Z28
         -1.1468472          -0.9930228          -1.9126488
   Pontiac Firebird           Fiat X1-9       Porsche 914-2
         -1.3256578           2.2299109           1.4452233
       Lotus Europa      Ford Pantera L        Ferrari Dino
          2.3423493          -1.4698954           0.2075584
      Maserati Bora          Volvo 142E
         -1.5204547           0.9821761
```

```
# Covariance Matrix
var(X)
```

```
            mpg         wt       disp       qsec
mpg   1.0000000 -0.8676594 -0.8475514  0.4186840
wt   -0.8676594  1.0000000  0.8879799 -0.1747159
disp -0.8475514  0.8879799  1.0000000 -0.4336979
qsec  0.4186840 -0.1747159 -0.4336979  1.0000000
```

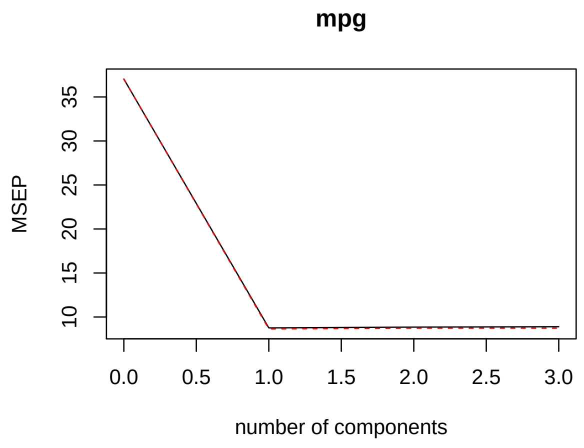
```
round(var(XW),7)
```

```
         Comp.1    Comp.2    Comp.3    Comp.4
Comp.1 2.919564 0.0000000 0.0000000 0.0000000
Comp.2 0.000000 0.8696873 0.0000000 0.0000000
Comp.3 0.000000 0.0000000 0.1542744 0.0000000
Comp.4 0.000000 0.0000000 0.0000000 0.0564742
```

R: Principle Component Regression

- Pick `ncomp` that minimize MSEP.

```r
library(pls)
train.i = traindex(mtcars_temp)

mtcars_train = mtcars_temp[train.i,]
mtcars_test = mtcars_temp[-train.i,2:4]
y_test = mtcars_temp[-train.i,1]

pcr.fit=pcr(mpg~., data=mtcars_train,scale=TRUE, validation="CV")
validationplot(pcr.fit,val.type="MSEP")
```



```r
pcr.pred=predict(pcr.fit,mtcars_test,ncomp=1)
mean((pcr.pred-y_test)^2)
```

```
[1] 8.623331
```

```r
pcr.fit=pcr(y~x,scale=TRUE,ncomp=1)
summary(pcr.fit)
```

```
Data:   X dimension: 263 19
    Y dimension: 263 1
Fit method: svdpc
Number of components considered: 1
TRAINING: % variance explained
   1 comps
X    38.31
y    40.63
```

Spline

- The truth is that association between variables are hardly ever linear. However, polynomial regression often cannot capture all features and associations among variables. To achieve this, fit linear regression piece-wise, and fit different polynomials for each pieces, then smooth the entire curve. Thus, spline works on one continuous predictor.
- Keep bias-variance trade off in mind. Linear model is highly biased, but if we attempt to increase the flexibility of the model, the bias decreases but variability increases. Add more flexibility until the bias is low enough yet the variability is not too high.
- The junctions of pieces is called "knot".
- Step functions are used to divide the predictor variables and add knots. `cut` command divides the predictor variables into K many equal length intervals and assign each observation into the appropriate interval. Use `break` option to make your own cut if desired.
- Spline is the combine step functions and polynomial regression. One draw back is that piecewise polynomials can create a discontinuity at knots. Thus, we smooth the curve.
- `bs(X, knot)`: Any degree splines e.g. Linear splines and cubic splines (piece-wise cubic polynomials). A cubic spline with K knots has K+4 df.
- `ns(X, df)`: Natural cubic splines. A natural cubic spline extrapolates linearly beyond the boundary knots. This adds 4 extra constraints, and allows us to put more internal knots for the same degrees of freedom as a regular cubic spline. Thus, natural splines with K knots has K df.
- `s(X, df)`: Smoothing splines, fit model via minimize $SSE + \lambda \int g''(t)dt$. Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen. Here, `smooth.spline()` function fit a smoothing spline, and we choose df instead of λ.
- `lo(X, span)`: Local regression spline, also called loess. We fit separate linear fits over the range of the predictor variable by weighted least squares. Highly flexible. Use `loess()` function as an alternative.

R: Step function

```r
library(splines)
m = lm(Sepal.Width~cut(Sepal.Length, 3), data=iris)
summary(m)
```

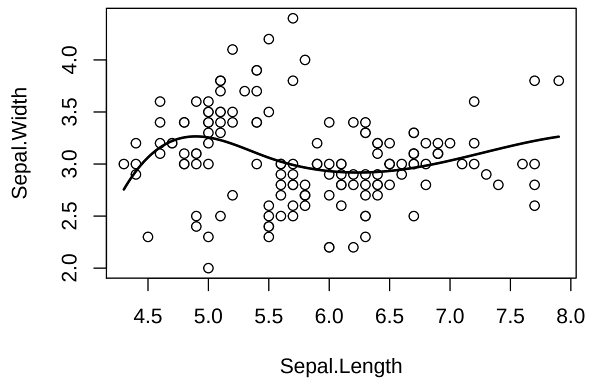
```

Call:
lm(formula = Sepal.Width ~ cut(Sepal.Length, 3), data = iris)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.2000 -0.2075 -0.0100  0.2000  1.4761 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                    3.20000    0.05453  58.686  < 2e-16 ***
cut(Sepal.Length, 3)(5.5,6.7] -0.27606    0.07378  -3.741 0.000262 ***
cut(Sepal.Length, 3)(6.7,7.9] -0.09000    0.10837  -0.830 0.407618    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4188 on 147 degrees of freedom
Multiple R-squared:  0.08901,	Adjusted R-squared:  0.07662 
F-statistic: 7.182 on 2 and 147 DF,  p-value: 0.001057
```

R: Spline

```r
library(splines)
grids = seq(min(iris$Sepal.Length), max(iris$Sepal.Length), by=0.01)

m = lm(Sepal.Width~bs(Sepal.Length, knots=c(4.3, 5.5, 6.7, 7.9)), data=iris)
summary(m)
```

```

Call:
lm(formula = Sepal.Width ~ bs(Sepal.Length, knots = c(4.3, 5.5,
    6.7, 7.9)), data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.25514 -0.22417 -0.00521  0.24486  1.40843

