

# Introduction to Data Science using R

# Topics

- Basics of building predictive models
  - Linear Regression
  - Logistic Regression
  - Bayes Models
  - Assessing accuracy of models, Plotting results of models for interpretation and tuning purposes.
  - scatterplot, histogram, density plots

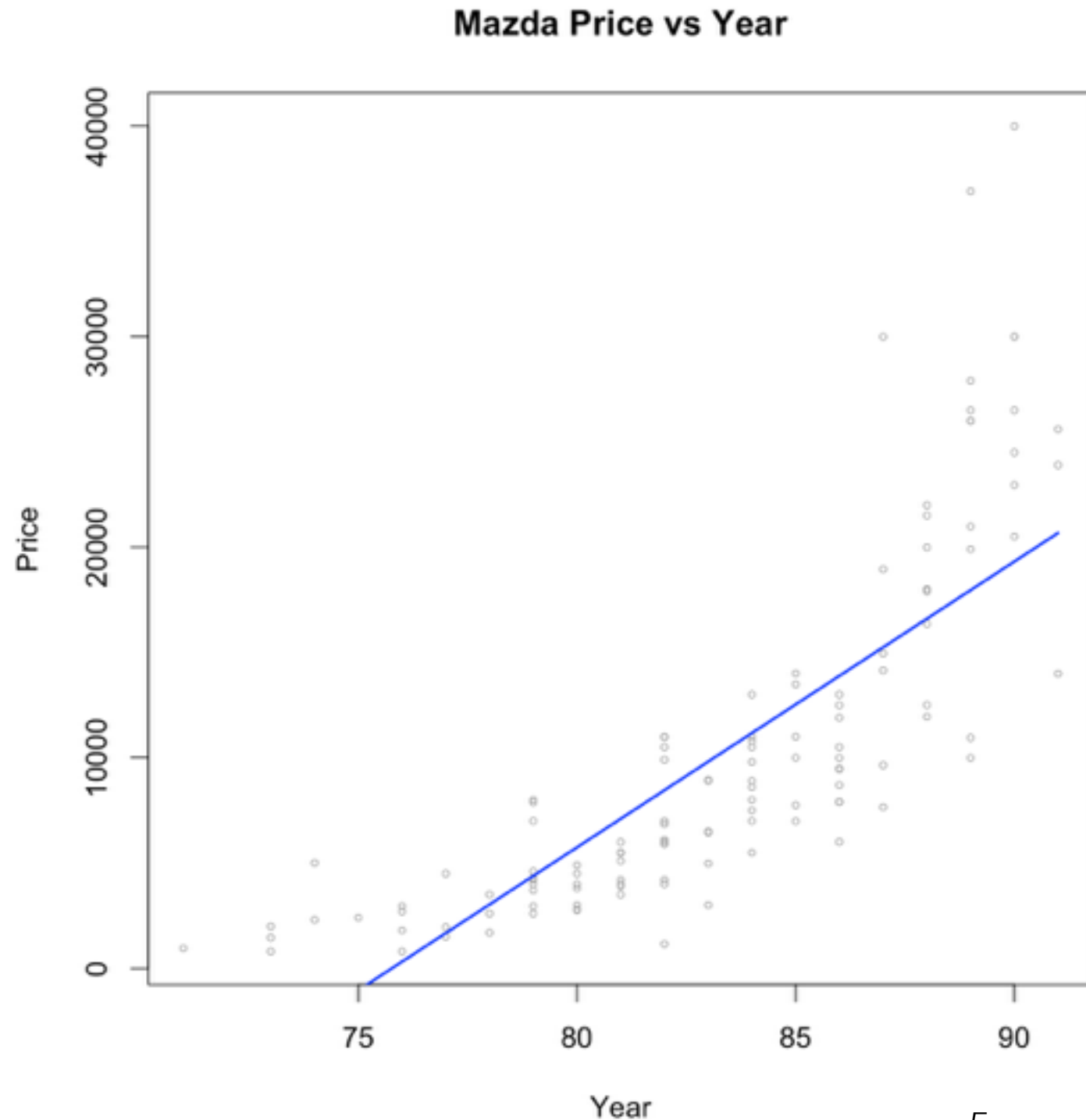
# Linear Regression

- Linear regression is a simple approach to supervised learning.
- Relationship between  $X$  and  $Y$  is assumed to be linear
- Linear regression is simple and very useful
- $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$

# Sample Linear Models

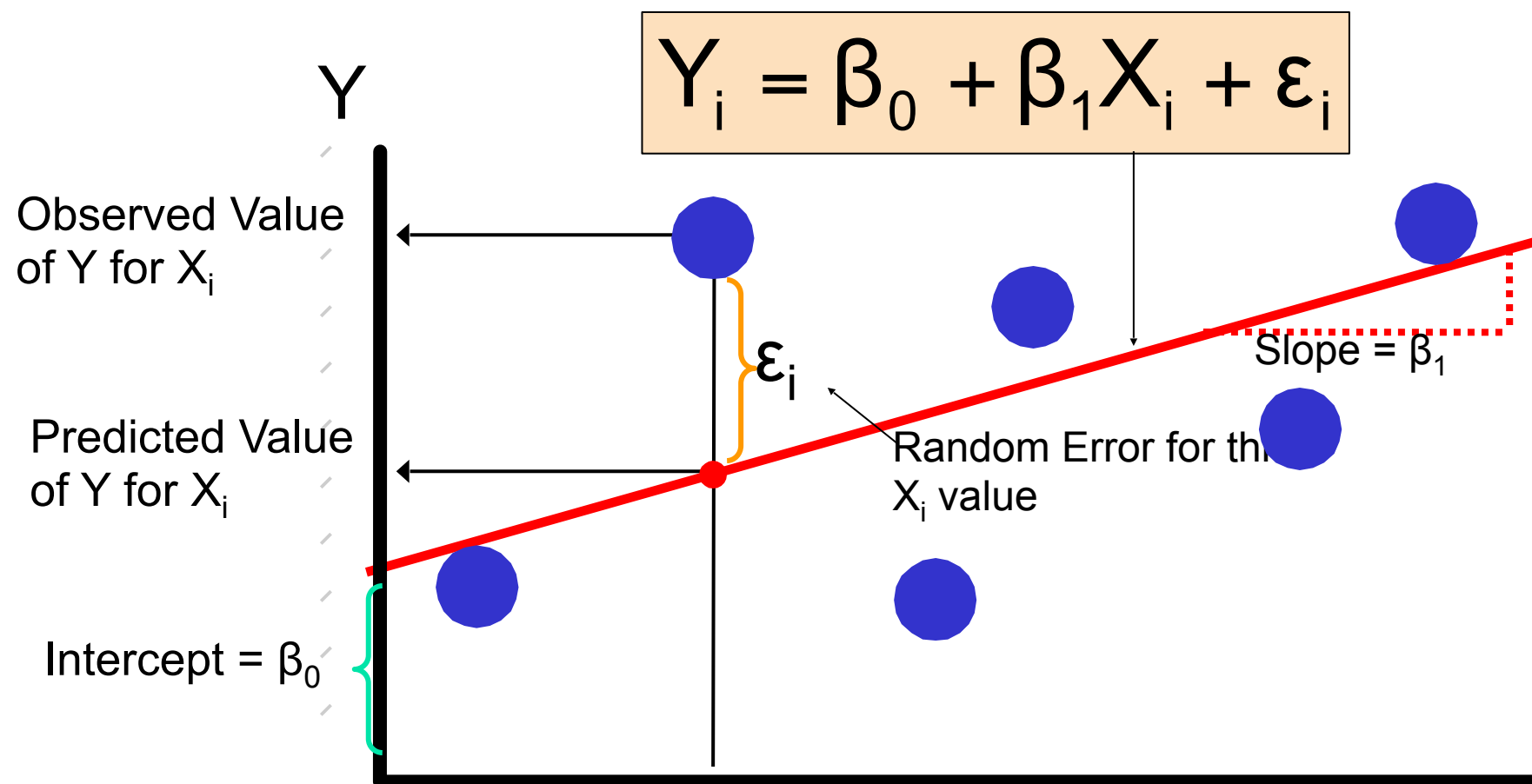
- $\text{Sales} = f(\text{TV}, \text{Radio}, \text{Newspaper}) + \varepsilon$
- $\text{Sales} \approx \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{Radio} + \beta_3 \times \text{Newspaper}$
- $\text{Sales} \approx \beta_0 + (\beta_1 \times \text{TV}) + (\beta_2 \times \text{Radio}) + (\beta_3 \times \text{Radio}^2) + (\beta_4 \times \text{Newspaper})$
- $\text{Sales} \approx \beta_0 + (\beta_1 \times \text{TV}) + (\beta_2 \times \text{Radio}) + (\beta_3 \times \text{Radio}^2) + (\beta_4 \times \text{Radio} \times \text{Newspaper}) + (\beta_5 \times \text{Newspaper})$

# Linear Regression



- $Y = \beta_0 + \beta_1 X + \varepsilon$ 
  - $\beta_0$  and  $\beta_1$  are parameters
  - $\varepsilon$  is the random error in Y for  $X_i$  value

# Simple Linear Regression Model



# Estimation of parameters

- Determine  $\beta_0$  and  $\beta_1$  by minimizing RSS (Residual sum of squares)
- $RSS = \sum (y_i - (\beta_0 + \beta_1 x_i))^2$
- Determine  $\beta_0$  and  $\beta_1$  by minimizing RSS (Residual sum of squares)

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

# Is the parameter significant in predicting?

	Estimate	Std Err	t - statistic	p - value
(Intercept)	-104349.2	8600.4	-12.13	<2e-16
Year	1375.5	103.2	13.33	<2e-16

Null Hypothesis:  $\beta_1 = 0$

t statistic =  $(\text{est } \beta_1 - 0) / \text{SE}(\text{est } \beta_1)$

p - value =  $\Pr(\text{t statistic} < 13.33)$



# Linear Regression

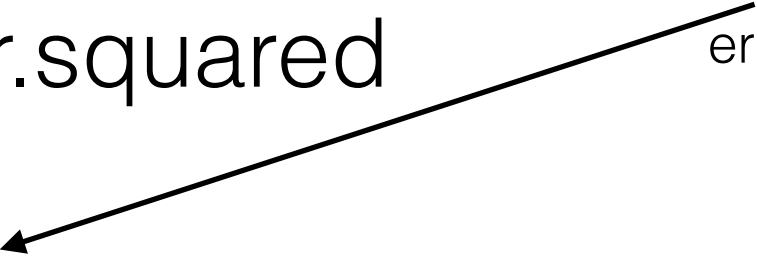
- Fraction of residuals explained or  $R^2 = (TSS - RSS) / TSS$
- $TSS = \sum (y_i - \text{mean}(y))^2$

- R code

```
> summary(lm)$r.squared
```

64.35%  
errors explained

[1] 0.6435645



# Lab

- Use Mazda.txt data to fit a linear model (Predict Price based on Year)
- Use Mazda.txt data from <http://www.statsci.org/data/first.html>

# mazda sales model

```
# read data; you can get the data from http://www.statsci.org/data/first.html

mazdas <- read.csv("~/Desktop/ML/mazdas.txt", sep="\t")

attach(mazdas)

# Create two datasets.
# Training dataset: used to build the model (80% of of the data)
# Test dataset: used to test the model

train=sample(nrow(mazdas), nrow(mazdas)*.8)

#
# check the results of train and -train
#

train
-train
```

# mazda sales model

```
# Build linear model
# Response: Price
# Predictor(s): Year

lm.fit = lm(mazdas$Price~.,data=mazdas, subset=train)

#
# Check model summary
#

summary(lm.fit)
```

# mazda sales model

```
#  
# Draw scatter plot of data (Price vs Year)  
#  
plot(mazdas$Year,mazdas$Price,xlab="Year", ylab="Price", cex=.5,col="darkgrey")  
  
# Plot the fit  
abline(lm.fit,lwd=2,col="blue")
```

# mazda sales model

```
# Check accuracy of the model using test data

ytest = mazdas$Price[-train]

ytestp = predict(lm.fit,newdata=data.frame(Year = mazdas$Year[-train]))

# Mean squared error

mse1 <- mean((ytest - ytestp)^2)
mse1
```

# mazda sales model

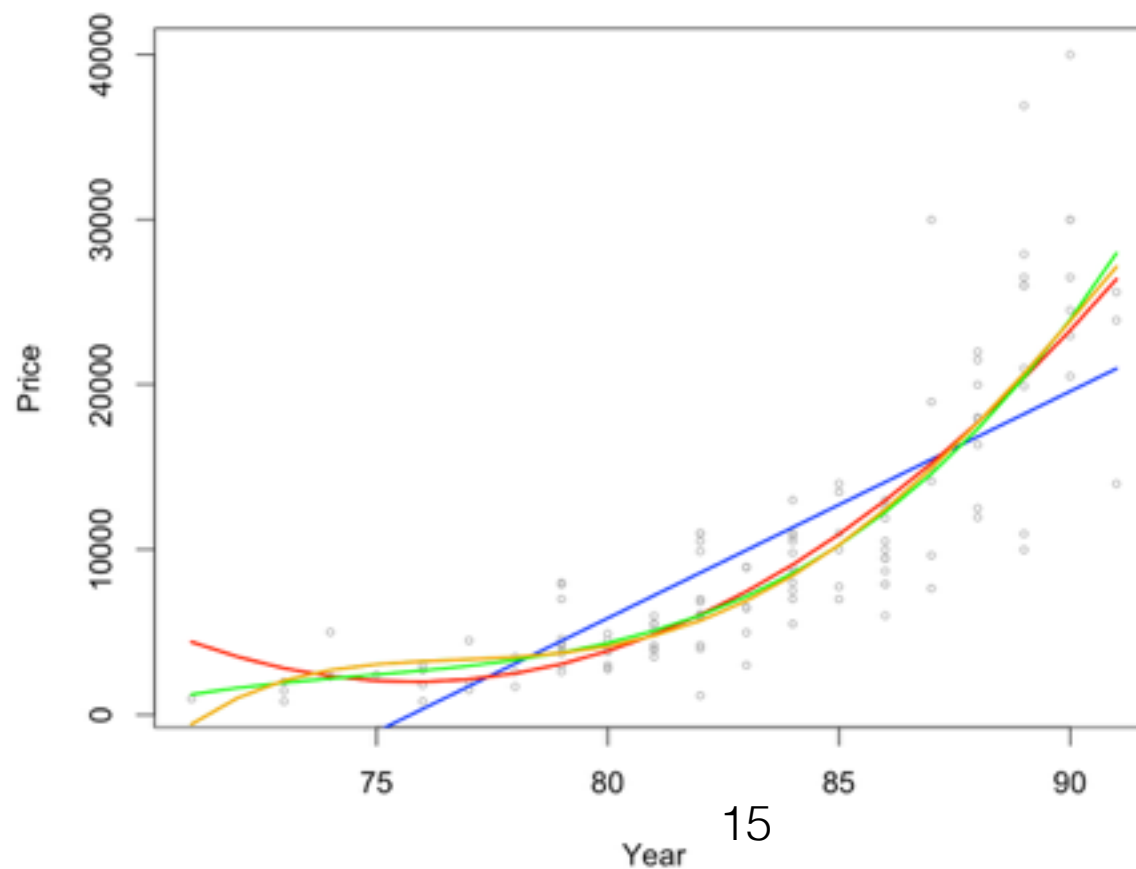
# Try the models by adding additional non-linear functions

```
lm.fit2 = lm(mazdas$Price~poly(Year,2),data=mazdas, subset=train)
```

```
lm.fit3 = lm(mazdas$Price~poly(Year,3),data=mazdas, subset=train)
```

```
lm.fit4 = lm(mazdas$Price~poly(Year,4),data=mazdas, subset=train)
```

# Compare MSEs



# mazda sales model

```
#Compare Training MSE for polynomial models
```

```
par(mfrow=c(1,2))
```

```
Year.range = seq(min(mazdas$Year), max(mazdas$Year))
```

```
plot(mazdas$Year, mazdas$Price, xlab="Year", ylab="Price", cex=.5, col="darkgrey")
```

```
title("Mazda Sales vs Year - Linear Models")
```

```
colors.list = c('blue','green','brown','red','black','cyan','yellow','pink','orange','gold')
```

```
degree=1:10
```

```
mse = rep(0.0, length(degree))
```



# mazda sales model

```
#Compare Training MSE for polynomial models
```

```
for(d in degree){
```

```
# FIT A MODEL
```

```
  lm.fit = lm(mazdas$Price~poly(Year,d),data=mazdas, subset=train)
```

```
# PREDIT USING Year Range DATA
```

```
  preds=predict(lm.fit, newdata=data.frame(Year = Year.range))
```

```
  lines(Year.range, preds, lwd=2, col=colors.list[d])
```

```
# Calculate mean squared error
```

```
  mse[d] = mean(lm.fit$residuals^2)
```

```
}
```

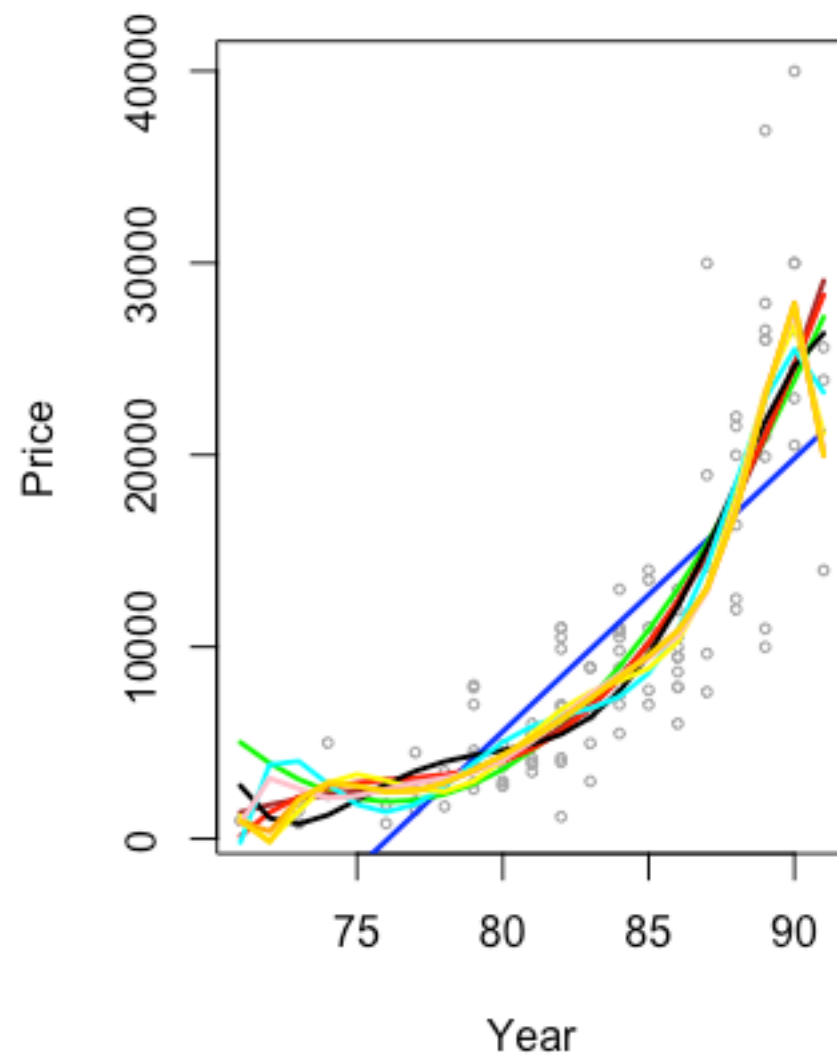
```
mse
```

```
plot(mse, xlab="Degree of Polynomial", ylab="Training MSE", type="l")
```