Coefficients: (2 not defined because of singularities)
                                                 Estimate Std. Error
(Intercept)                                       3.26262    0.31055
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))1 -0.50691    0.40431
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))2  0.28136    0.34303
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))3 -0.35769    0.41998
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))4 -0.38126    0.31564
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))5 -0.06646    0.55917
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))6       NA         NA
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))7       NA         NA
                                                 t value Pr(>|t|)
(Intercept)                                       10.506   <2e-16 ***
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))1  -1.254    0.212
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))2   0.820    0.413
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))3  -0.852    0.396
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))4  -1.208    0.229
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))5  -0.119    0.906
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))6      NA       NA
bs(Sepal.Length, knots = c(4.3, 5.5, 6.7, 7.9))7      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.423 on 144 degrees of freedom
Multiple R-squared:  0.08962,	Adjusted R-squared:  0.05801
F-statistic: 2.835 on 5 and 144 DF,  p-value: 0.01793
```

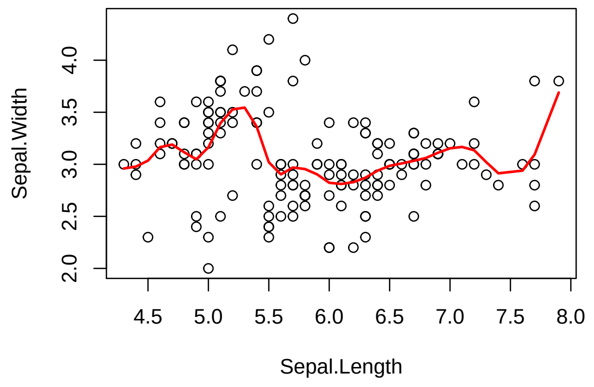
```r
pred=predict(m,newdata=list(Sepal.Length=grids),se=T)
plot(Sepal.Width~Sepal.Length, data=iris);lines(grids,pred$fit,lwd=2)
```



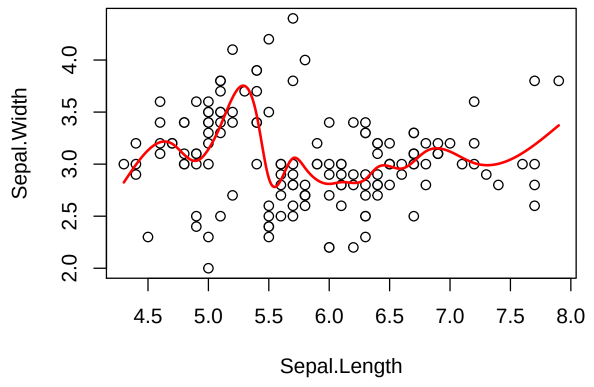
```
fit=smooth.spline(iris$Sepal.Length,iris$Sepal.Width,cv=T);fit$df
```

```
[1] 16.04326
```

```
plot(Sepal.Width~Sepal.Length, data=iris);lines(fit,col="red",lwd=2)
```



```
m=lm(Sepal.Width~ns(Sepal.Length,df=16),data=iris)
pred=predict(m,newdata=list(Sepal.Length=grids),se=T)
plot(Sepal.Width~Sepal.Length, data=iris);lines(grids, pred$fit,col="red",lwd=2)
```



Generalized Additive Models

- Allows flexible non-linrarities in multiple variables, but retains the additive structure of linear models. For instance, We can fit multiple splines or local regression.
- Use `anova()` to compare models. In the case below, model2 is the best.

```r
library(gam)
m1 = gam(Sepal.Width~s(Sepal.Length,df=16), data=iris)
m2 = gam(Sepal.Width~s(Sepal.Length,df=16)+s(Petal.Width,3), data=iris)
m3 = gam(Sepal.Width~s(Sepal.Length,df=16)+s(Petal.Width,3)+Petal.Length, data=iris)
anova(m1,m2,m3,test="F")
```

```
Analysis of Deviance Table

Model 1: Sepal.Width ~ s(Sepal.Length, df = 16)
Model 2: Sepal.Width ~ s(Sepal.Length, df = 16) + s(Petal.Width, 3)
Model 3: Sepal.Width ~ s(Sepal.Length, df = 16) + s(Petal.Width, 3) +
    Petal.Length
  Resid. Df Resid. Dev     Df Deviance      F    Pr(>F)
1       133    21.5745
2       130    10.2482 3.0002  11.3264 52.179 < 2.2e-16 ***
3       129     9.3333 1.0000   0.9149 12.645 0.0005274 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(m2)
```

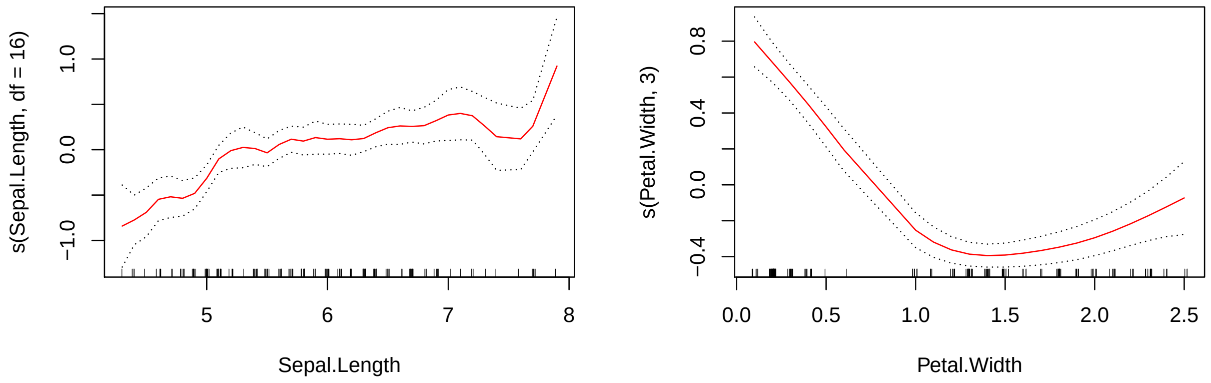
```

Call: gam(formula = Sepal.Width ~ s(Sepal.Length, df = 16) + s(Petal.Width,
    3), data = iris)
Deviance Residuals:
     Min       1Q   Median       3Q      Max
-0.72047 -0.14473 -0.01395  0.17672  0.77893

(Dispersion Parameter for gaussian family taken to be 0.0788)

    Null Deviance: 28.3069 on 149 degrees of freedom
Residual Deviance: 10.2482 on 130 degrees of freedom
AIC: 65.1512

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                          Df  Sum Sq Mean Sq F value Pr(>F)
s(Sepal.Length, df = 16)   1  0.3696  0.3696  4.6879 0.0322 *
s(Petal.Width, 3)          1  7.0604  7.0604 89.5625 <2e-16 ***
Residuals                130 10.2482  0.0788
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Anova for Nonparametric Effects
                         Npar Df Npar F     Pr(F)
(Intercept)
s(Sepal.Length, df = 16)      15  3.023 0.0003461 ***
s(Petal.Width, 3)              2 72.446 < 2.2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

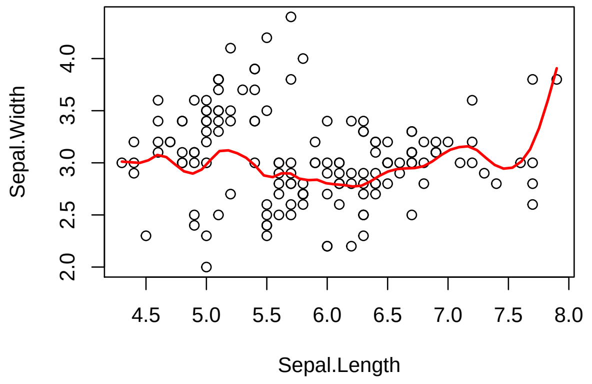
```
# training MSE
mean((iris$Sepal.Width-predict(m2,newdata=iris))^2)
```

```
[1] 0.06832059
```

```
library(akima)
plot(m2, se=TRUE, col="red")
```



```
grids1 = seq(min(iris$Sepal.Length),max(iris$Sepal.Length), length.out = 50)
grids2 = seq(min(iris$Petal.Length),max(iris$Petal.Length), length.out = 50)
grids3 = seq(min(iris$Petal.Width),max(iris$Petal.Width), length.out = 50)
irisdf = data.frame(Sepal.Length=grids1, Petal.Length=grids2,Petal.Width=grids3)
pred = predict(m2, newdata=irisdf)
plot(Sepal.Width~Sepal.Length, data=iris);lines(grids1, pred,col="red",lwd=2)
```