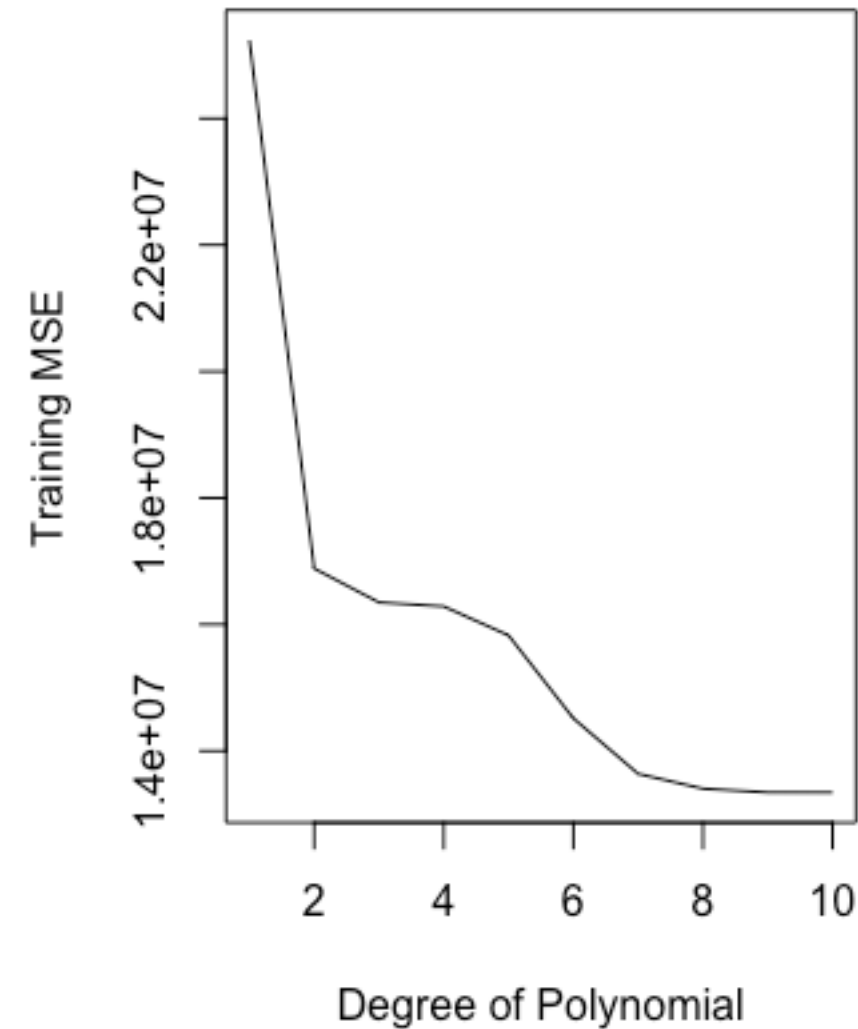
```
title("Training MSE vs Degree of Polynomial")
```

# mazda sales model

Mazda Sales vs Year - Linear Models



Training MSE vs Degree of Polynomial



# mazda sales model

```
#Compare Test MSE for polynomial models
```

```
par(mfrow=c(1,2))  
plot(mazdas$Year, mazdas$Price, xlab="Year", ylab="Price", cex=.5, col="darkgrey")  
title("Mazda Sales vs Year - Linear Models")  
colors.list = c('blue','green','brown','red','black','cyan','yellow','pink','orange','gold')
```

```
degree=1:10  
mse = rep(0.0, length(degree))
```

```
for(d in degree){
```

```
# fit linear model and draw the fit  
  lm.fit = lm(mazdas$Price~poly(Year,d),data=mazdas, subset=train)
```

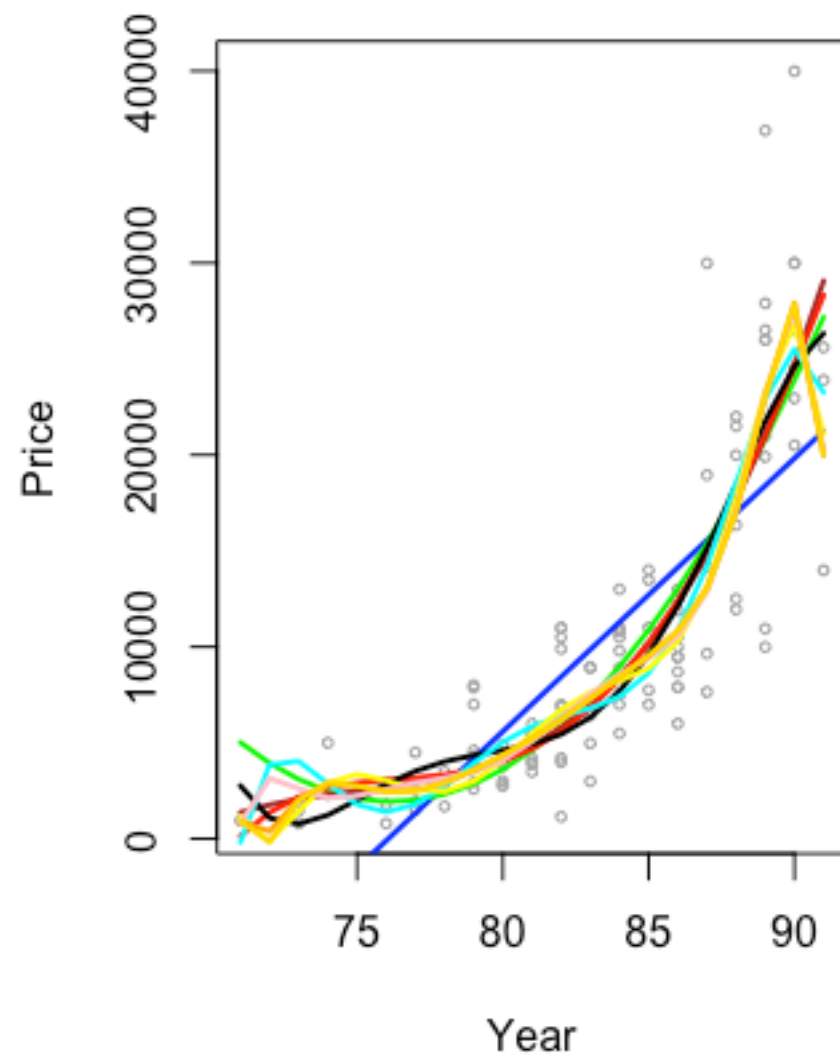
```
# PREDIT USING Year Range DATA  
  preds=predict(lm.fit, newdata=data.frame(Year = Year.range))  
  lines(Year.range, preds, lwd=2, col=colors.list[d])
```

```
# Calculate mean squared error  
  preds=predict(lm.fit, newdata=data.frame(Year = mazdas$Year[-train]))  
  mse[d] <- mean((ytest - preds)^2)  
}
```

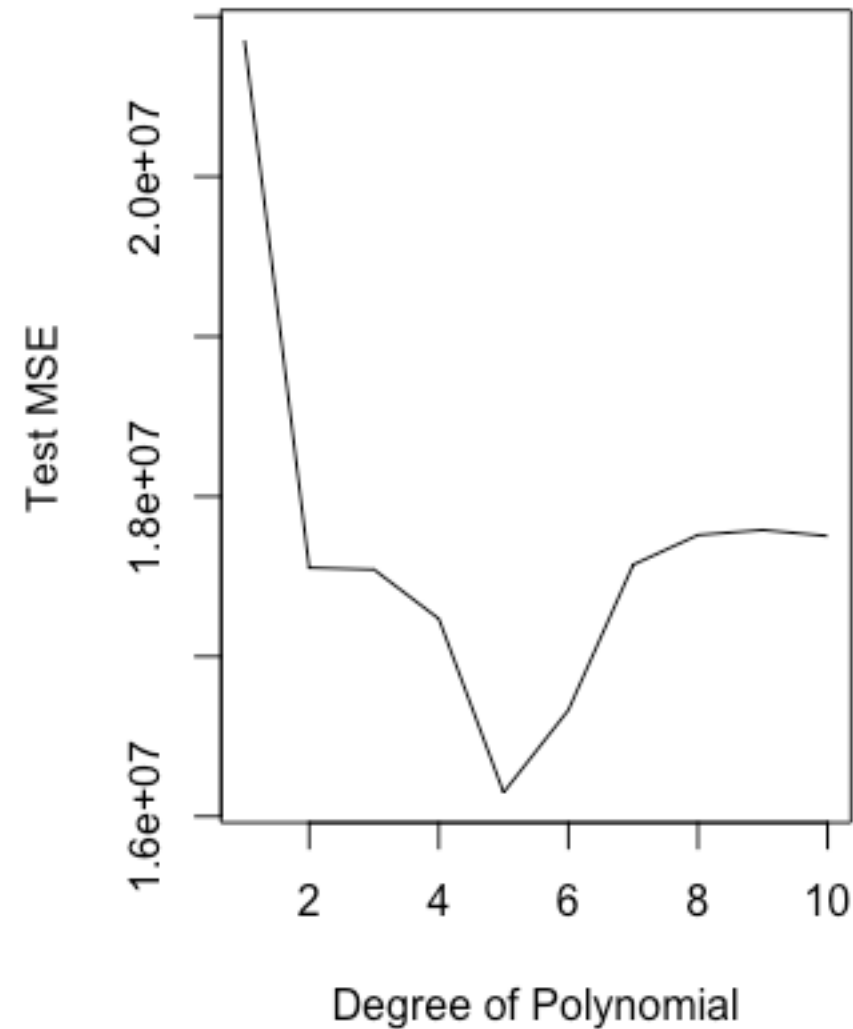
```
# Plot Test MSE  
plot(mse, xlab="Degree of Polynomial", ylab="Test MSE", type="l")  
title("Test MSE vs Degree of Polynomial")
```

# mazda sales model

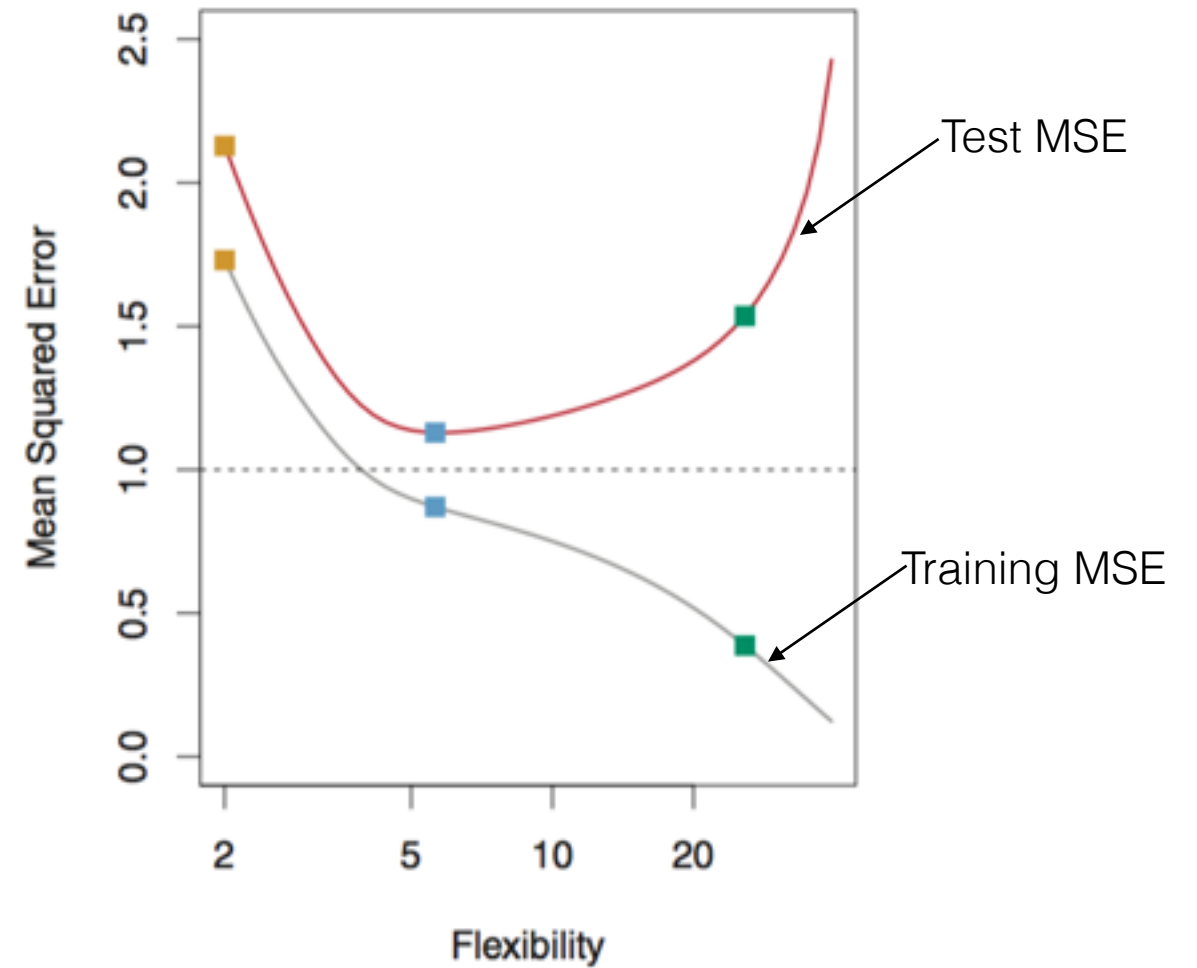
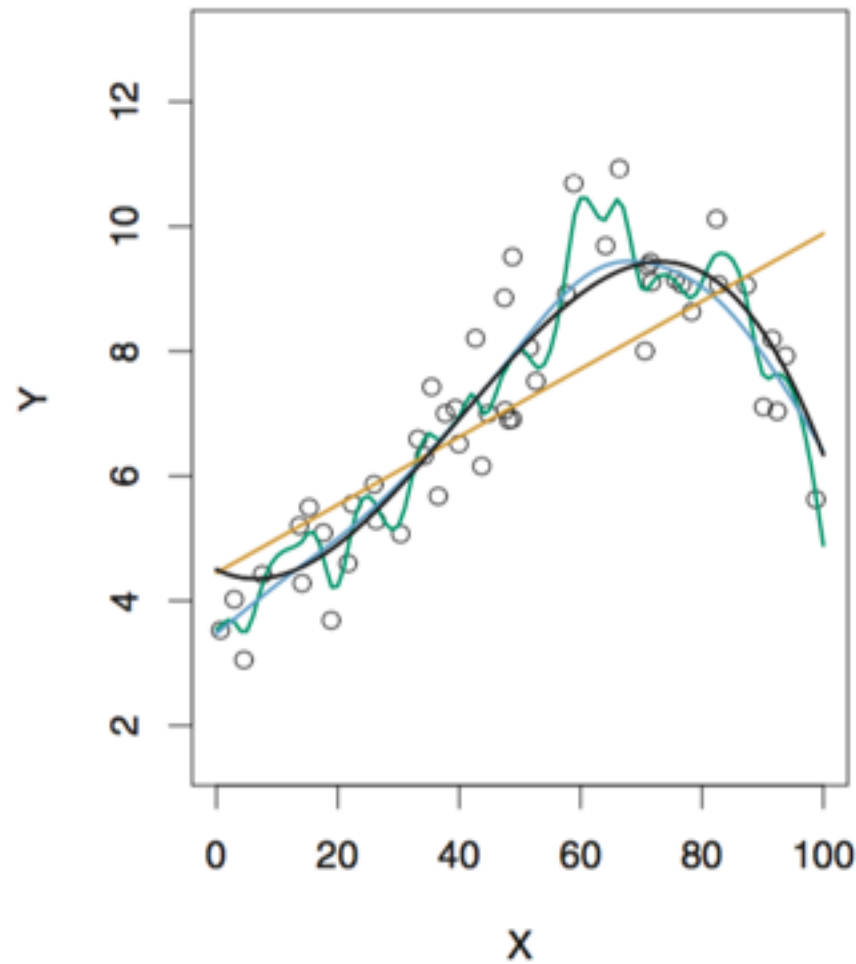
**Mazda Sales vs Year - Linear Models**