Logistic Regression

- Probabilistic, supervised learning, classification
- Linear regression produce a result ranged in $(-\infty, \infty)$, which is not appropriate here. Thus, when binary outcome is present, logistic regression is appropriate.
- $\theta(x_i) = \frac{exp(\beta_0+\beta_1 x_i)}{1+exp(\beta_0+\beta_1 x_i)} = \frac{1}{1+exp(-\{\beta_0+\beta_1 x_i\})}$
- $\log(\frac{\theta(x_i)}{1-\theta(x_i)}) = \beta_0 + \beta_1 x_i$ where $\log(\frac{\theta(x_i)}{1-\theta(x_i)})$ is called logit.
- $\log(\frac{\hat{\theta(x_i)}}{1-\hat{\theta(x_i)}}) = \hat{\beta_0} + \hat{\beta_1} x_i$
- We use maximum likelihood estimates for coefficients, but no closed-form solutions exist. Thus, we take advantage of computation results.
- "For every unit increase in X, the odds that the characteristic is present is multiplied by $exp(\beta_1)$."
- Recall the PDF of binomial: $P(Y_i = y_i|x_i) = \binom{m_i}{y_i}\theta(x_i)(1-\theta(x_i))^{m_i-y_i}$
- Recall the log likelihood of binomial : $\log(L) = \sum_{i=1}^{n}(y_i(\beta_0 + \beta_1 x_i) - m_i \log(1 + exp(\beta_0 + \beta_1 x_i)) + \log\binom{m_i}{y_i})$.

Goodness of fit test

- H_o : The logistic regression is appropriate vs. H_a : Not appropriate.
- We use two log likelihood : $\log(L_M)$ vs. $\log(L_S)$ where M refers the logistic regression model and S refers to the saturated model, a model with a theoretically perfect fit. Thus, $\hat{y_i} = y_i$ under the saturated model.
- Set $\hat{y_i} = \hat{\theta_M}(x_i)m_i$

1. Deviance: $G^2 = 2(\log(L_S) - \log(L_M)) = 2\sum_{i=1}^{n}[y_i \log(\frac{y_i}{\hat{y_i}}) + (m_i - y_i)\log(\frac{m_i-y_i}{m_i-\hat{y_i}})] \sim \chi^2_{n-p-1}$. IF the model is appropriate, then G^2 is smaller so we fail to reject the null. P-value $= P(\chi^2_{n-p-1} > G^2_{obs})$
2. Pearson χ^2 statistic: $\chi^2 = \sum \frac{(y_i/m_i - \hat{\theta}(y_i))^2}{\hat{\theta}(y_i)(1-\hat{\theta}(y_i))/m_i} \sim \chi^2_{n-p-1}$
3. R-squared: $R^2_{dev} = 1 - \frac{G^2_{H_a}}{G^2_{H_o}}$

Comparing models

- $H_o : \theta(x) = \frac{1}{1+exp(-\beta_0)}$ vs. $H_a : \theta(x) = \frac{1}{1+exp(-\{\beta_0+\beta_1 x\})}$
- $G^2_{H_o} - G^2_{H_a} \sim \chi^2_{df_1-df_2}$ where $df_1 = n -$ (num. of predictors in H_0) $- 1$ and $df_2 = n -$ (num. of predictors in H_a) $- 1$

Marginal model plot

Compare the following two models, and determine wheather the logistic regression is appropriate. If two models are significantly different, then the logistic regression ins not appropriate.

1. Parametric model : $\theta(x_i) = \frac{1}{1+exp(-\{\beta_0+\beta_1 x_i\})}$
2. Nonparametric model : $\theta(x) = f(x_1, \ldots, x_p)$. For the model with p predictors, we need p many marginal model plots. If any of them show discrepancy from parametric model, then the parametric model (logistic regression) is not appropriate.

R: Logistic Regression

- Some predictor variables cause an error:`glm.fit: algorithm did not converge` when the sample size is not enough.
- If `Gdiff < 0.05`, reject H_o and full model is better.

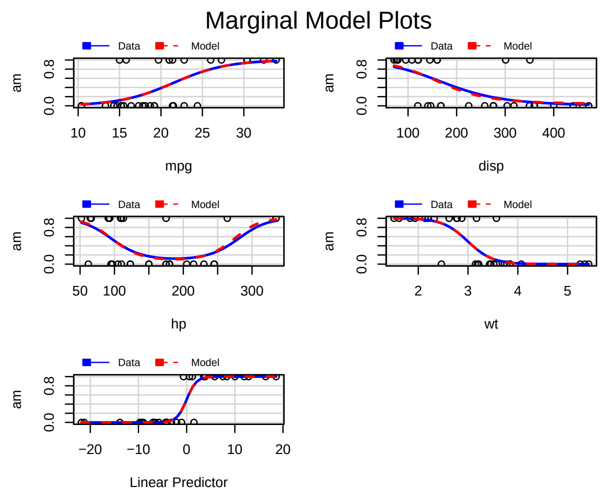
```
m = glm(am~mpg+disp+hp+wt, family=binomial(), data=mtcars)
summary(m)
```

```
Call:
glm(formula = am ~ mpg + disp + hp + wt, family = binomial(),
    data = mtcars)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.84992  -0.15966  -0.00615   0.01257   1.46081

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -18.48207   40.90451  -0.452    0.651
mpg           1.13503    1.55720   0.729    0.466
disp         -0.02588    0.04087  -0.633    0.527
hp            0.10871    0.09837   1.105    0.269
wt           -4.80560    3.97978  -1.208    0.227

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.230  on 31  degrees of freedom
Residual deviance:  8.162  on 27  degrees of freedom
AIC: 18.162

Number of Fisher Scoring iterations: 9
```

```
Gdiff = m$null.deviance-m$deviance
pchisq(Gdiff,4,lower=FALSE)
```

```
[1] 4.498811e-07
```