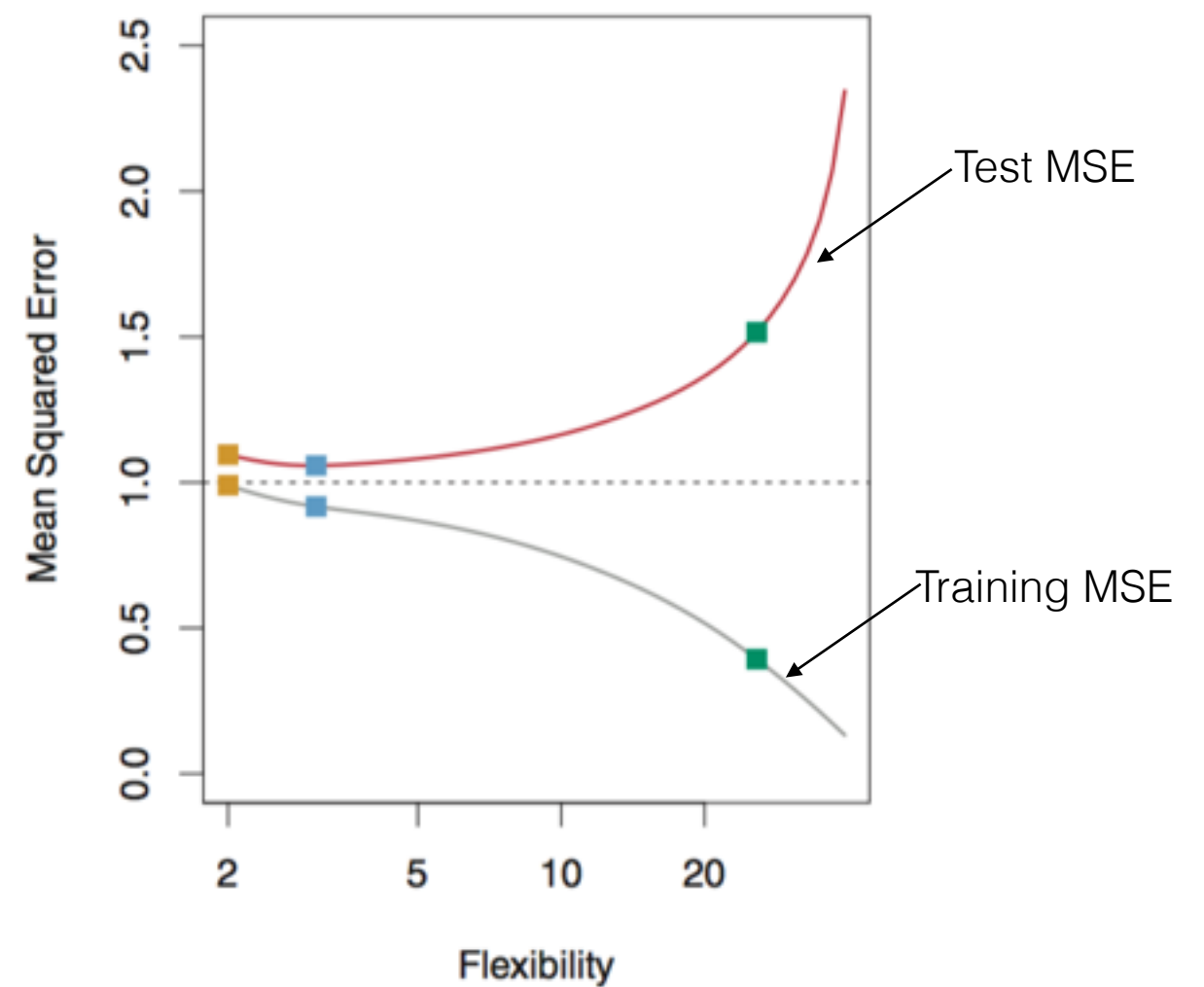
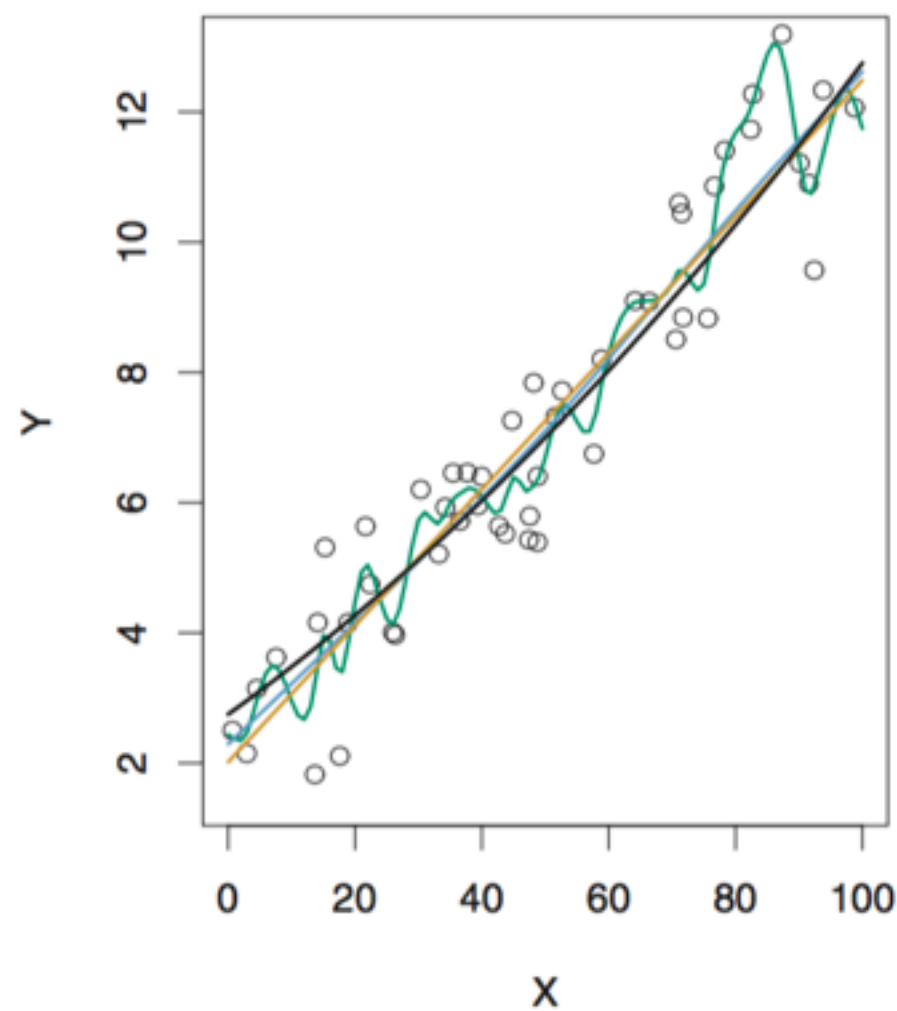
**Test MSE vs Degree of Polynomial**



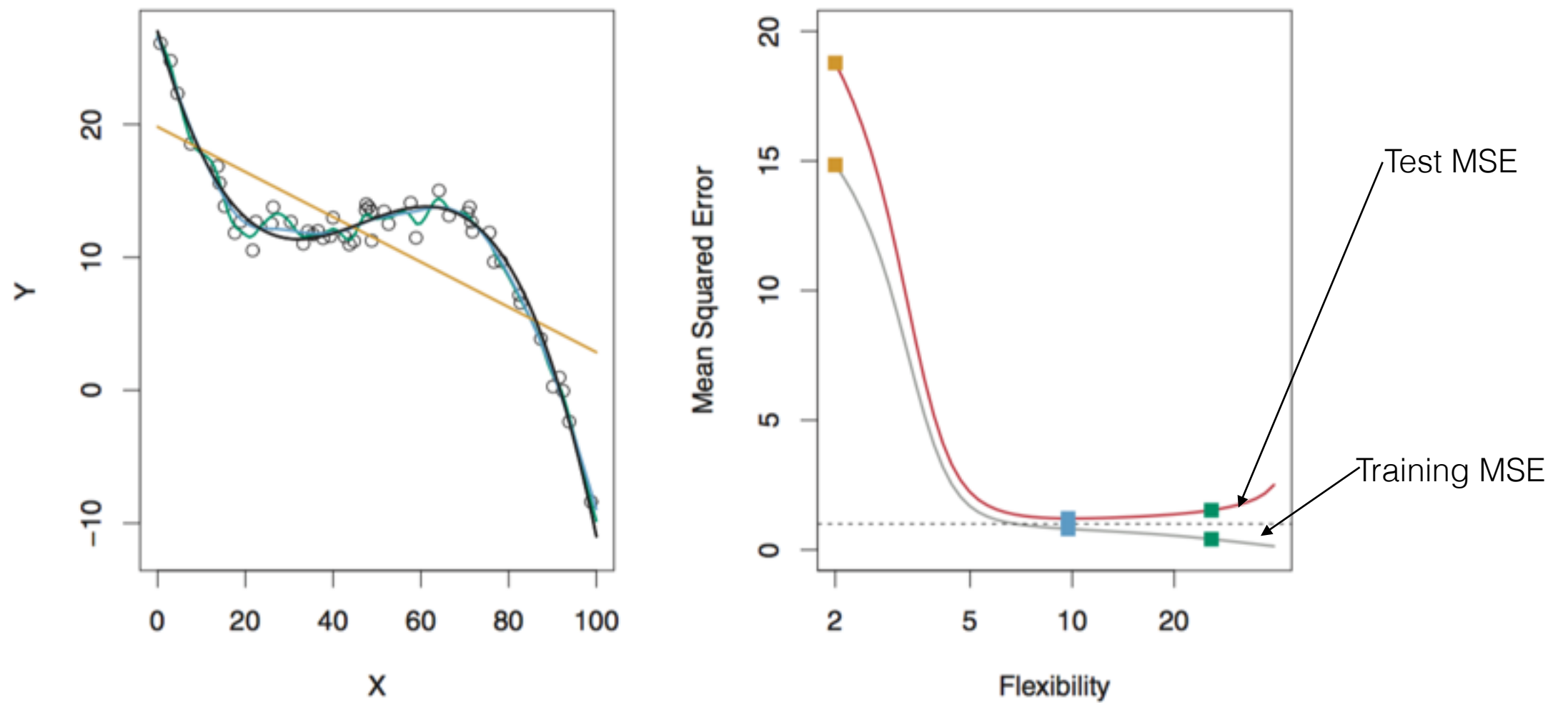
# Assessing Model Accuracy



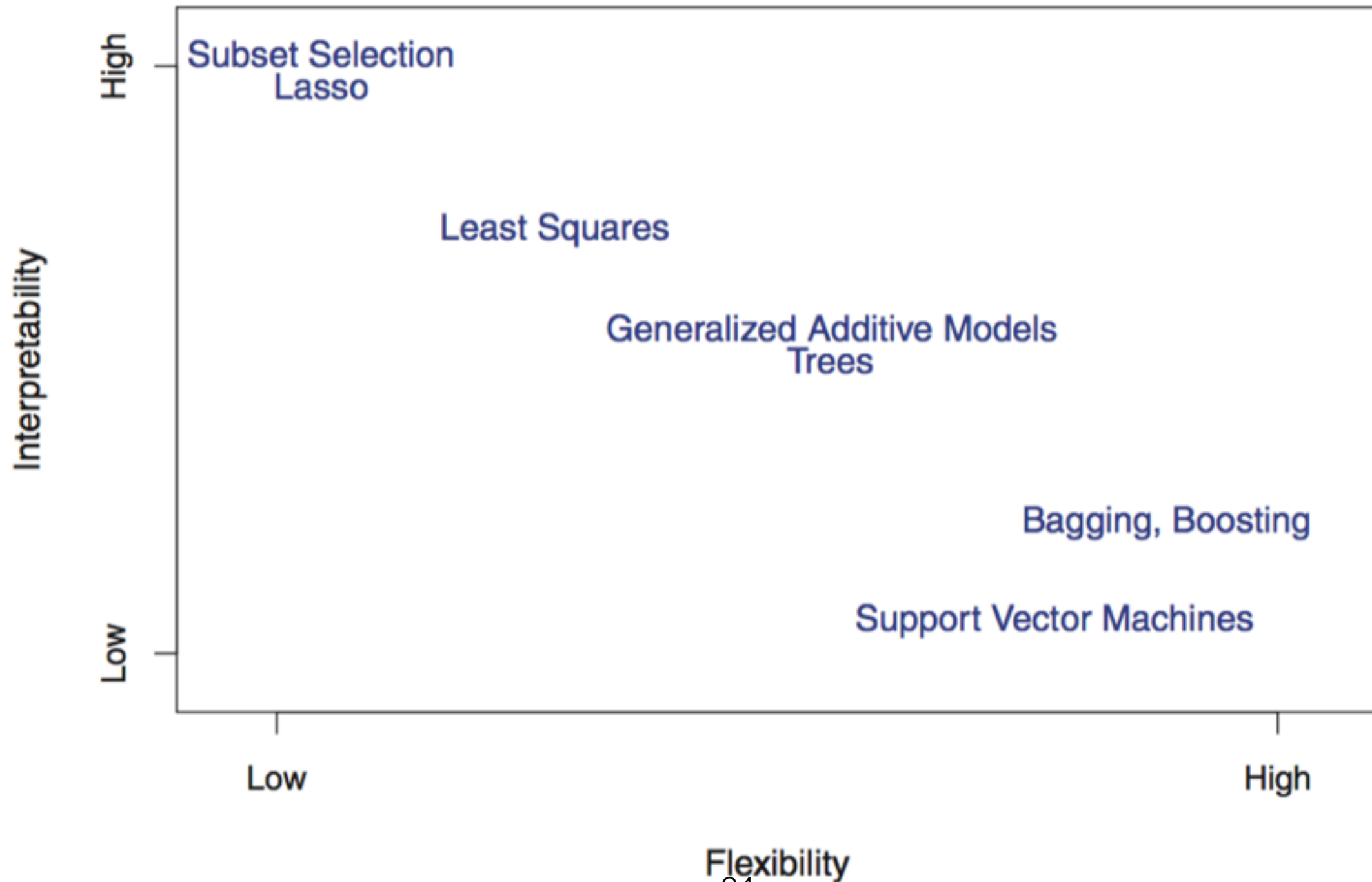
# Assessing Model Accuracy



# Assessing Model Accuracy



# Interpretability vs Flexibility





# Multiple Linear Regression

- $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$
- What to consider
  - Is any of predictors useful?
  - Which predictors are useful?
  - Using qualitative predictors
  - Interactions (  $X_1 * X_2$  )

# Linear Regression Lab - Carseats dataset

- Use Carseats data to fit a linear model (Predict Sales)
- Use ?Carseats to understand the data
- Use `summary(lm.model1)` to inspect the importance of the variables
- How does ShelfLoc represented in the model?

# Linear Regression Lab - Carseats dataset

```
#predict sales of car seats

# Get training dataset
attach(Carseats)
fix(Carseats)
names(Carseats)

# Build Linear Model 1
lm.fit1 = lm(Sales~.+Income:Advertising+Price:Age,data=Carseats)

# View the Model
summary(lm.fit1)
contrasts(ShelveLoc)

# Build Linear Model 2
train <- sample(nrow(Carseats), nrow(Carseats)*.8)
lm.fit1 <- lm(Sales~.,data=Carseats, subset=train)
ytrainp <- predict(lm.fit1,newdata=Carseats[train,-1])
ytrain <- Carseats$Sales[train]

mean((ytrain - ytrainp)^2)

# Build Linear Model3
lm.fit2 <- lm(Sales~.+Income:Advertising+Price:Age,data=Carseats, subset=train)
ytrainp <- predict(lm.fit2,newdata=Carseats[train,-1])
ytrain <- Carseats$Sales[train]

mean((ytrain - ytrainp)^2)
```

# Linear Regression Lab - Carseats dataset

```
# check accuracy using test data - Linear Model2
```

```
ytestp <- predict(lm.fit2, newdata=Carseats[-train,-1])  
ytest <- Carseats$Sales[-train]  
mean((ytest - ytestp)^2)
```

```
# check accuracy using test data - Linear Model3
```

```
ytestp <- predict(lm.fit3,newdata=Carseats[-train,-1])  
ytest <- Carseats$Sales[-train]  
mean((ytest - ytestp)^2)
```

# Regression vs Classification

- Regression
  - Typically quantitative responses
  - Examples: Person's weight, Stock Price, Car Price
- Classification
  - Typically qualitative responses
  - Examples: Person's gender, Stock Price direction, Disease diagnosis

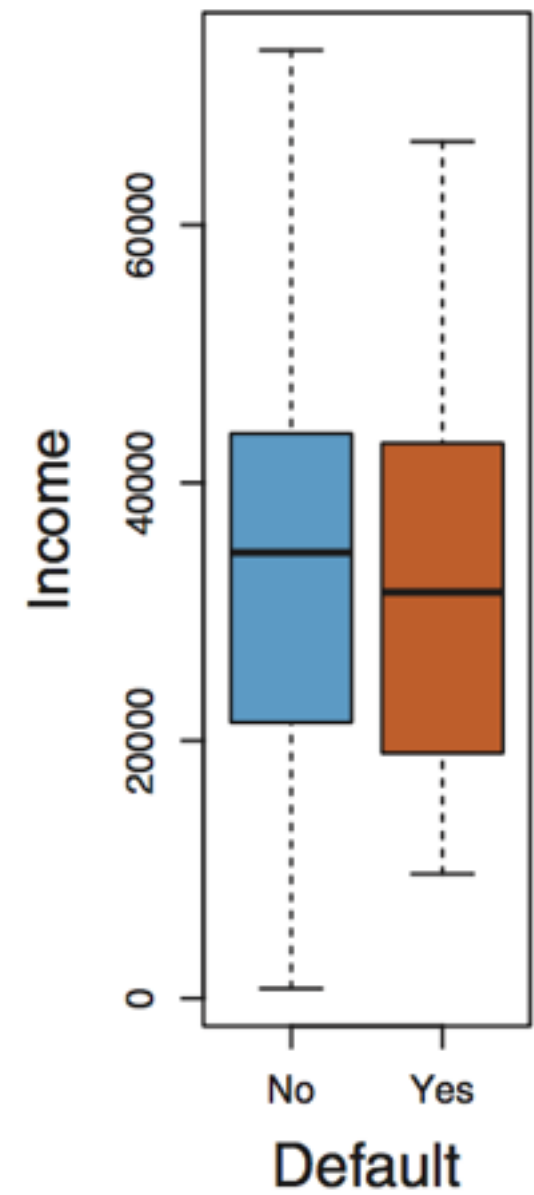
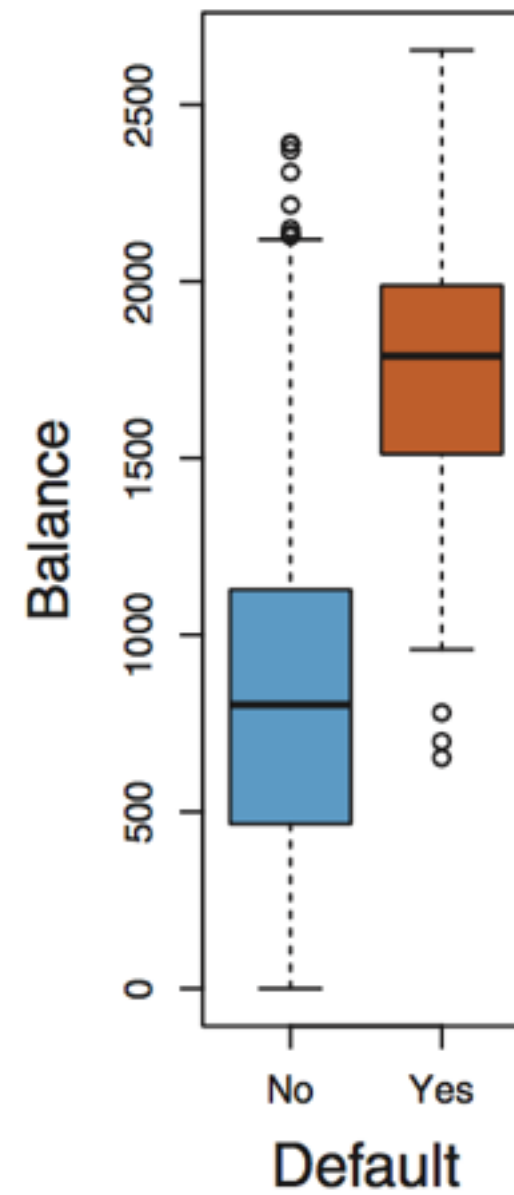
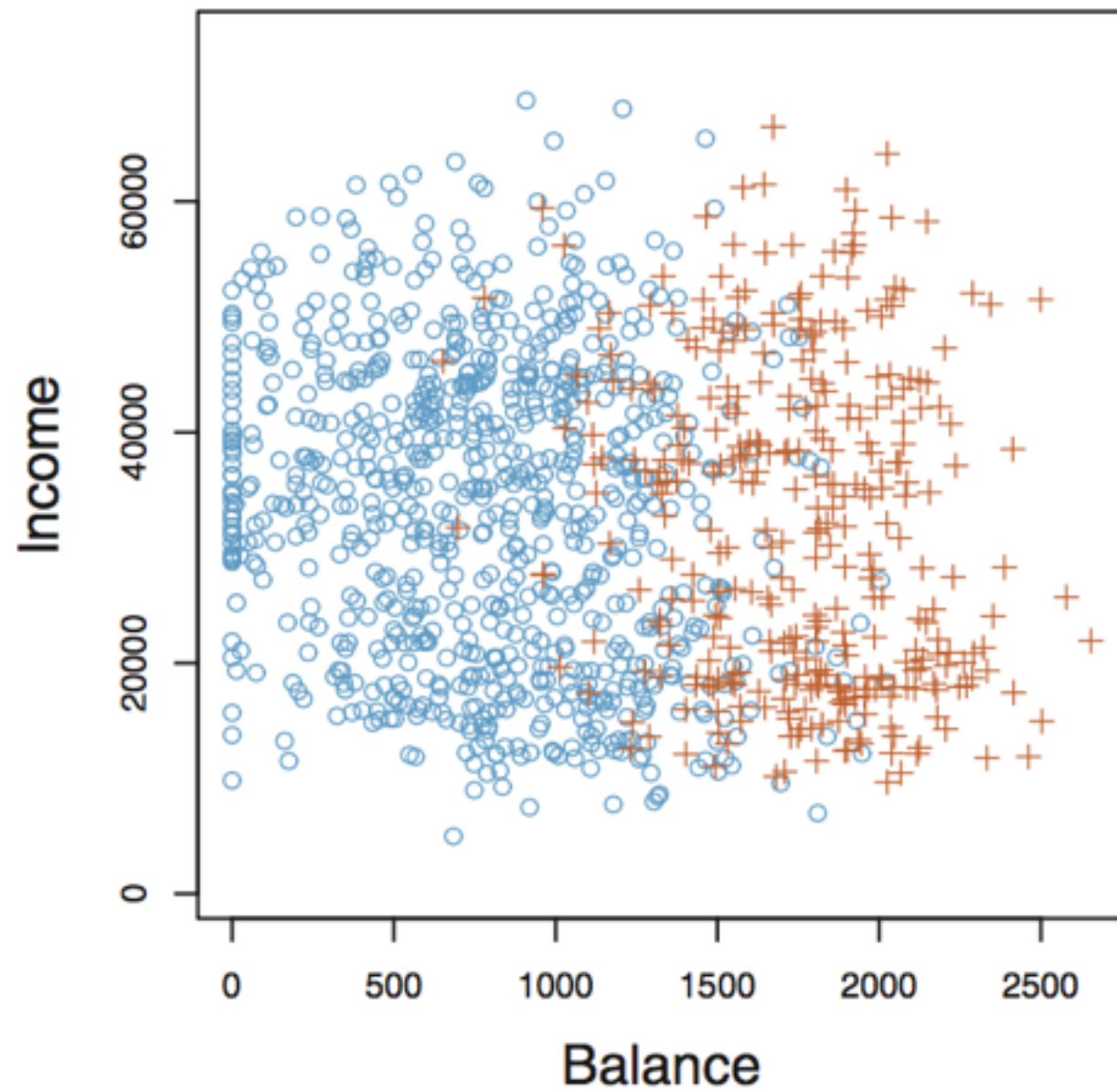
# Classification - Examples