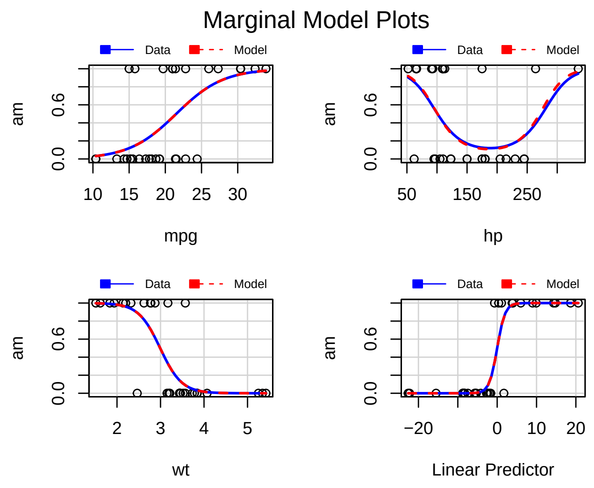
```
library(alr3)
mmps(m)
```



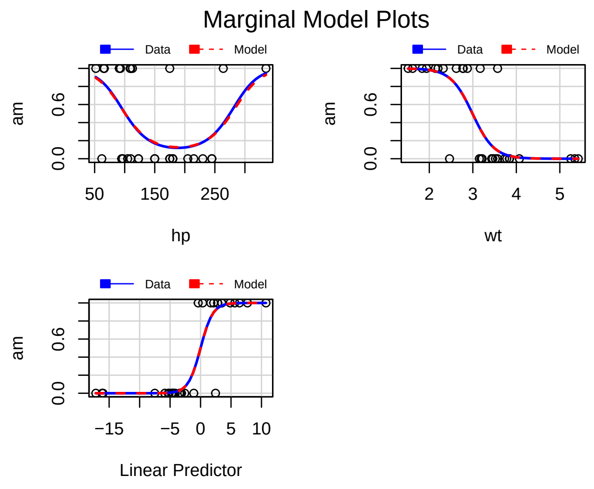
R: Comparing models

- Here, failed to reject Null, `m3` model is better.
- Also, one of the plot shows a descrepancy. Logistic regression for `m2` might not be appropriate.

```
m3 = glm(am~mpg+hp+wt, family=binomial(), data=mtcars)
mmps(m3)
```



```
m2 = glm(am~hp+wt, family=binomial(), data=mtcars)
mmps(m2)
```



```
anova(m3,m2,test="Chisq")
```

```
Analysis of Deviance Table

Model 1: am ~ mpg + hp + wt
Model 2: am ~ hp + wt
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        28     8.7661
2        29    10.0591 -1   -1.293   0.2555
```

R: Prediction

```
set.seed(1)
train.i = traindex(mtcars, trainn = 25)

Y_train = mtcars[train.i, c("am")]
Y_test = mtcars[-train.i, c("am")]
X_train = mtcars[train.i, c("mpg","hp","wt")];X_train = cbind(X_train,Y_train)
X_test = mtcars[-train.i, c("mpg","hp","wt")]

m = glm(Y_train~mpg+hp+wt, family = binomial(), data=X_train)
Y_pred = ifelse(predict(m, X_test, type="response")>=0.5,1,0)
table(Y_test, Y_pred)
```

```
      Y_pred
Y_test 0 1
     0 4 0
     1 0 3
```

Bayes Classifier

- $P() = \frac{P()P()}{P()}$

K Nearest Neighbor

- Non-probablistic, supervised learning, classification
- Judge the category of a point via the category of K many Euclidian nearest others.
- No assumption on distribution of X, but they have to be numeric continuous.
- Draw decision boundary according to K many closest neighbors based on Euclidian distance.
- Make sure to standarize the predictor variable X because the algorithm takes Euclidan distance to determine the line, and standarizing increase the accuracy of the results.
- Highly non-linear boundry for smaller K.

```r
library(class)
set.seed(1)
test.i = sample(1:nrow(Heart),50, replace=F)

# Standarize predictors (continuous)
Xs = scale(cbind(Heart$Age, Heart$MaxHR))
Xs_test = Xs[test.i,]
Xs_train = Xs[-test.i,]
Y_test = Heart$AHD[test.i]
Y_train = Heart$AHD[-test.i]
knn_output = knn(Xs_train, Xs_test, Y_train, k=1)

# MSE
mean(knn_output!=Y_test)
```

```
[1] 0.4
```

```r
# Confusion Matrix
table(knn_output, Y_test)
```

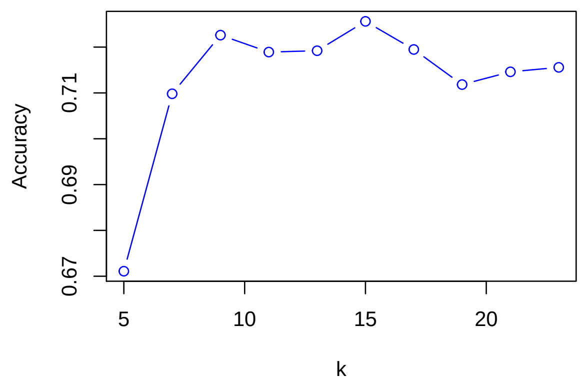
```
          Y_test
knn_output No Yes
       No  18   9
       Yes 11  12
```

Find the best K

```
library(caret)
library(e1071)
Heart_train = cbind(Xs_train,Y_train);colnames(Heart_train) = c("Age","MaxHR","AHD")
ctrl = trainControl(method="repeatedcv", repeats=13)
knnfit = train(as.factor(AHD)~., data=Heart_train, trControl=ctrl, method="knn", preProcess = c("center
knnfit
```

```
k-Nearest Neighbors

253 samples
  2 predictor
  2 classes: '1', '2'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold, repeated 13 times)
Summary of sample sizes: 228, 227, 228, 228, 228, 228, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.6710947  0.3378439
   7  0.7098107  0.4156400
   9  0.7226105  0.4411433
  11  0.7189132  0.4332314
  13  0.7192101  0.4333105
  15  0.7256154  0.4473893
  17  0.7194822  0.4360253
  19  0.7118235  0.4208896
  21  0.7146055  0.4269985
  23  0.7155789  0.4287789

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 15.
```

```
knnfit_val = data.frame(knnfit[4])
plot(knnfit_val[,1], knnfit_val[,2], type="b", col="blue", xlab="k", ylab="Accuracy")
```



```
maxacu = which(knnfit$results$Accuracy==max(knnfit$results$Accuracy));maxacu
```

```
[1] 6
```

Repeat with the best K

```
knn_output = knn(Xs_train, Xs_test, Y_train, k=maxacu, prob=T)
# MSE
mean(knn_output!=Y_test)
```

```
[1] 0.32
```

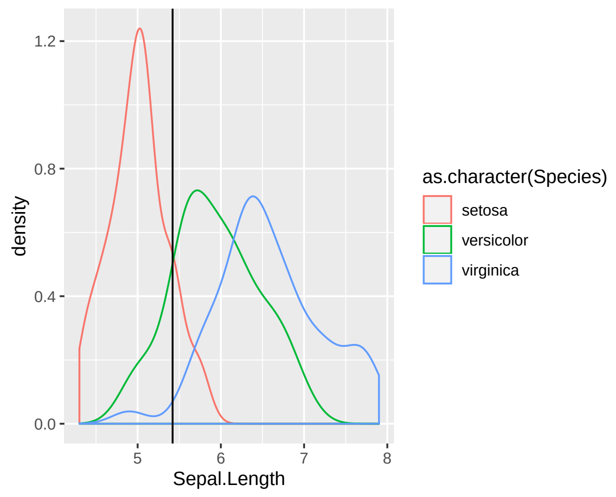
```
# Confusion Matrix
table(knn_output, Y_test)
```

```
          Y_test
knn_output No Yes
       No  20   7
       Yes  9  14
```

Linear Descriminant Analysis

- non-probablistic, supervised learning, classification.
- Finds the best place to make the best split linear boundary between two (or more) distributions. Relies on the Bayes Classifier. Tries to find the K-dimenstional projection that creates the greatest between group separation.
- Assumption: Normaility and small sample size, and same variance among all group (cateogry).
- Make the dataset linearly independent. LD1, LD2... are the coefficients such that makes the dataset linearly independent i.e. eigenvectors.
- Dimension reduction.
- The devision line $x = \frac{\mu_1+\mu_2}{2}$

```
ggplot(iris) + geom_density(mapping=aes(x=Sepal.Length, color=as.character(Species)))+ geom_vline(xinte
```



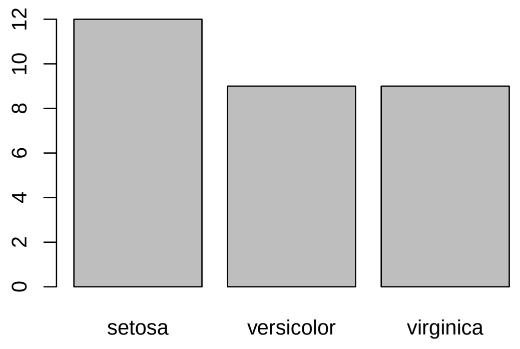
- The lda$svd represents the eigenvalues, and bigger the better split the data.
- LD1 and LD2 on the bottom represents the trace. Bigger the more important role.
- LD1, LD2 with coefficients are the eigenvectors. Make sure that they are all standarized.

```
library(MASS)
set.seed(1)
test.i = sample(1:nrow(iris),30, replace=F)
X_train = iris[-test.i, 1:4]
X_test = iris[test.i, 1:4]
Y_train = iris[-test.i, 5]
Y_test = iris[test.i, 5]

# LDA
model_lda = lda(Y_train~X_train$Sepal.Length+X_train$Sepal.Width);model_lda
```

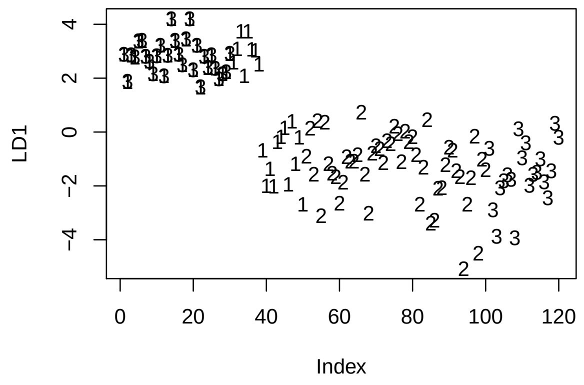
```
Call:
lda(Y_train ~ X_train$Sepal.Length + X_train$Sepal.Width)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3166667  0.3416667  0.3416667

Group means:
```

```
           X_train$Sepal.Length X_train$Sepal.Width
setosa                 5.018421            3.418421
versicolor             5.926829            2.760976
virginica              6.560976            2.960976

Coefficients of linear discriminants:
                           LD1        LD2
X_train$Sepal.Length -2.074508 -0.8362372
X_train$Sepal.Width   2.846646 -2.0806011

Proportion of trace:
   LD1    LD2
0.9602 0.0398
```

```
# Take LD1(The first split)
LD1 = predict(model_lda)$x[,1]

plot(Y_test)
```



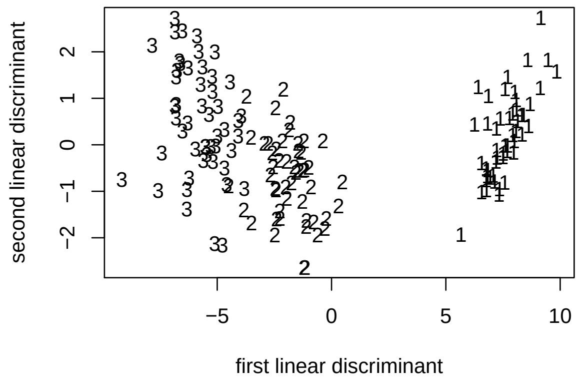
```
plot(LD1, type="n");text(LD1,labels=unclass(iris$Species))
```



```
m3 = lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = iris)
m3
```

```
Call:
lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
    data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica 
 0.3333333  0.3333333  0.3333333 

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785

Proportion of trace:
   LD1    LD2 
0.9912 0.0088 
```

```
LD1<-predict(m3)$x[,1]
LD2<-predict(m3)$x[,2]
plot(LD1,LD2,xlab="first linear discriminant",ylab="second linear discriminant",type="n")
text(cbind(LD1,LD2),labels=unclass(iris$Species))
```



```
head(2.105107+0.8293776*iris$Sepal.Length+1.5344731*iris$Sepal.Width-2.2012117*iris$Petal.Length-2.8104
```

```
[1] 8.061800 7.128688 7.489828 6.813201 8.132310 7.701947
```

```
head(LD1)
```

```
       1        2        3        4        5        6 
8.061800 7.128688 7.489828 6.813201 8.132309 7.701947 
```

```r
cor(iris[,1],LD1)
```

```
[1] -0.7918878
```

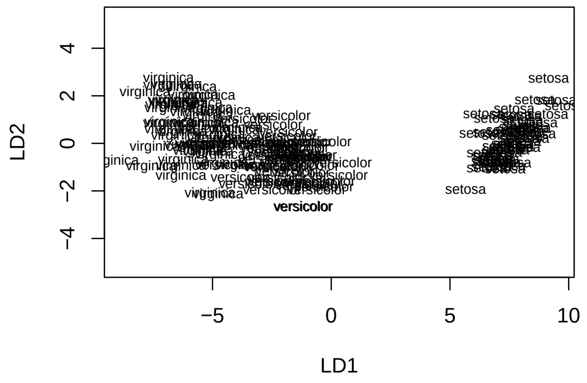
```r
m3$svd
```

```
[1] 48.642644  4.579983
```

```r
iris.lda<-lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,  data = iris)
iris.lda
```

```
Call:
lda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
    data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785

Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

```r
plot(iris.lda)
```



```r
iris.lda$svd
```

```
[1] 48.642644  4.579983
```

```r
iris.lda$counts
```

```
    setosa versicolor  virginica
        50         50         50
```

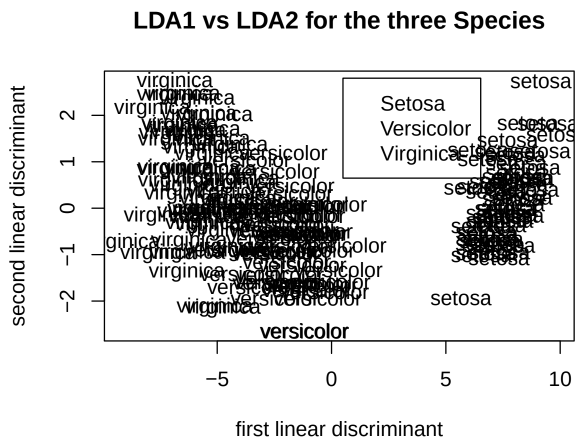
```
iris.lda$means
```

```
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026
```

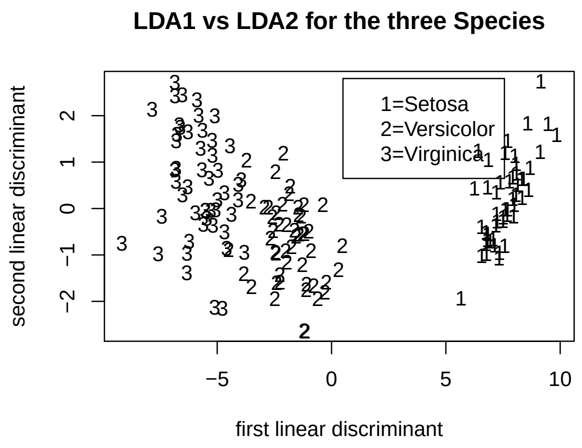
```
iris.lda$lev
```