- text categorization (e.g., spam filtering)
- fraud detection
- optical character recognition
- machine vision (e.g., face detection)
- market segmentation (e.g., predict if customer will respond to promotion)
- bioinformatics (e.g., classify proteins according to their function)

# ML methods

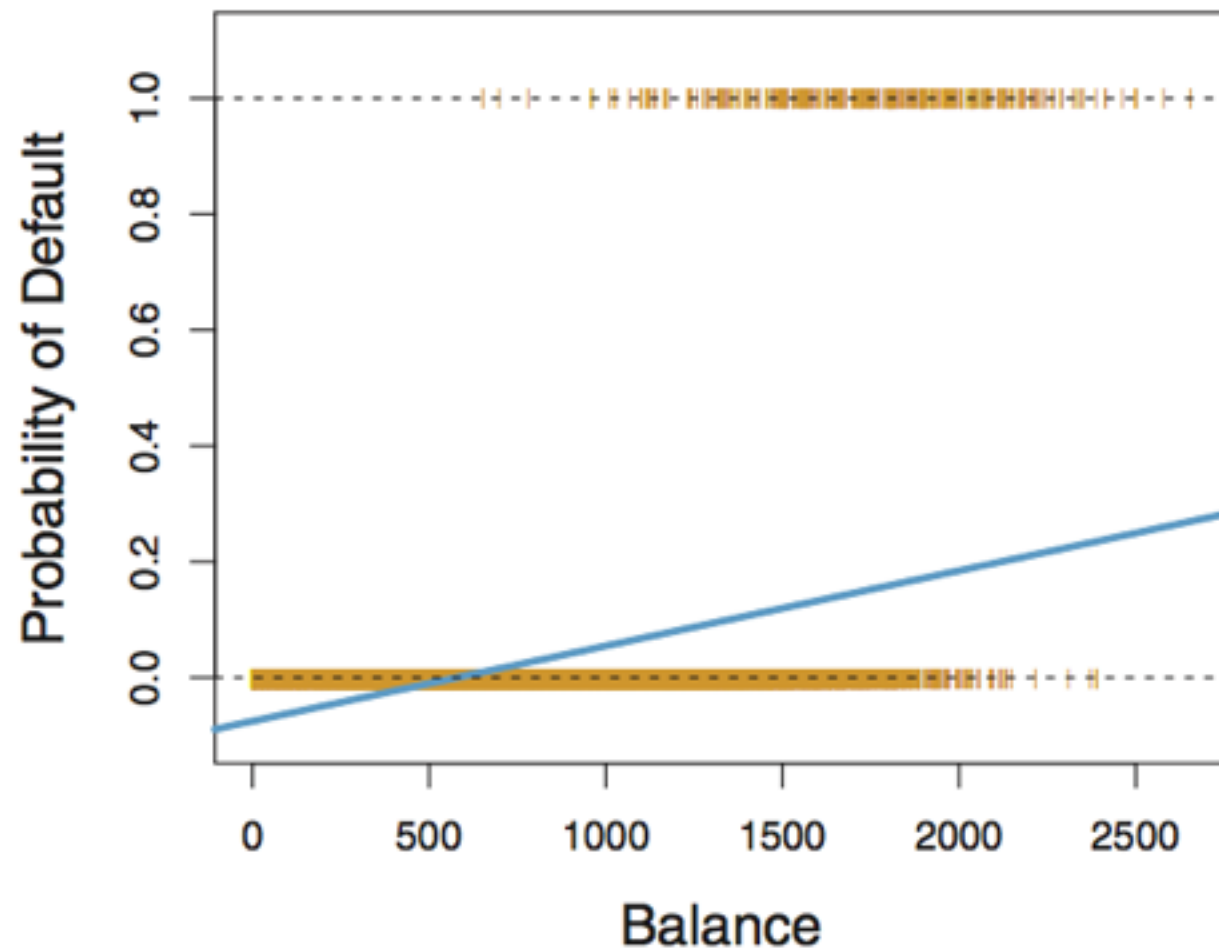
- Logistics Regression
- Bayes Theorem
- Decision Tree
- Bagging, Random Forests, Boosting
- Support Vector Machine
- Neural Networks

# Logistic Regression



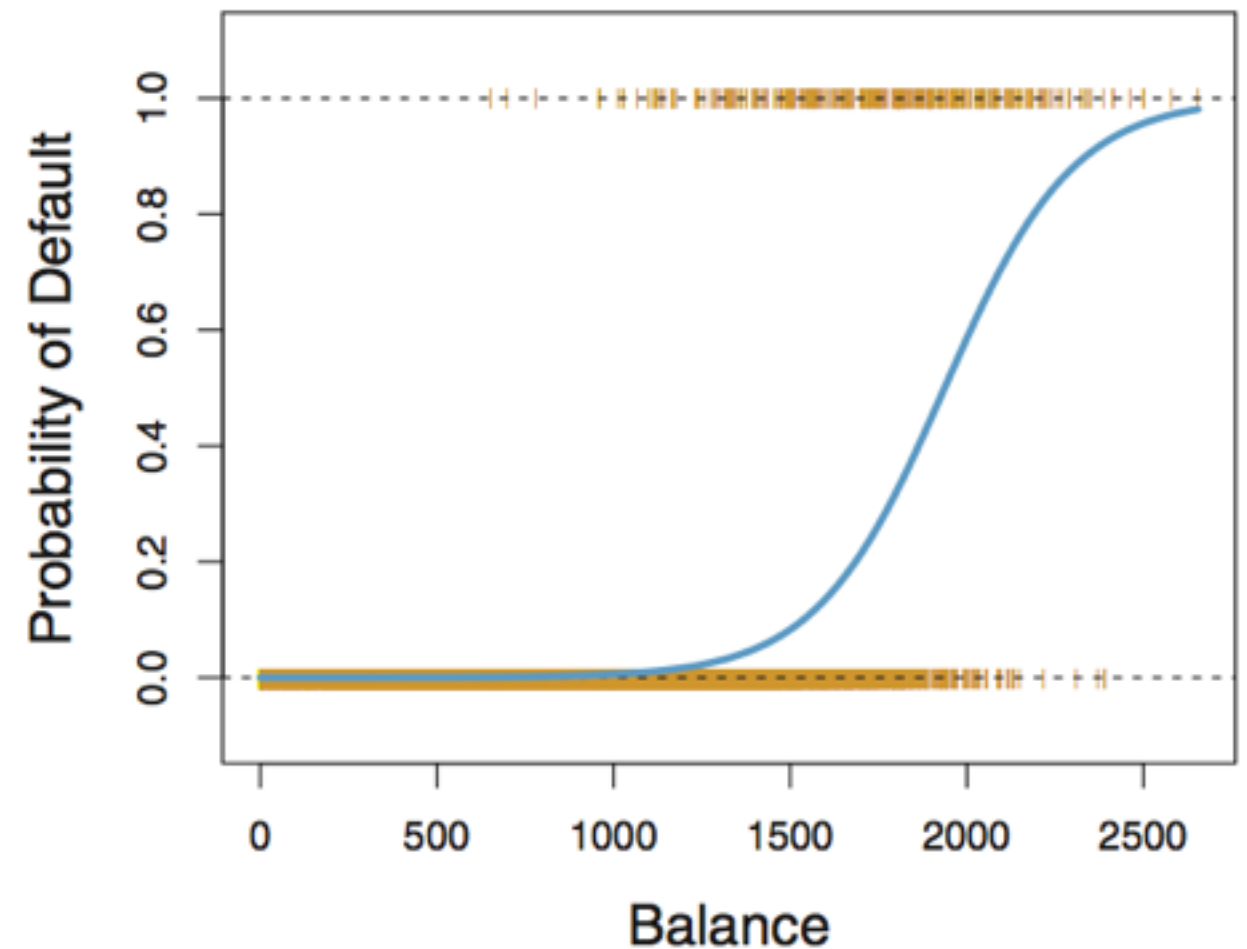


# Logistic Regression



$$p(X) = \beta_0 + \beta_1 X$$

Linear Regression Model



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

Logistic Regression Model

# Logistic Regression

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

$\beta_0 + \beta_1$  are estimated using Maximum Likelihood function

$$\ell(\beta_0, \beta_1) = \prod_{i: y_i=1} p(x_i) \prod_{i': y_{i'}=0} (1 - p(x_{i'})).$$

	Coefficient	Std. error	Z-statistic	P-value
<b>Intercept</b>	-10.6513	0.3612	-29.5	<0.0001
<b>balance</b>	0.0055	0.0002	24.9	<0.0001

# Logistic Regression

log-odds

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X.$$

Multiple Logistic Regression

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}.$$

# Logistic Regression

1. Estimates  $P(Y | X)$
2. Widely used in binary classification
3. Use r function glm to build a model
4. Can use thresholds to increase True Positives for desired class

# Lab - Predict Weekly S&P Stock Direction

- Use the Weekly dataset to predict the S&P Stock Direction
- training data: period from 1990 to 2008
- test data: period 2009 onwards

# Lab - Predict Weekly S&P Stock Direction

```
library(ISLR)
data(Weekly)

#training data
train=Weekly$Year < 2009

#fit logistic regression model
glm.fit=glm(Direction~Lag2,data=Weekly,family=binomial,subset=train)

#predict for test data
glm.probs = predict(glm.fit, Weekly[!train,], type = "response")

#convert predictions to directions "Down" or "Up"
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"

#Confusion Matrix
table(glm.pred, Weekly$Direction[!train])

#Overall fraction of correct predictions
mean(glm.pred == Weekly$Direction[!train])
```

# Lab - Predict Weekly S&P Stock Direction

```
> #Confusion Matrix
```

```
> table(glm.pred, Weekly$Direction[!train])
```

```
glm.pred Down Up
```

```
Down   9  5
```

```
Up    34 56
```

```
>
```

```
> #Overall fraction of correct predictions
```

```
> mean(glm.pred == Weekly$Direction[!train])
```

```
[1] 0.625
```

# Exercise

- a. Write the equation to predict S&P Stock Direction
- b. Using the equation, calculate the probability of Direction going up for the following observation

Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
2009	6.760	-1.698	0.926	0.418	-2.251	3.793110	-4.448

- c. Using the predict function, find out the probability of Direction going up for the observation

(Hint: `predict(glm.fit,`  
`newdata=data.frame(Year=2009,Lag1=6.760,Lag2=-1.698,Lag3=0.926,`  
`Lag4=0.418,Lag5=-2.251,Volume=3.793110,`  
`Today=-4.448), type = "response"))`



# Lab - Predict Default

- Use Default data to fit a logistic regression model (Predict Probability to Default based on Balance)

# Lab - Predict Default

```
library(ISLR)

# Analyze Default Data
par(mfrow=c(1,2))

boxplot(Default$balance~Default$student, xlab="Student", ylab="Balance", col=c("blue","brown"))
boxplot(Default$income~Default$student, xlab="Student", ylab="Income", col=c("blue","brown"))

par(mfrow=c(1,1))

# Plot training data: Default vs. Balance
default.prob <- ifelse(Default$default=="Yes", 1, 0)
plot(Default$balance, default.prob, col="goldenrod3", pch=" ", xlab="Balance", ylab="Probability to Default")

# Fit logit model: Default vs. Balance
glm.fit <- glm(default~balance,data=Default,family=binomial)

# Predict default for balances 1 to 2600
glm.probs=predict(glm.fit,newdata=data.frame(balance=seq(1:2600)),type="response")

# Draw default prediction
lines(seq(1:2600),glm.probs, cex=.2, col="blue")

# Calculate confusion matrix
glm.pred=rep("No", nrow(Default))
glm.probs=predict(glm.fit,type="response")
glm.pred[glm.probs > .5]="Yes"
table(glm.pred,Default$default)
```

# Lab - Predict Default

```
# Plot training data: Default vs. Balance and Student
default.prob <- ifelse(Default$default=="Yes", 1, 0)
plot(Default$balance, default.prob, col="goldenrod3", pch=" ", xlab="Balance", ylab="Probability to
Default")

# Fit logit model: Default vs. Balance
glm.fit2 <- glm(default~balance+student,data=Default,family=binomial)

# Predict default for balances 1 to 2600 and student=Yes
glm.probs=predict(glm.fit2,newdata=data.frame(balance=seq(1:2600),student='Yes'),type="response")

# Draw default prediction for student
lines(seq(1:2600),glm.probs, cex=.2, col="blue")

# Predict default for balances 1 to 2600 and student=No
glm.probs=predict(glm.fit2,newdata=data.frame(balance=seq(1:2600),student='No'),type="response")

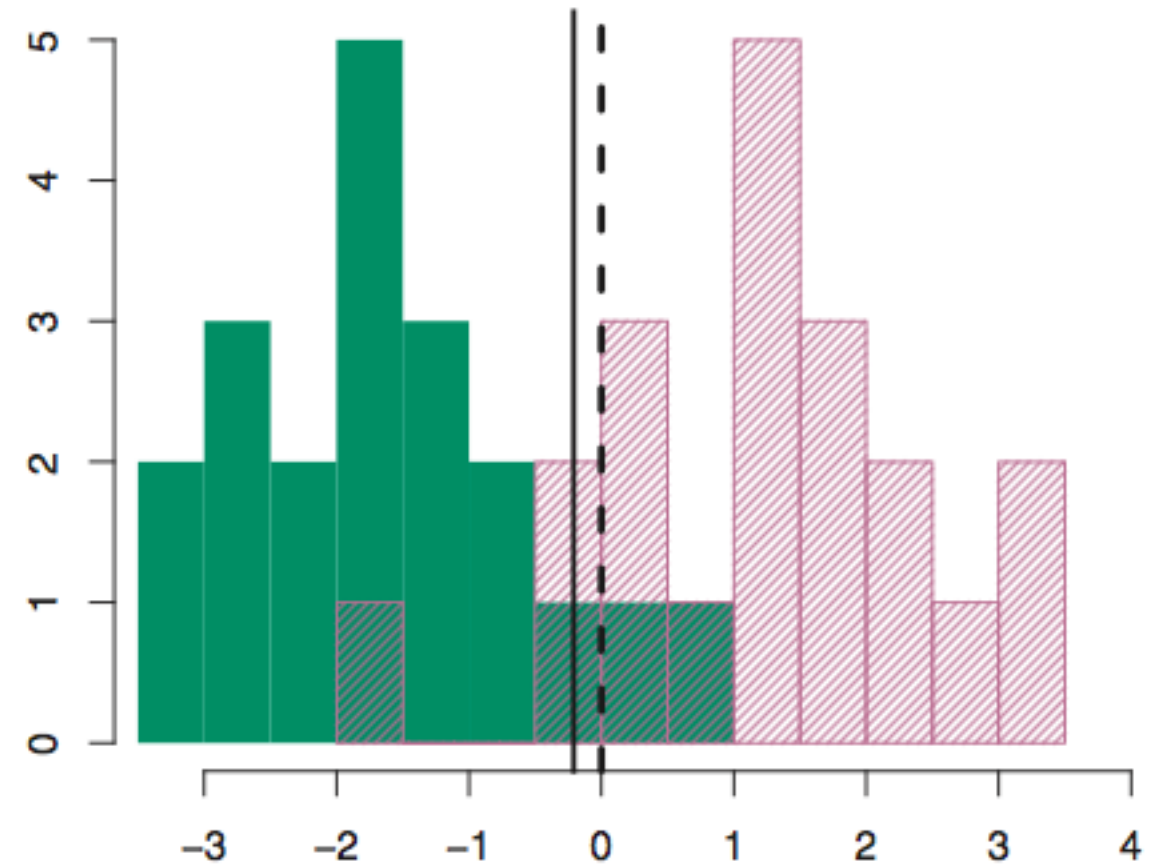
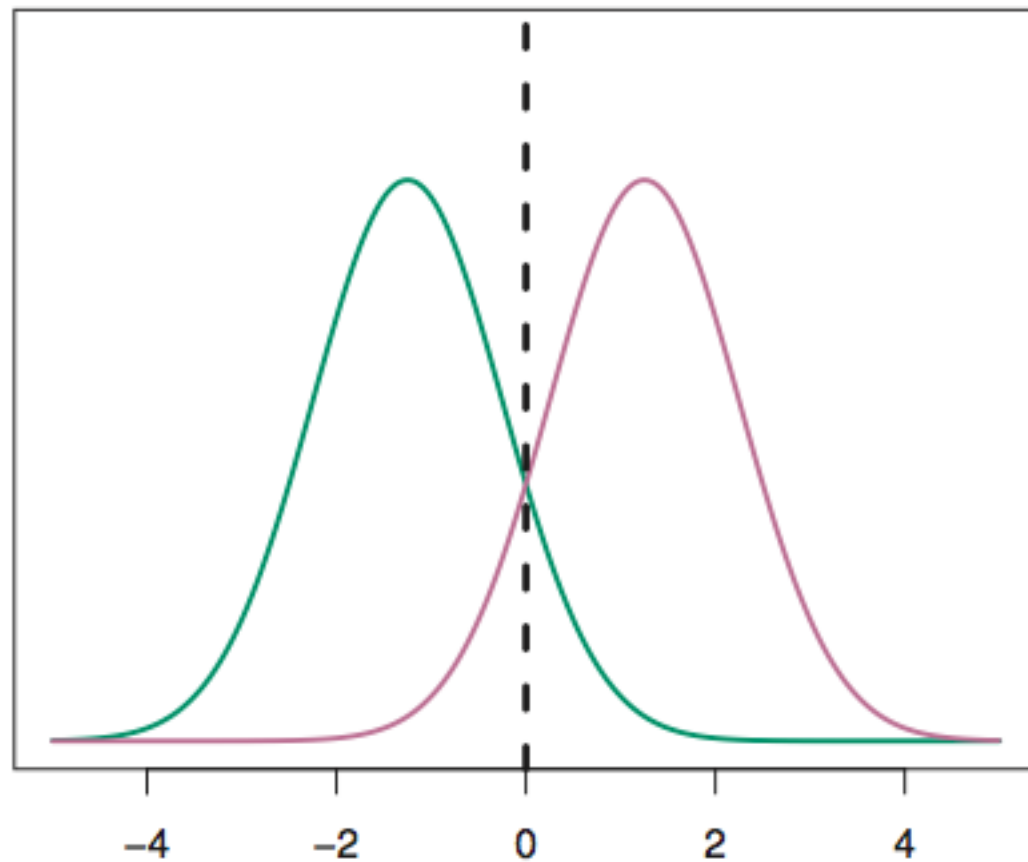
# Draw default prediction for non-student
lines(seq(1:2600),glm.probs, cex=.2, col="green")

# Calculate confusion matrix
glm.pred=rep("No", nrow(Default))
glm.probs=predict(glm.fit2,type="response")
glm.pred[glm.probs > .5]="Yes"
table(glm.pred,Default$default)
```

# Exercise

- a. Change the threshold to predict default from .5 to .4, .3, .2, .1  
Analyze the results in prediction accuracy.