```
[1] "setosa"     "versicolor" "virginica"
```

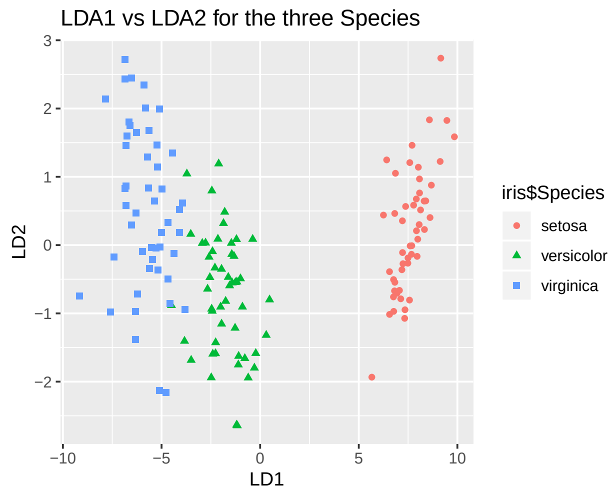
```
# Plots:
LD1<-predict(iris.lda)$x[,1]
LD2<-predict(iris.lda)$x[,2]

plot(LD1,LD2,xlab="first linear discriminant",ylab="second linear discriminant",col=2:4,type="n",main="
text(cbind(LD1,LD2),labels=iris$Species)
legend(0.5,2.8,legend=c("Setosa","Versicolor","Virginica"))
```



```
plot(LD1,LD2,xlab="first linear discriminant",ylab="second linear discriminant",col=2:4,type="n",main="
text(cbind(LD1,LD2),labels=unclass(iris$Species))
legend(0.5,2.8,legend=c("1=Setosa","2=Versicolor","3=Virginica"))
```



```
# 1="setosa"
# 2="versicolor"
# 3="virginica"

qplot(x = LD1, y = LD2, colour = iris$Species,shape = iris$Species,main="LDA1 vs LDA2 for the three Spe
```



```
# Group centroids

sum(LD1*(iris$Species=="setosa"))/sum(iris$Species=="setosa")
```

```
[1] 7.6076
```

```
sum(LD2*(iris$Species=="setosa"))/sum(iris$Species=="setosa")
```

```
[1] 0.215133
```

```
sum(LD1*(iris$Species=="versicolor"))/sum(iris$Species=="versicolor")
```

```
[1] -1.825049
```

```
sum(LD2*(iris$Species=="versicolor"))/sum(iris$Species=="versicolor")
```

```
[1] -0.7278996
```

```
sum(LD1*(iris$Species=="virginica"))/sum(iris$Species=="virginica")
```

```
[1] -5.78255
```

```
sum(LD2*(iris$Species=="virginica"))/sum(iris$Species=="virginica")
```

```
[1] 0.5127666
```

```
iris.predict<-predict(iris.lda,iris[,1:4])
iris.classify<-iris.predict$class
iris.classperc<-sum(iris.classify==iris[,5])/150
iris.classperc
```

```
[1] 0.98
```

```
table(Original=iris$Species,Predicted=predict(iris.lda)$class)
```

```
            Predicted
Original     setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          1        49
```

Quadratic Discriminant Analysis

- In stead of linear boundary, we use non-linear boundary.

```
m4 = qda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = iris)
m4
```

```
Call:
qda(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
    data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026
```

Support Vector Machine

K-Means Clustering

Kernelized Clustering

EM Type Algorithm

Decision Trees

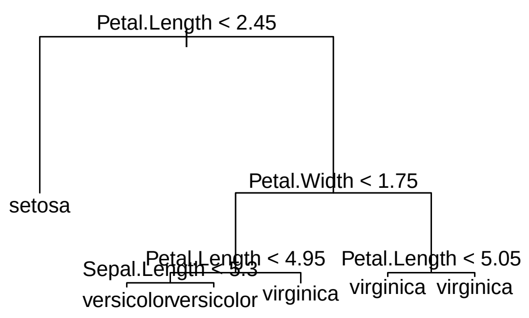
- Partition data points. Determine the value of response variable (if continuous, the mean of the response) according to which partition that new predictor belongs to (looks like a house diagram). If Y is continuous, it is called regression tree and if Y is cateogorical, it is called classification tree.
- Determine the splits lines that generate the lowest MSE. As we have more splits, the new value of Y can be determined on the tree structure, or "nested if" structure.
- Pruning tree: A large tree can be over-fitting, thus pruning tree allows cutting off some of the terminal nodes. Find the best pruning by cross validation.
- If the relationship between the predictors and response is linear, then linear regression is better; if the relationship is "rectangle" shapes, tree performs better.
- Easily interpretable (e.g. If weight is above 40kg, height is higher than 140cm in average). However, suffer from high variance. Thus, use random forest to resolve this issue.

R: Tree

```r
library(tree)
train.i = traindex(iris)
iris_train = iris[train.i,]
iris_test = iris[-train.i,1:4]
Y_test = factor(iris[-train.i,5])

tree1 = tree(Species~., data=iris_train)
summary(tree1)
```

```
Classification tree:
tree(formula = Species ~ ., data = iris_train)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"  "Sepal.Length"
Number of terminal nodes:  6
Residual mean deviance:  0.1329 = 17.14 / 129
Misclassification error rate: 0.02963 = 4 / 135
```

```r
plot(tree1);text(tree1)
```



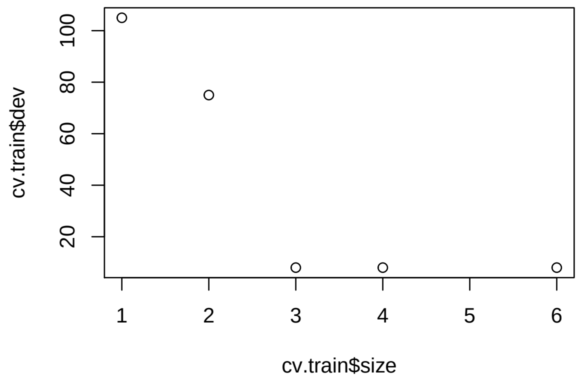
```
# Prediction
preds=predict(tree1,newdata=iris_test, type="class")
table(Y_test, preds)
```

```
            preds
Y_test        setosa versicolor virginica
  setosa           5          0         0
  versicolor       0          6         0
  virginica        0          0         4
```

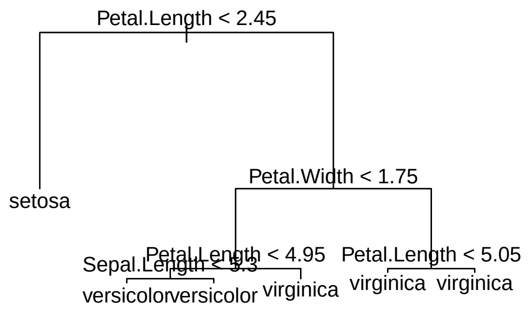
R: Pruning Tree

- Here, it shows that pruning with 5 branches is appropriate.

```
cv.train=cv.tree(tree1,FUN=prune.misclass)
plot(cv.train$dev~cv.train$size)
```



```
pruned.fit=prune.misclass(tree1,best=5)

plot(pruned.fit)
text(pruned.fit,pretty=TRUE)
```



```
summary(pruned.fit)
```

```
Classification tree:
tree(formula = Species ~ ., data = iris_train)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"  "Sepal.Length"
Number of terminal nodes:  6
Residual mean deviance:  0.1329 = 17.14 / 129
Misclassification error rate: 0.02963 = 4 / 135
```

```
# Prediction
pred.prune=predict(pruned.fit,newdata=iris_test,type="class")
table(Y_test, pred.prune)
```

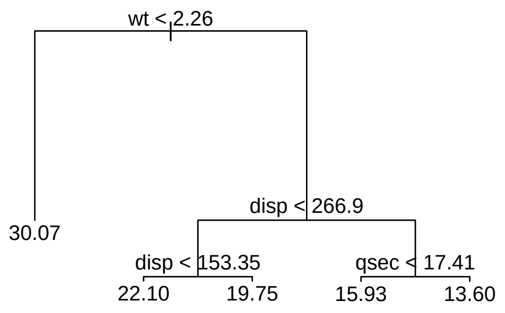
```
            pred.prune
Y_test        setosa versicolor virginica
  setosa           5          0         0
  versicolor       0          6         0
  virginica        0          0         4
```

R: Regression Tree

```
mtcars_temp = subset(mtcars,select=c(mpg,wt, disp, qsec))
tree2=tree(mpg~., data=mtcars_temp)
summary(tree2)
```

```
Regression tree:
tree(formula = mpg ~ ., data = mtcars_temp)
Number of terminal nodes:  5
Residual mean deviance:  5.104 = 137.8 / 27
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -4.067  -1.637   0.100   0.000   1.338   3.833
```

```
plot(tree2)
text(tree2,pretty=0)
```



Random Forest

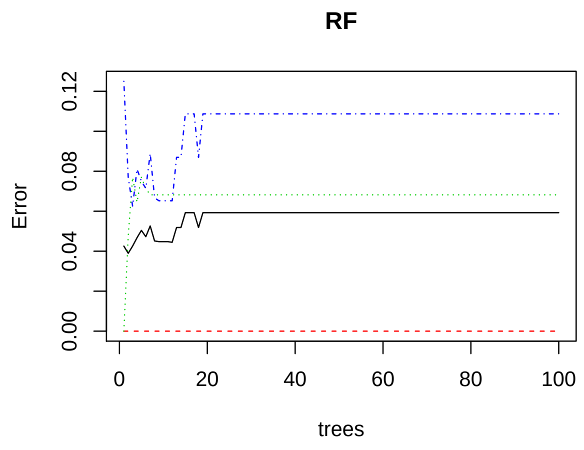
- Use bagging: Bootstrap + averaging. That is, generate B different bootstrapped training dataset. Train the statistical learning method on each of the B training datasets, and obtain the prediction, then take the average. In this case, construct B different trees using B bootstrapped dataset, then take the average of the results.
- If continuous, average all predictions from all B trees. If Classification, majority vote among all B trees. These trees are not pruned, so each individual tree has high variance but low bias. Averaging these trees reduces variance, and thus lowering both variance and bias can be achieved.
- Two methods for prediction: Record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one (majority vote). Or, if our classifier produces probability estimates we can just average the probabilities and then predict to the class with the highest probability.
- Bagging improves prediction accuracy at the expense of interpretability. But, we can use relative influence plots to see the contributions of each variables to the model. Larger the more influential.
- Random forest: Build a number of decision trees on bootstrapped training sample, but when building these trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. Only m predictors are used for the sake of "de-correlation" of the model; if all variables are used, each models for each bootstrapped dataset will be similar to each other, hence highly correlated. Averaging many highly correlated quantities does not lead to a large variance reduction.

R: RF, `mtry=4`

```
library(randomForest)

RF=randomForest(Species~.,data=iris_train,mtry=4,importance=TRUE, ntree=100)
summary(RF)
```

```
                Length Class  Mode
call              6    -none- call
type              1    -none- character
predicted       135    factor numeric
err.rate        400    -none- numeric
confusion        12    -none- numeric
votes           405    matrix numeric
oob.times       135    -none- numeric
classes           3    -none- character
importance       20    -none- numeric
importanceSD     16    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           14    -none- list
y               135    factor numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

```
plot(RF)
```



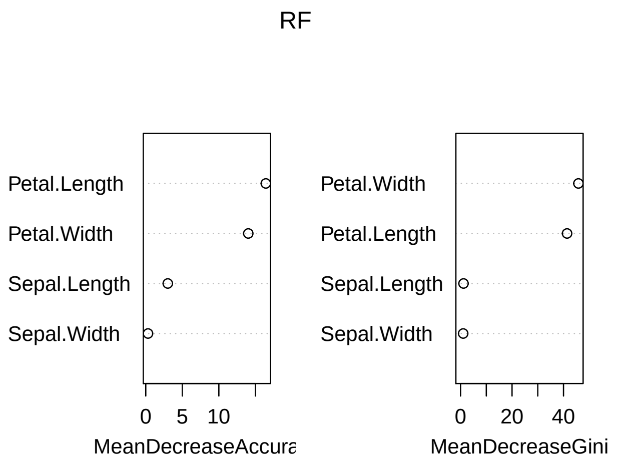
```r
print(RF)
```

```
Call:
 randomForest(formula = Species ~ ., data = iris_train, mtry = 4,      importance = TRUE, ntree = 100)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 4

        OOB estimate of  error rate: 5.93%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         45          0         0  0.00000000
versicolor      0         41         3  0.06818182
virginica       0          5        41  0.10869565
```

```r
importance(RF)
```

```
                setosa versicolor  virginica MeanDecreaseAccuracy
Sepal.Length  0.000000   3.534282 -0.1174515            3.0153919
Sepal.Width   0.000000  -1.549744  1.7981879            0.3076241
Petal.Length 10.658224  16.926435 13.5961096           16.4290251
Petal.Width   9.700194  15.143134 11.7300800           14.0252231
             MeanDecreaseGini
Sepal.Length         1.150223
Sepal.Width          1.013324
Petal.Length        41.384468
Petal.Width         45.788132
```

```r
varImpPlot (RF)
```



```r
preds = predict(RF,newdata=iris_test)
table(Y_test, preds)
```

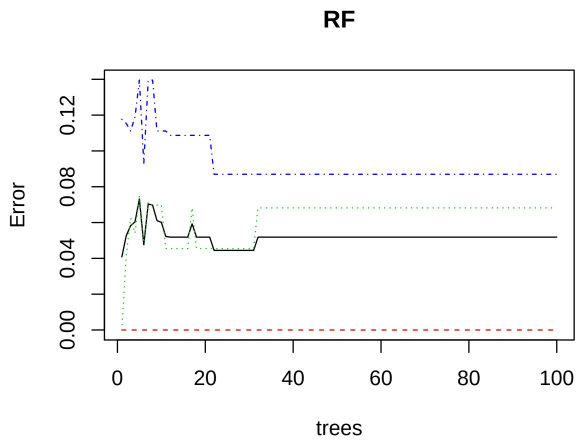
```
            preds
Y_test       setosa versicolor virginica
  setosa          5          0         0
  versicolor      0          6         0
  virginica       0          0         4
```

R: RF, `mtry=2`

```r
RF=randomForest(Species~.,data=iris_train,mtry=2,importance=TRUE, ntree=100)
summary(RF)
```

```
                Length Class  Mode
call              6    -none- call
type              1    -none- character
predicted       135    factor numeric
err.rate        400    -none- numeric
confusion        12    -none- numeric
votes           405    matrix numeric
oob.times       135    -none- numeric
classes           3    -none- character
importance       20    -none- numeric
importanceSD     16    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           14    -none- list
y               135    factor numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