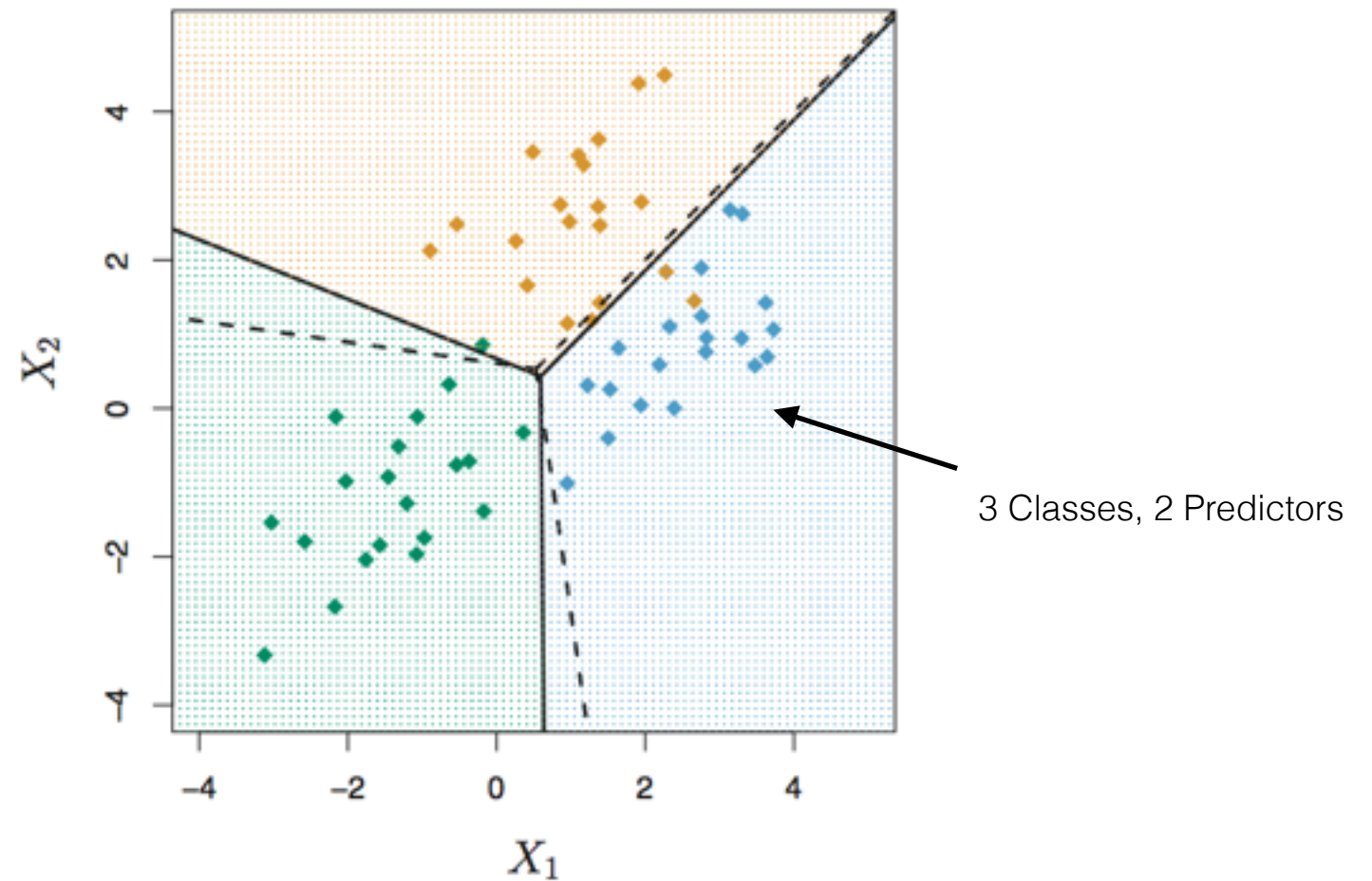
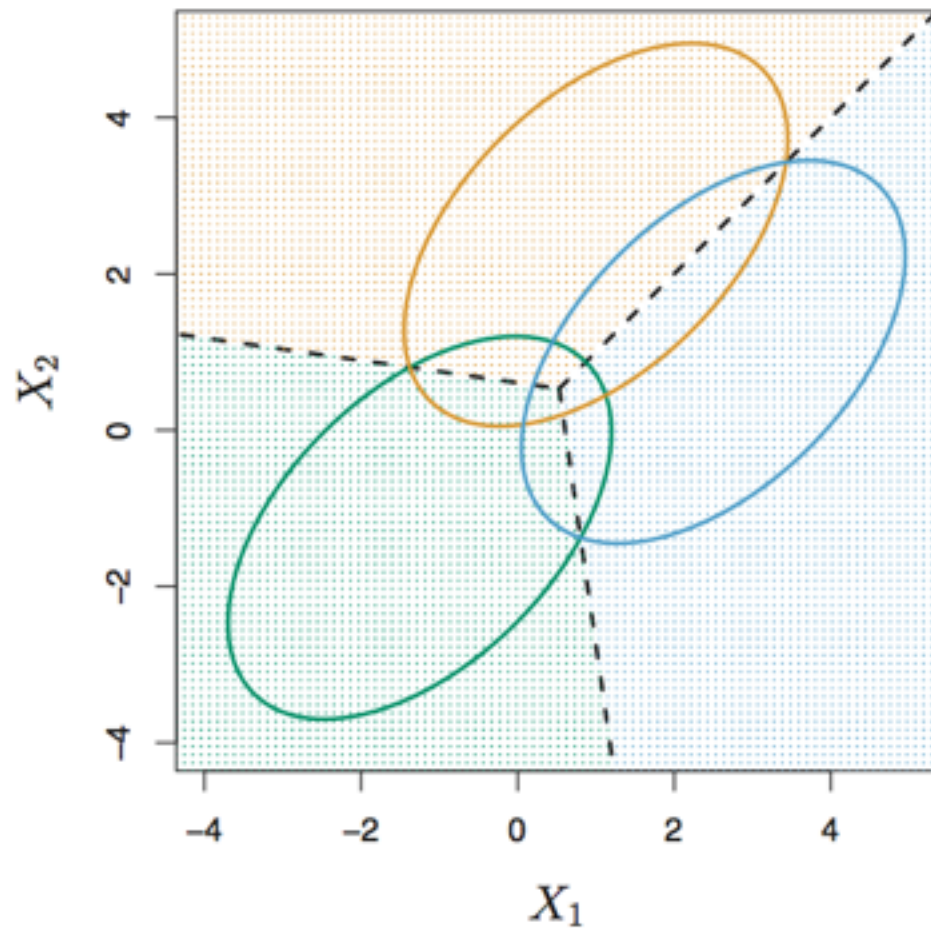
# Linear Discriminant Analysis



Bayes Theorem

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}.$$

# Bayes Classifier



Bayes Theorem

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)}$$

# Bayes Classifier

## Bayes Theorem

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)}$$

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)},$$

where

$$f_k(x) = \Pr(X = x|Y = k)$$

$$\pi_k = \Pr(Y = k)$$

# Lab

- Use Default data to fit a LDA & QDA models  
(Predict Probability to Default based on Balance)



# Lab - Use LDA Predict Default

```
library(ISLR)
library(MASS)

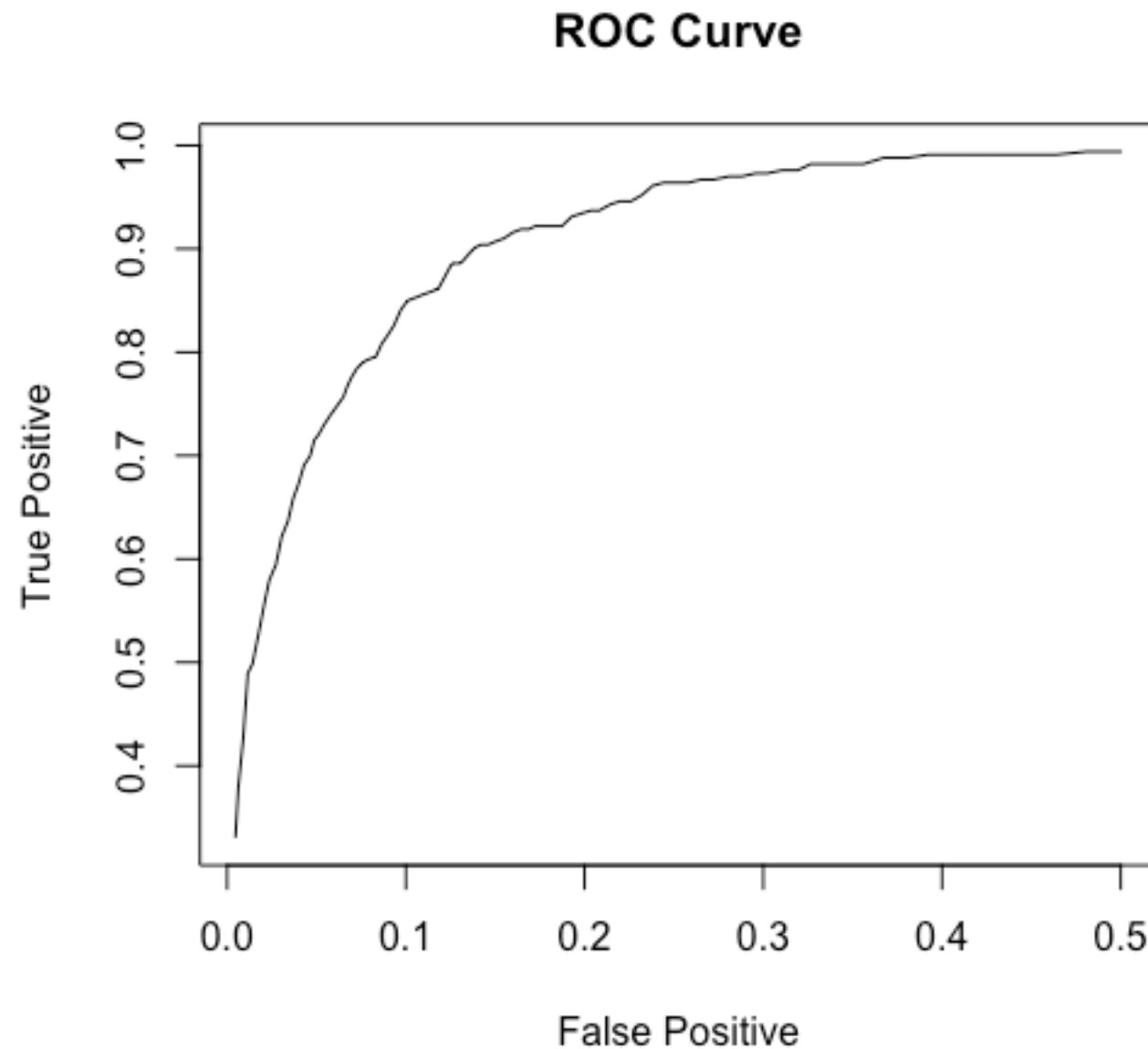
lda.fit=lda(default~balance,data=Default)
lda.pred=predict(lda.fit, Default)
table(lda.pred$class , Default$default)

threshold=seq(.05, .9, .01)
n=length(threshold)
TP = rep(n,0.0)
FP = rep(n,0.0)

for(i in 1:n){
  lda.fit=lda(default~.,data=Default, prior=c(1-threshold[i],threshold[i]))
  lda.pred=predict(lda.fit, Default)
  conf_tab <- table(lda.pred$class , Default$default)
  FP[i]= conf_tab[2,1] / sum(conf_tab[,1])
  TP[i]= conf_tab[2,2] / sum(conf_tab[,2])
}

plot(FP,TP,type="l", xlab="False Positive", ylab="True Positive")
title("ROC Curve")
```

# Lab - Use LDA Predict Default



# Lab - Use LDA Predict Default

```
library(ISLR)
library(MASS)

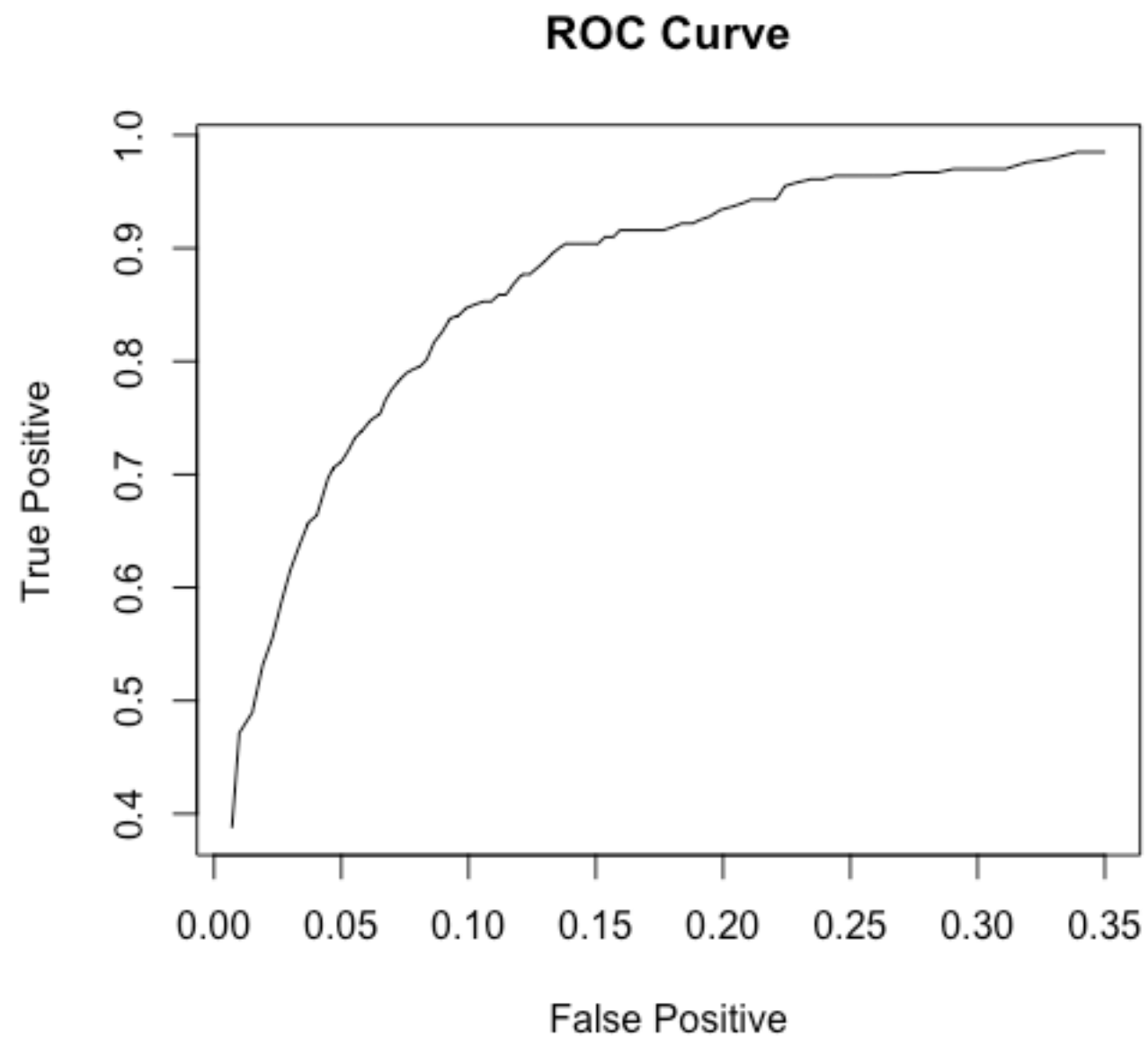
qda.fit=qda(default~balance,data=Default)
qda.pred=predict(qda.fit, Default)
table(qda.pred$class , Default$default)

threshold=seq(.05, .9, .01)
n=length(threshold)
TP = rep(n,0.0)
FP = rep(n,0.0)

for(i in 1:n){
  qda.fit=qda(default~.,data=Default, prior=c(1-threshold[i],threshold[i]))
  qda.pred=predict(qda.fit, Default)
  conf_tab <- table(qda.pred$class , Default$default)
  FP[i]= conf_tab[2,1] / sum(conf_tab[,1])
  TP[i]= conf_tab[2,2] / sum(conf_tab[,2])
}

plot(FP,TP,type="l", xlab="False Positive", ylab="True Positive")
title("ROC Curve")
```

# Lab - Use LDA Predict Default



# Ggplot2 – Visualization

# Ggplot2 - Visualization

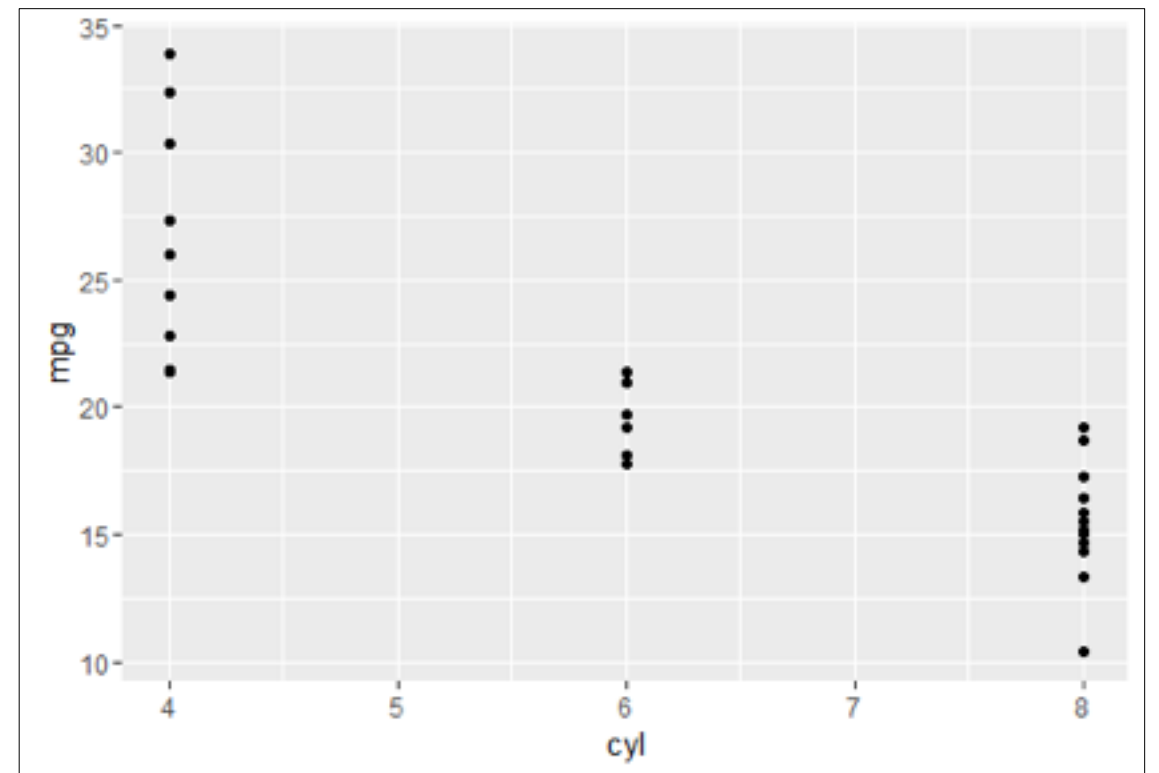
- Geoms :short for geometric objects, describe the type of plot you will produce.
- geom\_point : Points, as for a scatterplot
- aes : Generate aesthetic mappings that describe how variables in the data are
- alpha : technique is to use transparent points

# Ggplot2 - Visualization

```
# Load the ggplot2 package
library(ggplot2)

# structure of mtcars package
str(mtcars)

#plot using ggplot
ggplot(mtcars, aes(x = cyl, y = mpg)) +
  geom_point()
```



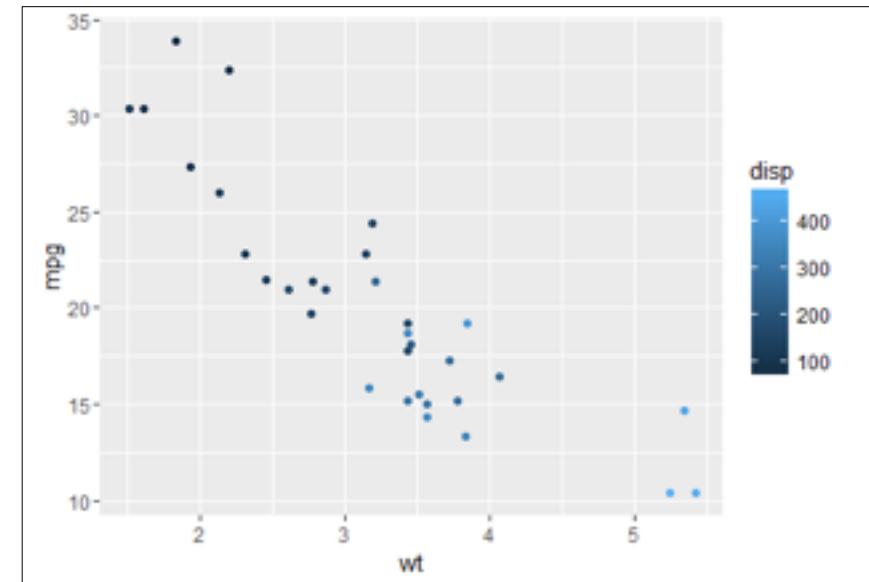
**geoms** : short for geometric objects, describe the type of plot you will produce.

**geom\_point** : Points, as for a scatterplot

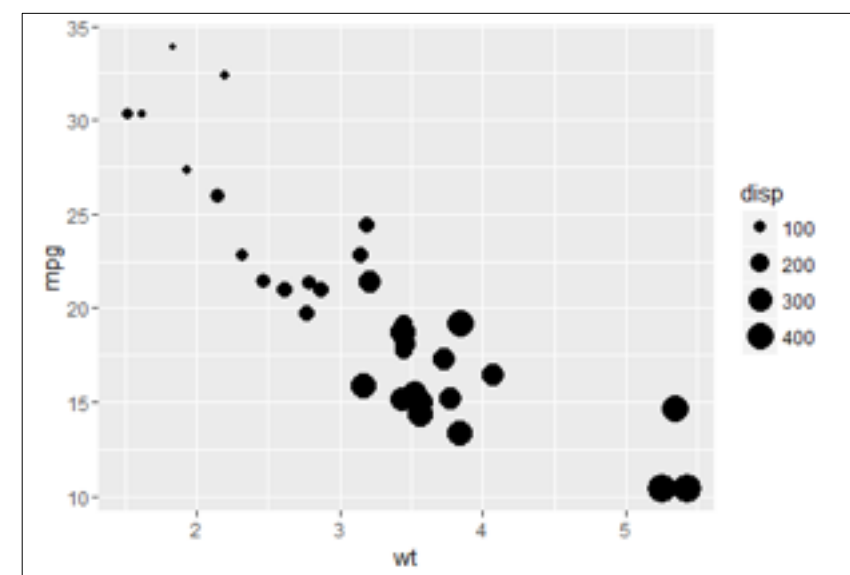
**aes** : Generate aesthetic mappings that describe how variables in the data are

# Ggplot2 - Visualization

```
#select the color based on 'disp' variable  
ggplot(mtcars, aes(x = wt, y = mpg, col =  
disp)) +  
  geom_point()
```



```
#display the plot with size of point based on  
disp variable  
ggplot(mtcars, aes(x = wt, y = mpg, size =  
disp)) +  
  geom_point()
```





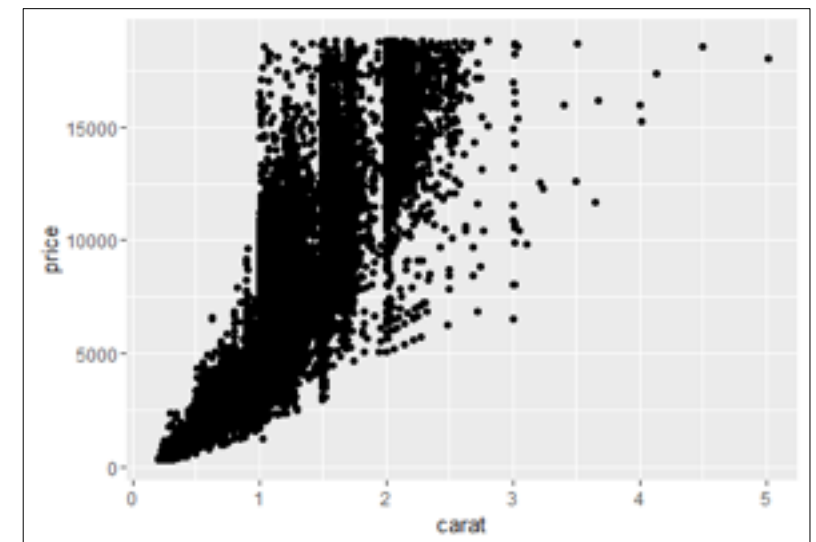
# Ggplot2 – Visualization on Diamond dataset

# Exploring the dataset

```
data("diamonds")  
summary(diamonds)  
str(diamonds)  
nrow(diamonds)  
ncol(diamonds)  
head(diamonds)
```

## Basic Scatter Plot

```
ggplot(diamonds, aes(x=carat, y=price)) +  
geom_point()
```



# Scatter Plot

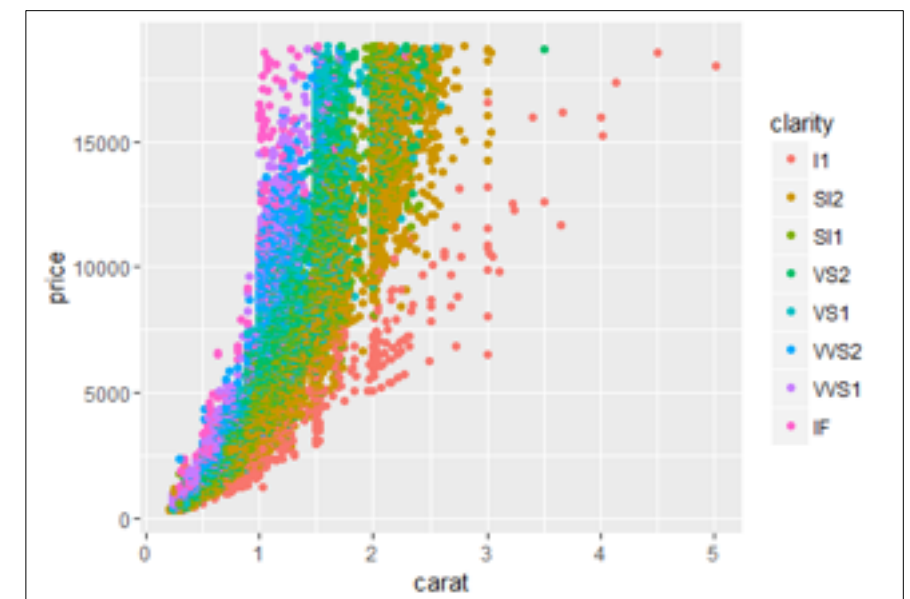
setting the title, x label and ylabel

```
ggplot(diamonds, aes(x=carat, y=price)) +  
  geom_point() +  
  ggtitle("My scatter plot") +  
  xlab("Weight (carats)") +  
  ylab("Price(USD)")
```



Scatter Plot with adding 3<sup>rd</sup> parameter as color

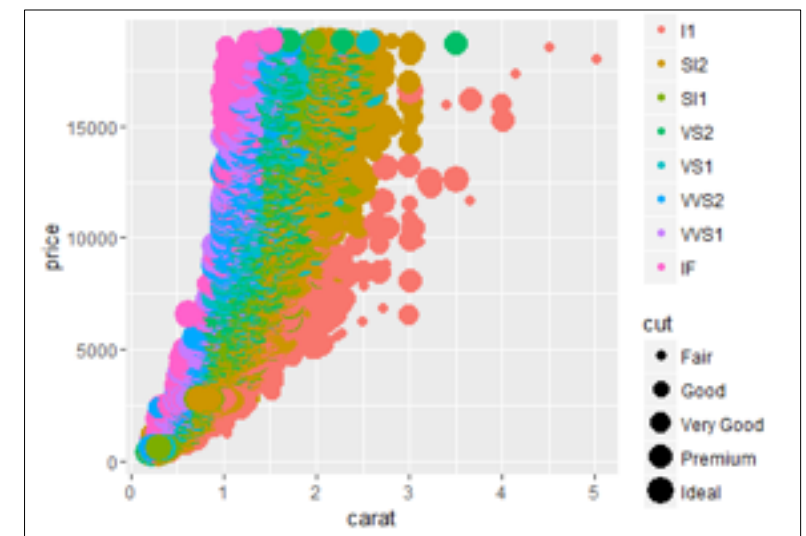
```
ggplot(diamonds, aes(x=carat, y=price, col =  
clarity)) + geom_point()
```



# Scatter Plot

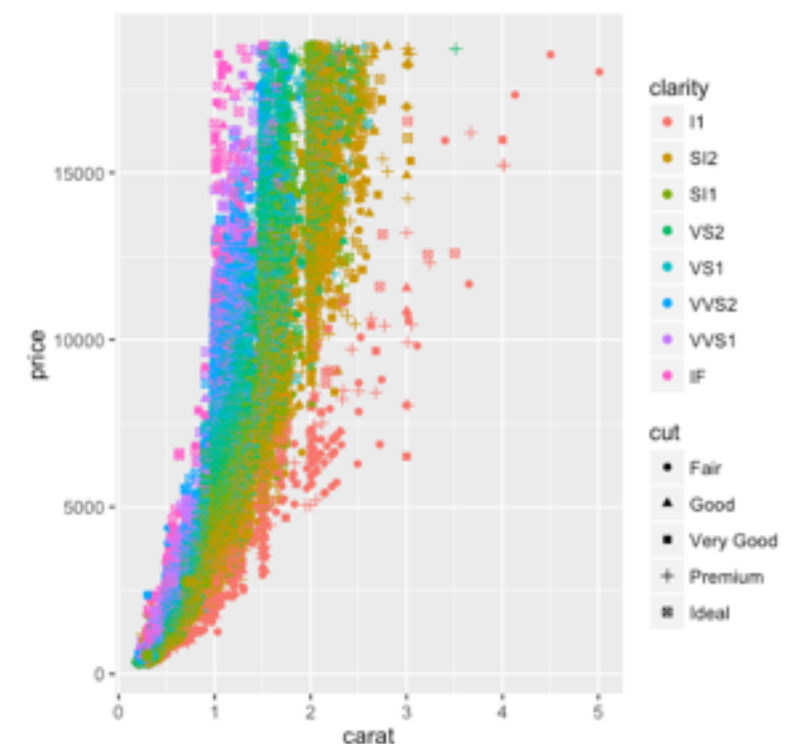
Scatter Plot with adding 4th parameter as **SIZE** of the variable

```
ggplot(diamonds, aes(x=carat, y=price,  
color=clarity, size=cut)) + geom_point()
```



Scatter Plot with adding parameter as **SHAPE** of the variable

```
ggplot(diamonds, aes(x=carat, y=price,  
color=clarity, shape=cut) ) + geom_point()
```



# Scatter Plot

## `geom_smooth` :

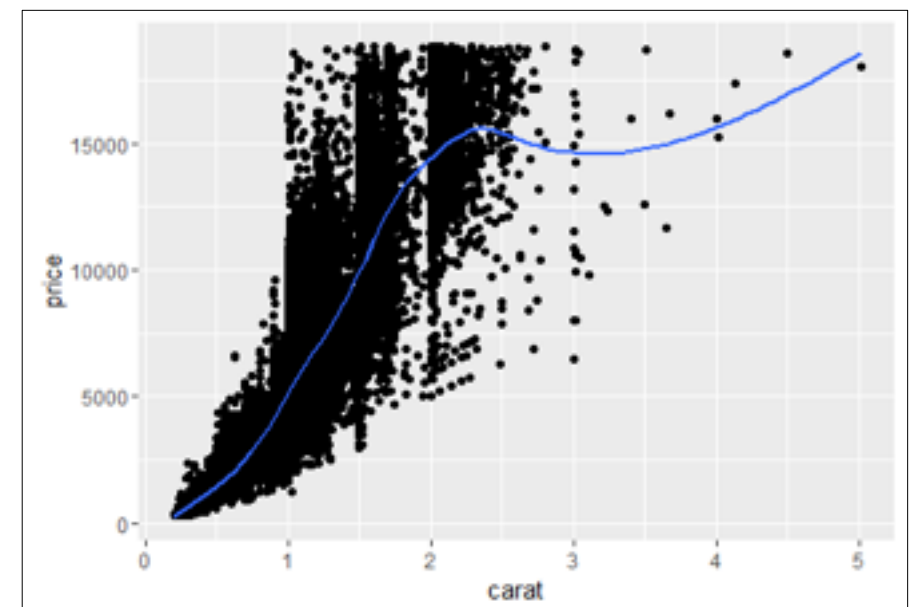
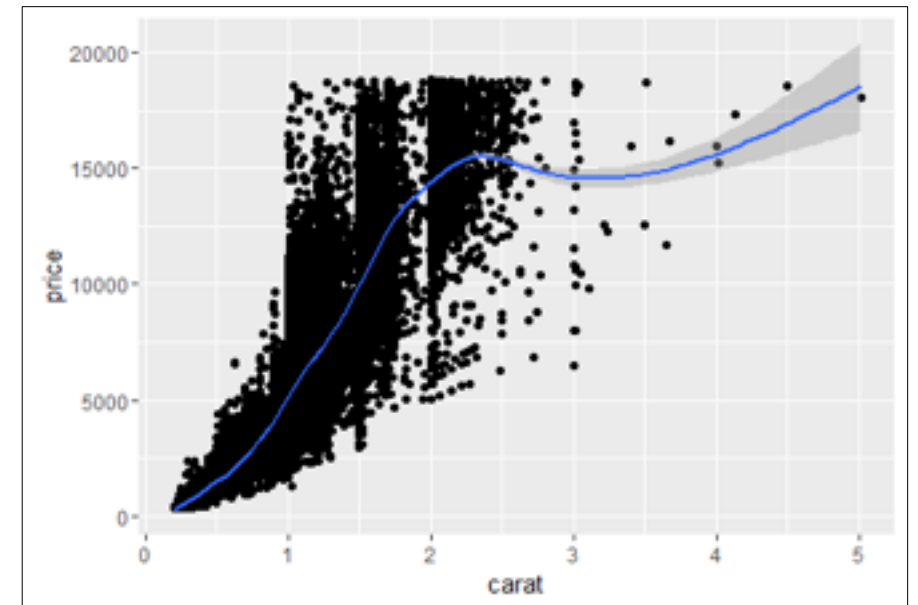
To add a smoothing curve that shows the general trend of the data

```
ggplot(diamonds, aes(x=carat, y=price)) +  
geom_point() + geom_smooth()
```

The gray area around the curve is a confidence interval, suggesting how much uncertainty there is in this smoothing curve.