```
plot(RF)
```



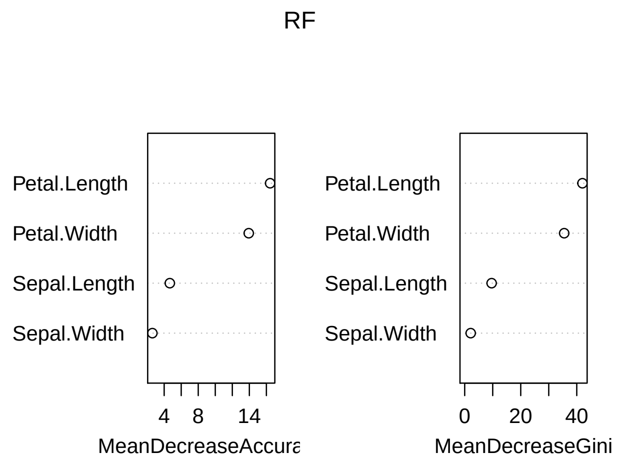
```
print(RF)
```

```

Call:
 randomForest(formula = Species ~ ., data = iris_train, mtry = 2,      importance = TRUE, ntree = 100)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 5.19%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         45          0         0  0.00000000
versicolor      0         41         3  0.06818182
virginica       0          4        42  0.08695652
```

```
importance(RF)
```

```
                setosa  versicolor virginica MeanDecreaseAccuracy
Sepal.Length  3.003956  4.01824632  1.885682             4.663758
Sepal.Width   2.015281  0.02670148  2.832304             2.617572
Petal.Length 11.846486 15.56040968 13.867746            16.436637
Petal.Width   8.171265 14.30018424 13.637917            13.928544
             MeanDecreaseGini
Sepal.Length         9.618752
Sepal.Width          2.143676
Petal.Length        42.021088
Petal.Width         35.507632
```

```
varImpPlot (RF)
```



```r
preds = predict(RF,newdata=iris_test)
table(Y_test, preds)
```

```
            preds
Y_test       setosa versicolor virginica
  setosa          5          0         0
  versicolor      0          6         0
  virginica       0          1         3
```

Neural Network

Comparing all classification methods

Method	Linearity	Normality	Constant Variance	Big Sample Size	K>2
Logistic Regression	Yes	Yes on Y	Yes	Yes	K=2
LDA	Yes	Yes on X and Y	Yes	No	Yes
QDA	No	Yes	No	No	Yes
KNN	No	No	No	Yes	Yes

Experimental Design

Computational Techniques

Standarization / Normalization

K-Fold Cross Validation

- A method to validate the model and its parameters. MSE is highly variable measurement for the model. Thus, cross validation reduce this variablity and help us obtain the most stable MSE.
- Devide the data set into K different parts. Remove the first part, fit a model using the rest of the parts, and test on this removed first part and take MSE. Repeat this procedure for all folds. At last, average all K different MSE and obtain the MSE for the model.
- When $K = n$, it's called "Leace-One-Out Cross Validation".

R: CV to choose order of polynomial

```r
library(boot)
set.seed(1)
K = 5
CV_MSE = rep(0,K)
for(i in 1:K){
  glm.fit = glm(Sepal.Width~poly(Sepal.Length,i), data=iris)
  CV_MSE[i] = cv.glm(iris, glm.fit)$delta[1]
}
CV_MSE
```

```
[1] 0.1908667 0.1890806 0.1859583 0.1863217 0.1926010
```

R: Package "crossval"

```r
library(crossval)
```

Bootstrap

R: Obtain mean and CI using package "boot"

```r
library(boot)
set.seed(1)

# Mean
mean.fn = function(data, index){
return(c(mean(data[index,1])))
}
mean_boot = boot(iris, mean.fn, R=1000)

# CI

CI_norm_boot = boot.ci(mean_boot, type = 'norm')
CI_norm_boot$normal
```

```
     conf
[1,] 0.95 5.702502 5.978229
```

R Features

Basics

Functions

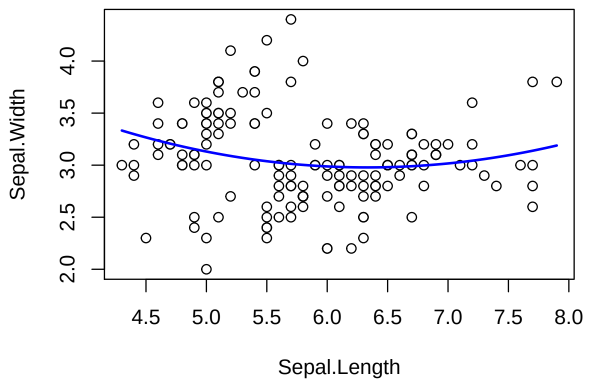
I() and poly() for polynomial regression

- I() function simply grants lm() function to add higher order variables.
- poly() function uses an orthogonal basis to fit polynomial regression. That is, the variables are linearly transformed into linearly independent set, thus no multicolinearity is present. Moreover, the statistics generated such as r^2 and resulting plots are the same as using I() functions. However, the coefficients generated are different hence it suffers from interpretations.

```
grids=seq(from=min(iris$Sepal.Length),to=max(iris$Sepal.Length), by=0.01)

m_I = lm(Sepal.Width~Sepal.Length+I(Sepal.Length^2), data=iris)
summary(m_I)
```

```
Call:
lm(formula = Sepal.Width ~ Sepal.Length + I(Sepal.Length^2),
    data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.13070 -0.26310 -0.02446  0.25728  1.38725

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        6.41584    1.58499   4.048 8.33e-05 ***
Sepal.Length      -1.08556    0.53625  -2.024   0.0447 *
I(Sepal.Length^2)  0.08571    0.04476   1.915   0.0574 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4304 on 147 degrees of freedom
Multiple R-squared:  0.03783,   Adjusted R-squared:  0.02474
F-statistic:  2.89 on 2 and 147 DF,  p-value: 0.05877
```

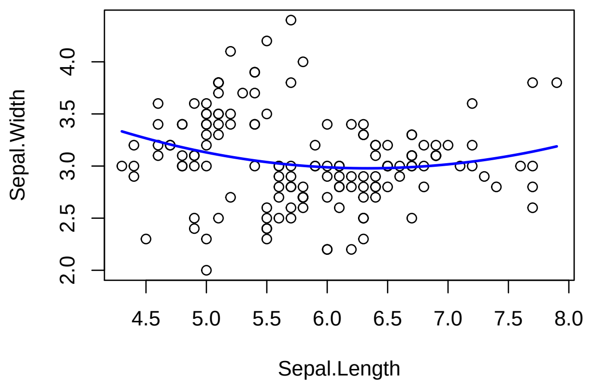
```
pred = predict(m_I,newdata=list(Sepal.Length=grids),se=TRUE)
plot(Sepal.Width~Sepal.Length, data=iris);lines(grids,pred$fit, lwd=2, col="blue")
```



```
m_poly = lm(Sepal.Width~poly(Sepal.Length,2), data=iris)
summary(m_poly)
```

```
Call:
lm(formula = Sepal.Width ~ poly(Sepal.Length, 2), data = iris)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.13070 -0.26310 -0.02446  0.25728  1.38725 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)             3.05733    0.03515  86.991   <2e-16 ***
poly(Sepal.Length, 2)1 -0.62552    0.43044  -1.453   0.1483    
poly(Sepal.Length, 2)2  0.82430    0.43044   1.915   0.0574 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4304 on 147 degrees of freedom
Multiple R-squared:  0.03783,	Adjusted R-squared:  0.02474 
F-statistic:  2.89 on 2 and 147 DF,  p-value: 0.05877
```

```
pred = predict(m_poly,newdata=list(Sepal.Length=grids),se=TRUE)
plot(Sepal.Width~Sepal.Length, data=iris);lines(grids,pred$fit, lwd=2, col="blue")
```



Normalization and Standarization

`apply()`, `sapply()`, `tapply()`, `lapply()`

Package: `dplyr`