To turn off the confidence interval, use  
“`geom_smooth(se=FALSE)`”

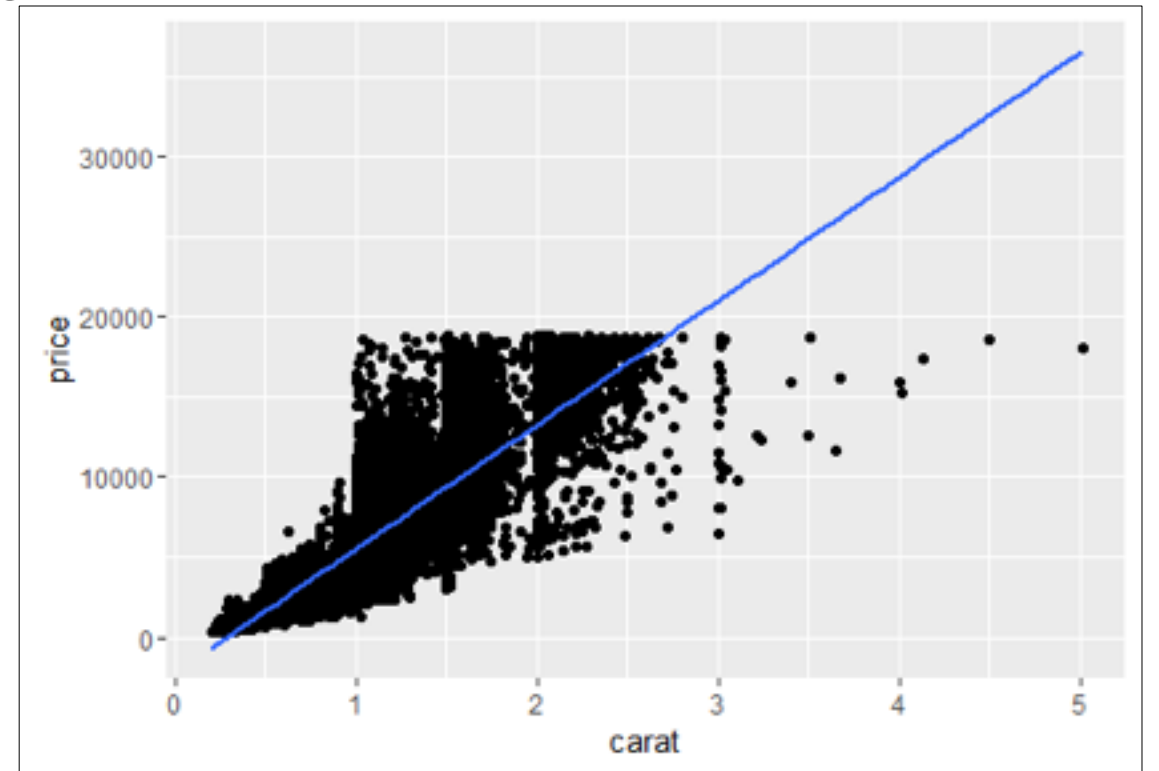
```
ggplot(diamonds, aes(x=carat, y=price)) +  
geom_point() + geom_smooth(se=FALSE)
```



# Scatter Plot

To fit a linear model and displaying best fit straight line

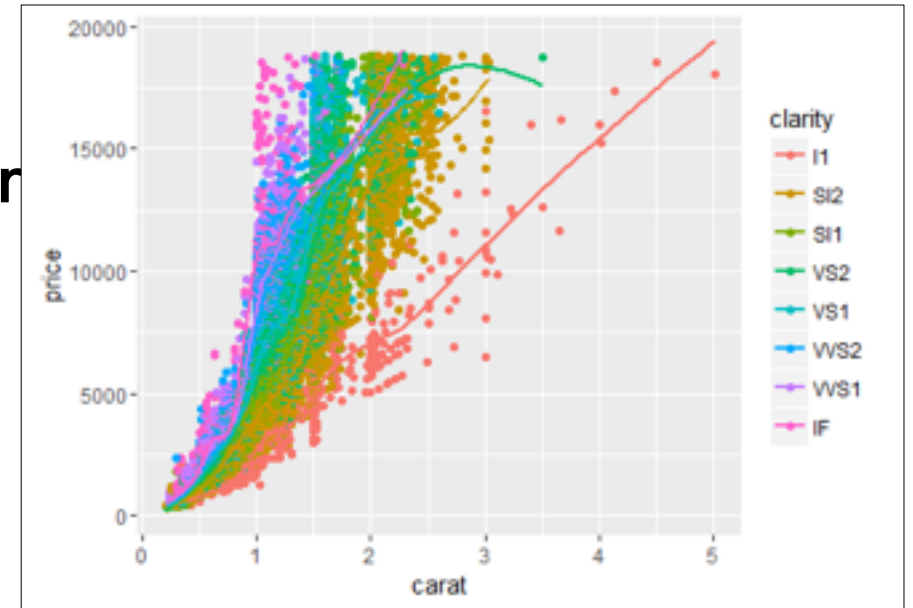
```
ggplot(diamonds, aes(x=carat, y=price)) +  
  geom_point() + geom_smooth(method = "lm",  
    se=FALSE)
```



# Scatter Plot

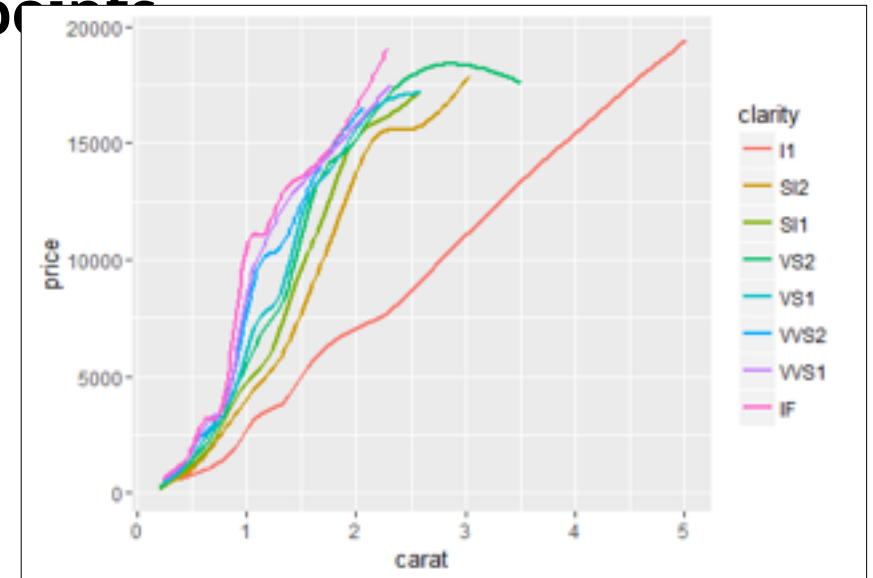
to plot multiple trend lines based on color parameter

```
ggplot(diamonds, aes(x=carat, y=price,  
color=clarity)) + geom_point() +  
geom_smooth(se=FALSE)
```



tidying up the data to analyze trends by removing points

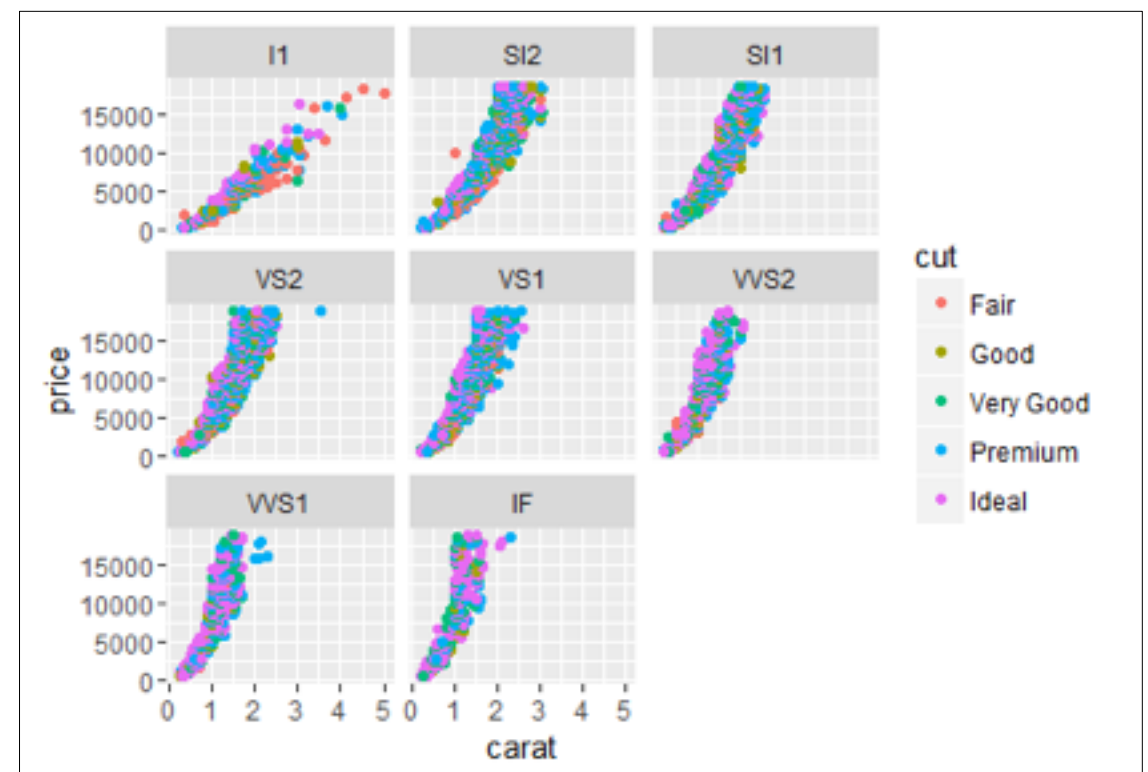
```
ggplot(diamonds, aes(x=carat, y=price,  
color=clarity)) + geom_smooth(se=FALSE)
```



# Plotting Multiple plots

the "facet\_wrap" function

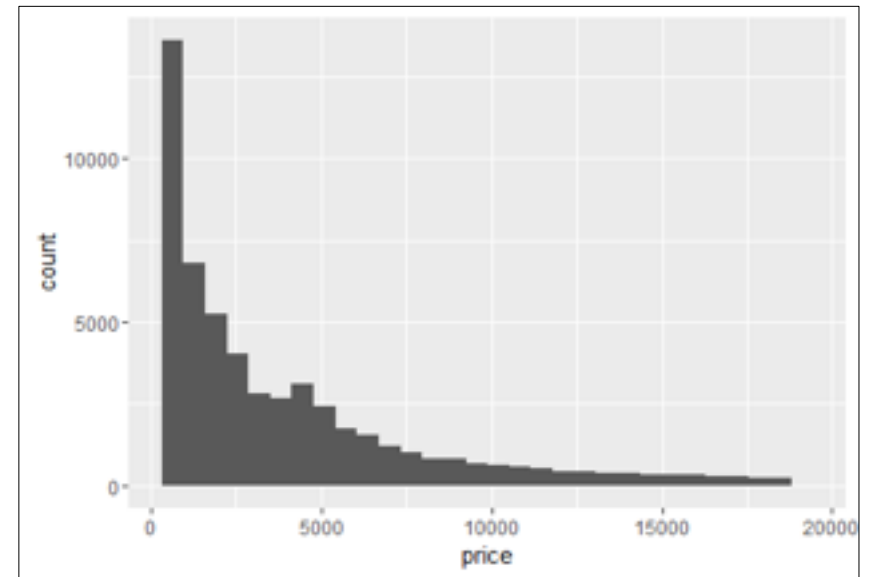
```
ggplot(diamonds, aes(x=carat, y=price,  
color=cut)) + geom_point() + facet_wrap(~  
clarity)
```



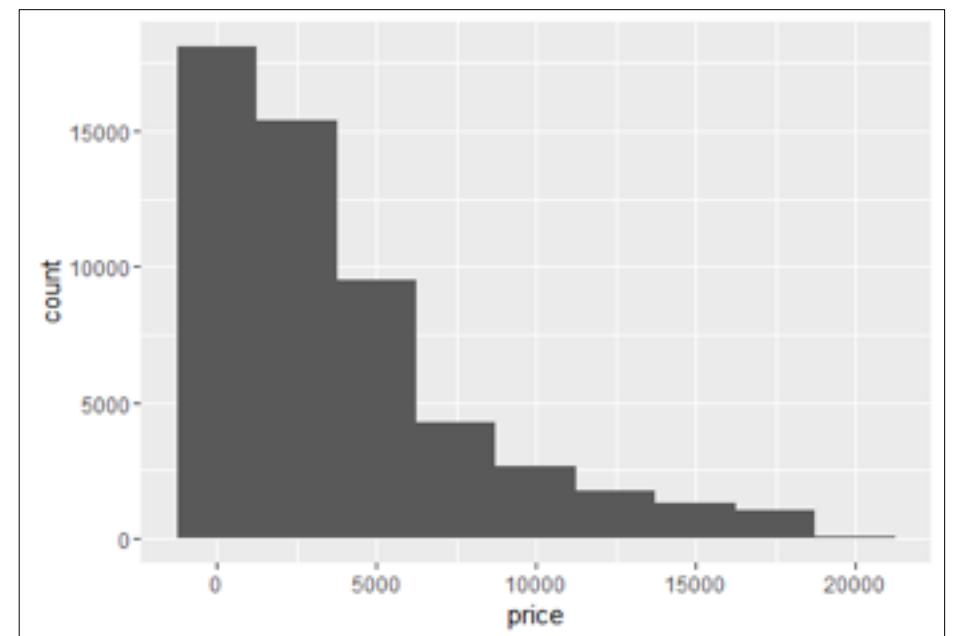


# Histogram

```
ggplot(diamonds, aes(x=price)) +  
geom_histogram()
```



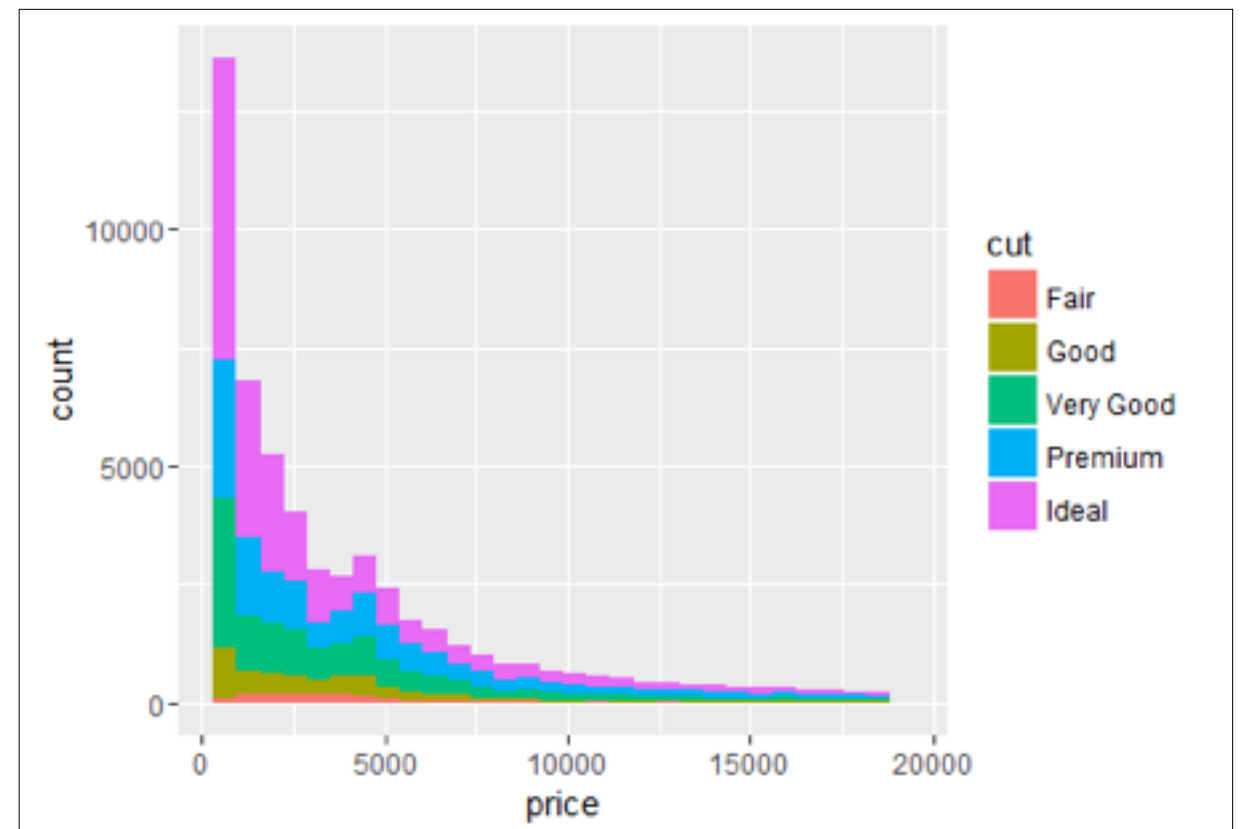
```
ggplot(diamonds, aes(x=price)) +  
geom_histogram(binwidth = 2000)
```



# Histogram

Stacked histogram using fill aesthetic

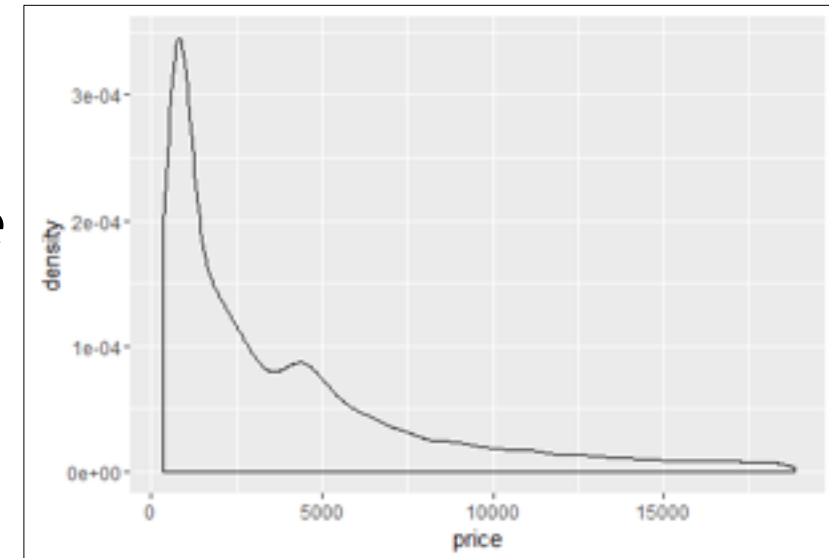
```
ggplot(diamonds, aes(x=price,  
fill=cut)) + geom_histogram()
```



# Density Distribution

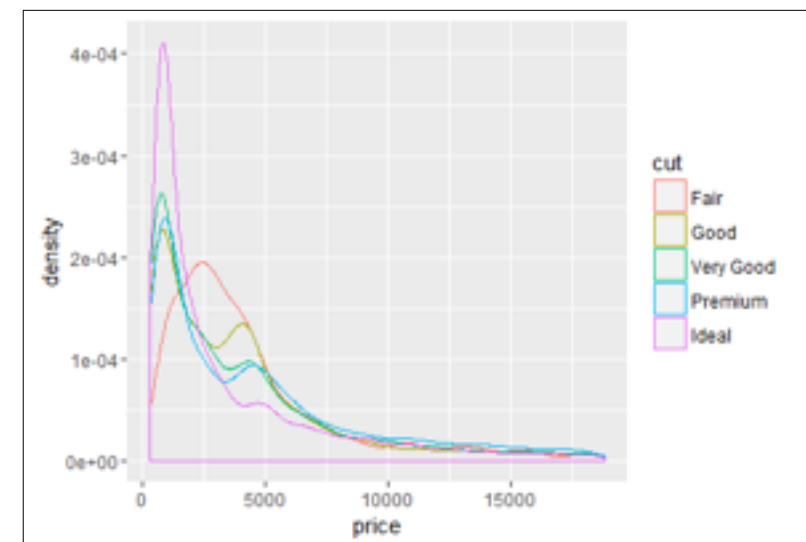
**geom\_density** : to view the distribution is as a density curve

```
ggplot(diamonds, aes(x=price)) +  
geom_density()
```



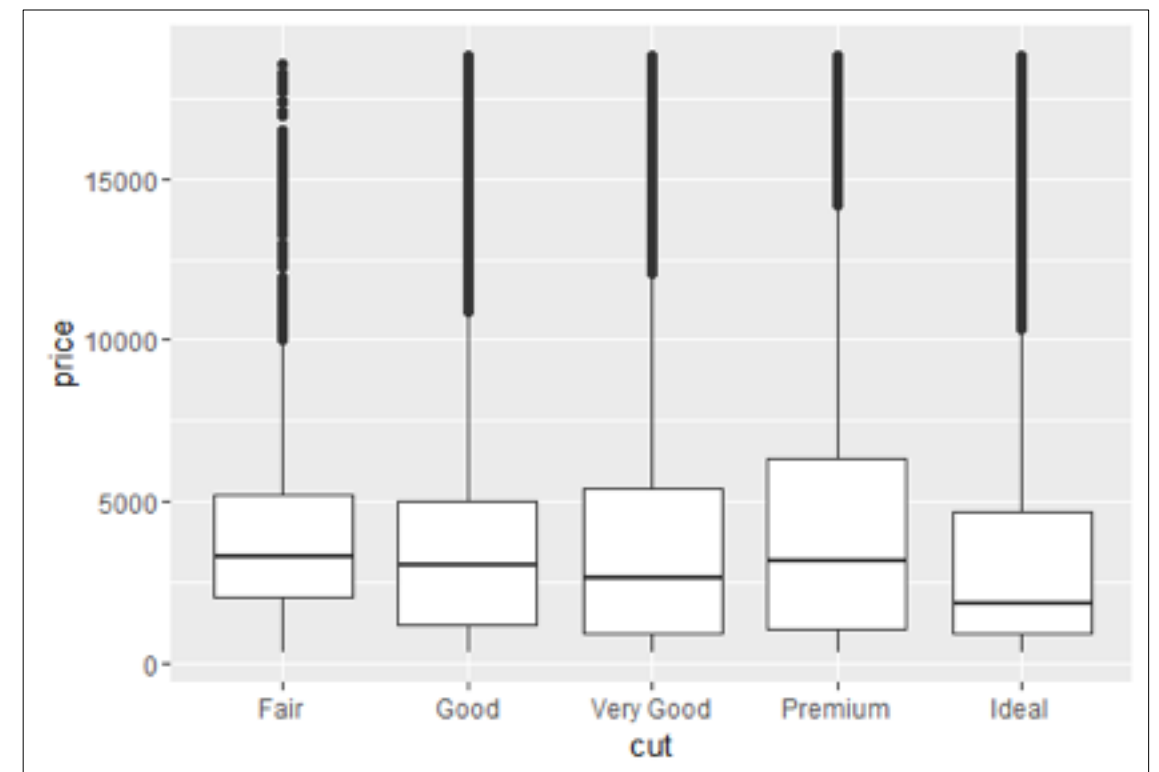
To divide the density curve up based on one of the attributes

```
ggplot(diamonds, aes(x=price, color=cut)) +  
geom_density()
```



# Boxplot

```
ggplot(diamonds, aes(x=color, y=price)) +  
geom_boxplot()
```



# More slides

# Multiple Linear Regression

- $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$
- What to consider
  - Is any of predictors useful? F Statistics
  - Which predictors are useful?
  - Using qualitative predictors
  - Interactions

# Multiple Linear Regression

- Is any of predictors useful?

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$$

versus the alternative

$$H_a : \text{at least one } \beta_j \text{ is non-zero.}$$

This hypothesis test is performed by computing the *F-statistic*,

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n - p - 1)},$$

- Check F Static corresponding p-value to reject hypothesis

# Multiple Linear Regression

- Which predictors useful?

	Coefficient	Std. error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	−0.001	0.0059	−0.18	0.8599

- Use
  - Forward selection
  - Backward selection
  - Mixed



# Multiple Linear Regression

- Using Qualitative predictors

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is female} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is male.} \end{cases}$$

$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ 0 & \text{if } i\text{th person is male,} \end{cases}$$

# Multiple Linear Regression

- Using Qualitative predictors

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i \\ \beta_0 + \beta_2 + \epsilon_i \\ \beta_0 + \epsilon_i \end{cases}$$

$$x_{i1} = \begin{cases} 1 & \text{if } i\text{th person is Asian} \\ 0 & \text{if } i\text{th person is not Asian,} \end{cases}$$

$$x_{i2} = \begin{cases} 1 & \text{if } i\text{th person is Caucasian} \\ 0 & \text{if } i\text{th person is not Caucasian.} \end{cases}$$

# Multiple Linear Regression

- Using Qualitative predictors

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i \\ \beta_0 + \beta_2 + \epsilon_i \\ \beta_0 + \epsilon_i \end{cases}$$

$$x_{i1} = \begin{cases} 1 & \text{if } i\text{th person is Asian} \\ 0 & \text{if } i\text{th person is not Asian,} \end{cases}$$

$$x_{i2} = \begin{cases} 1 & \text{if } i\text{th person is Caucasian} \\ 0 & \text{if } i\text{th person is not Caucasian.} \end{cases}$$

	Coefficient	Std. error	t-statistic	p-value
Intercept	531.00	46.32	11.464	< 0.0001
ethnicity[Asian]	-18.69	65.02	-0.287	0.7740
ethnicity[Caucasian]	-12.50	56.68	-0.221	0.8260

# Multiple Linear Regression

- Interactions

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon.$$

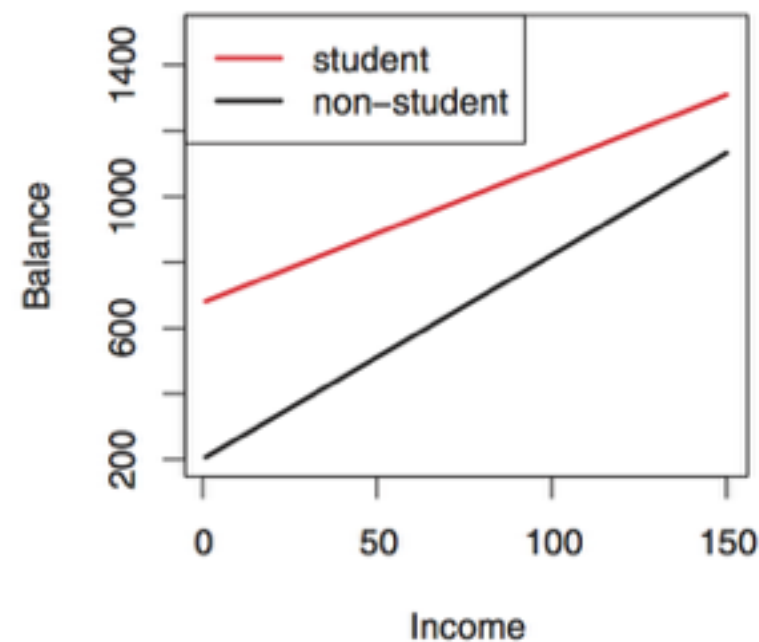
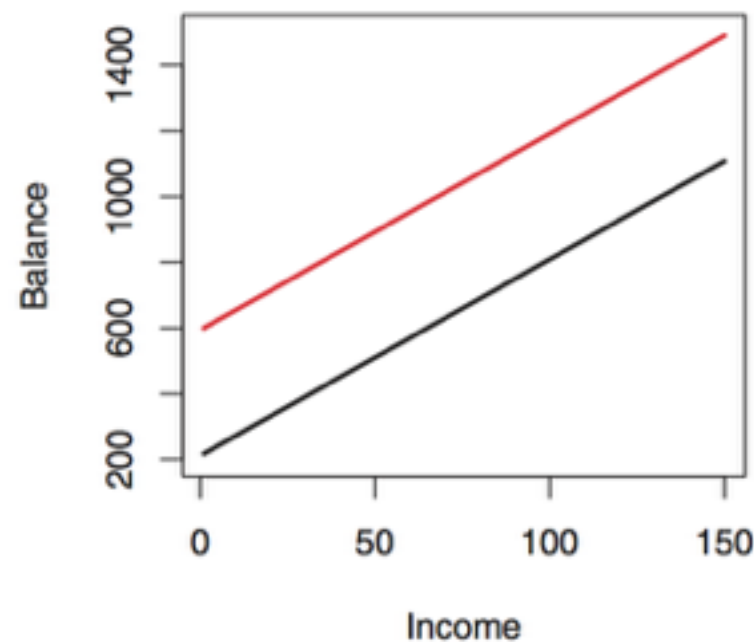
	Coefficient	Std. error	t-statistic	p-value
Intercept	6.7502	0.248	27.23	< 0.0001
TV	0.0191	0.002	12.70	< 0.0001
radio	0.0289	0.009	3.24	0.0014
TV×radio	0.0011	0.000	20.73	< 0.0001

Should we remove?

$$\begin{aligned}\text{units} &\approx 1.2 + 3.4 \times \text{lines} + 0.22 \times \text{workers} + 1.4 \times (\text{lines} \times \text{workers}) \\ &= 1.2 + (3.4 + 1.4 \times \text{workers}) \times \text{lines} + 0.22 \times \text{workers}.\end{aligned}$$

# Multiple Linear Regression

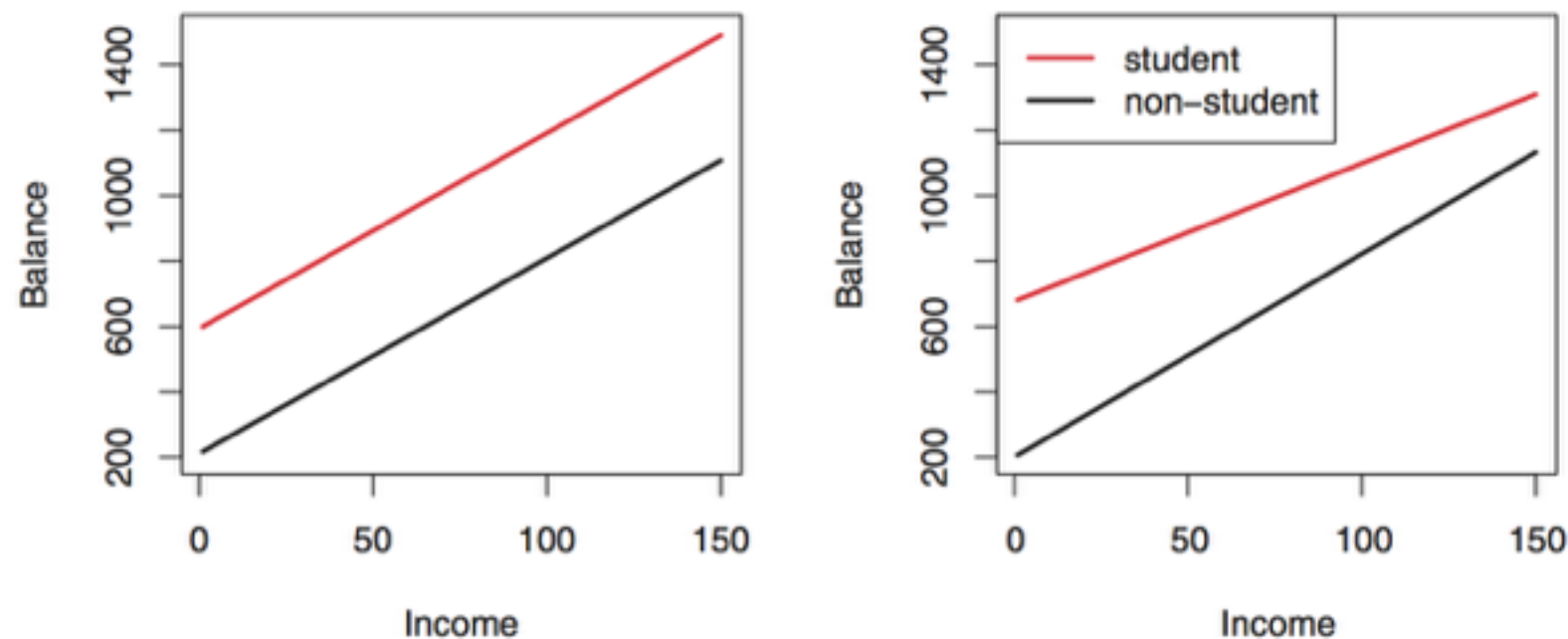
- Interactions



$$\begin{aligned} \text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 & \text{if } i\text{th person is a student} \\ 0 & \text{if } i\text{th person is not a student} \end{cases} \\ &= \beta_1 \times \text{income}_i + \begin{cases} \beta_0 + \beta_2 & \text{if } i\text{th person is a student} \\ \beta_0 & \text{if } i\text{th person is not a student.} \end{cases} \end{aligned}$$

# Multiple Linear Regression

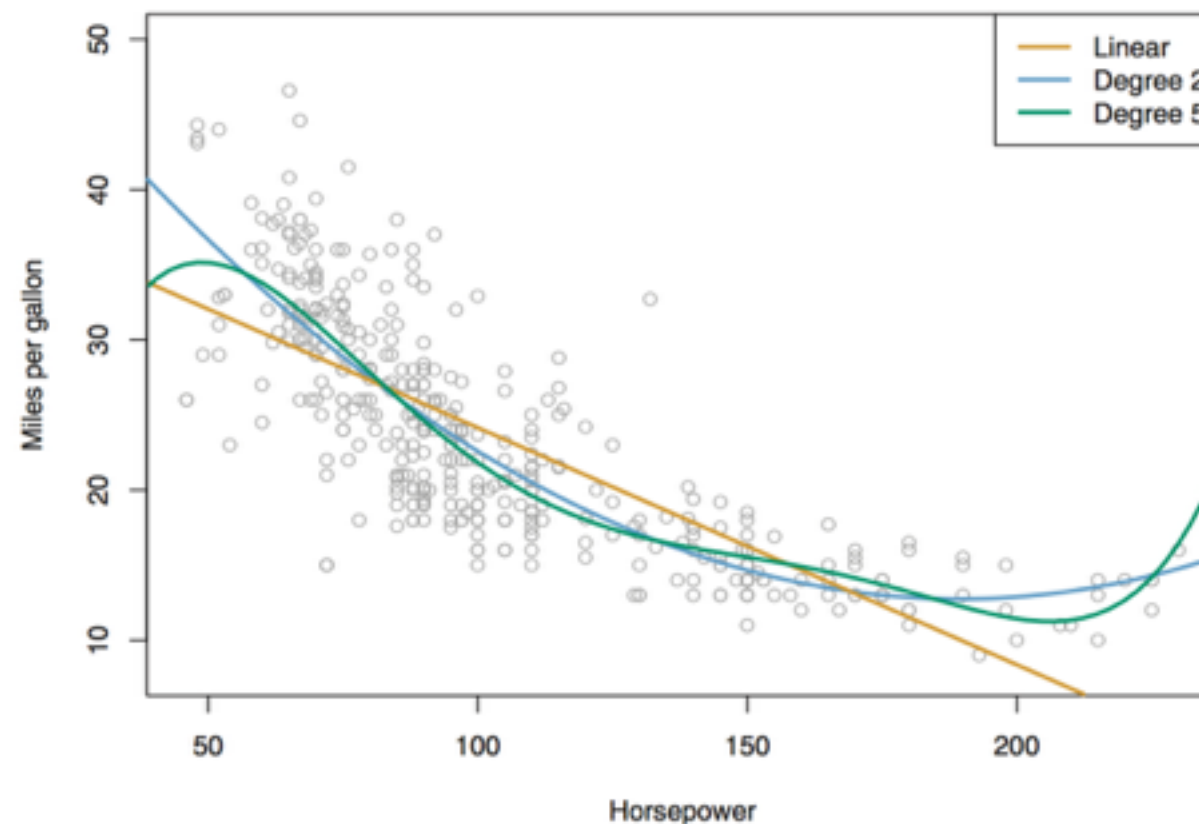
- Interactions



$$\begin{aligned} \text{balance}_i &\approx \beta_0 + \beta_1 \times \text{income}_i + \begin{cases} \beta_2 + \beta_3 \times \text{income}_i & \text{if student} \\ 0 & \text{if not student} \end{cases} \\ &= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \text{income}_i & \text{if student} \\ \beta_0 + \beta_1 \times \text{income}_i & \text{if not student} \end{cases} \end{aligned}$$

# Linear Regression - extended

- Non-Linear relationships



$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower} + \beta_2 \times \text{horsepower}^2 + \epsilon$$

	Coefficient	Std. error	t-statistic	p-value
Intercept	56.9001	1.8004	31.6	< 0.0001
horsepower	-0.4662	0.0311	-15.0	< 0.0001
horsepower <sup>2</sup>	0.0012	0.0001	10.1	< 0.0001

# Exercise - Simple Linear Model

- Use firearms.txt data to fit a linear model (Predict Rate based on Year)
- Use firearms.txt data from <http://www.statsci.org/data/first.html>



# Exercise - Boston Dataset

a) Load in the Boston data set. The Boston data set is part of the MASS library in R

```
> library(MASS)
```

b) Read about the data set

```
> ?Boston
```

c) How many rows are in this data set? How many columns? What do the rows and columns represent?

```
> nrow(Boston)
```

```
> ncol(Boston)
```

d) Make some pairwise scatterplots of the predictors (columns) in this data set. Describe your findings.

```
> attach(Boston)
```

```
> plot(age~crim)
```

```
> plot(zn~crim)
```

# Exercise - Boston Dataset

e) Which predictors appear to be associated with per capita crime rate? What are the apparent relationships?

f) How many of the suburbs in the Boston data set bound the Charles river?

```
> nrow(Boston[Boston$chas==1,])
```

g) What is the median pupil-teacher ratio among the towns in the Boston data set?

```
> median(Boston$ptratio)
```

h) In the Boston data set, how many of the suburbs average more than seven rooms per dwelling?  
More than eight rooms per dwelling?

```
> df = data.frame(Boston)
```

```
> nrow(df[df$rm>7,])
```

```
> nrow(df[df$rm>8,])
```

# Exercise - Boston Dataset

i) Comments on suburbs that average >8 rooms / dwelling

```
> summary(df$crim)
```

```
> summary(df[df$rm>8,1])
```

```
> summary(df$age)
```

```
> summary(df[df$rm>8,7])
```

```
> summary(df$black)
```

```
> summary(df[df$rm>8,12])
```

# Exercise - Boston Dataset

j) Develop a linear regression model to predict per capita crime rate by town using the other variables in the Boston data set. Use 80% of the data as training dataset. The rest will be test dataset.

```
> idx = sample(1:nrow(Boston), nrow(Boston)*.8, replace = FALSE)
> train = Boston[idx,]
> test = Boston[-idx,]
```

k) Find Training MSE and Test MSE for the model

```
> fit=lm(crim~.,train)
> ytrain = train$crim
> ytrainp = predict(fit,train[2:14])
> mean((ytrain - ytrainp)^2)
> ytest = test$crim
> ytestp = predict(fit,test[2:14])
> mean((ytest - ytestp)^2)
```

l) Comment on the results obtained. How accurately can we predict per capita crime rate by town? What are the most important predictors?

```
> summary (fit)
```

# Linear Regression Lab - Boston dataset

- Use Boston data to fit a linear model (Predict nox using distance)

# Linear Regression Lab - Boston dataset

```
#predict nox using dis - cubic polynomial regression and plot the line
```

```
# Get training dataset
```

```
library(MASS)
```

```
attach(Boston)
```

```
# Build Linear Model
```

```
#fit=lm(nox~poly(dis, 3),data=Boston)
```

```
fit=lm(nox~dis+I(dis^2)+I(dis^3),data=Boston)
```

```
# View the Model
```

```
summary(fit)
```

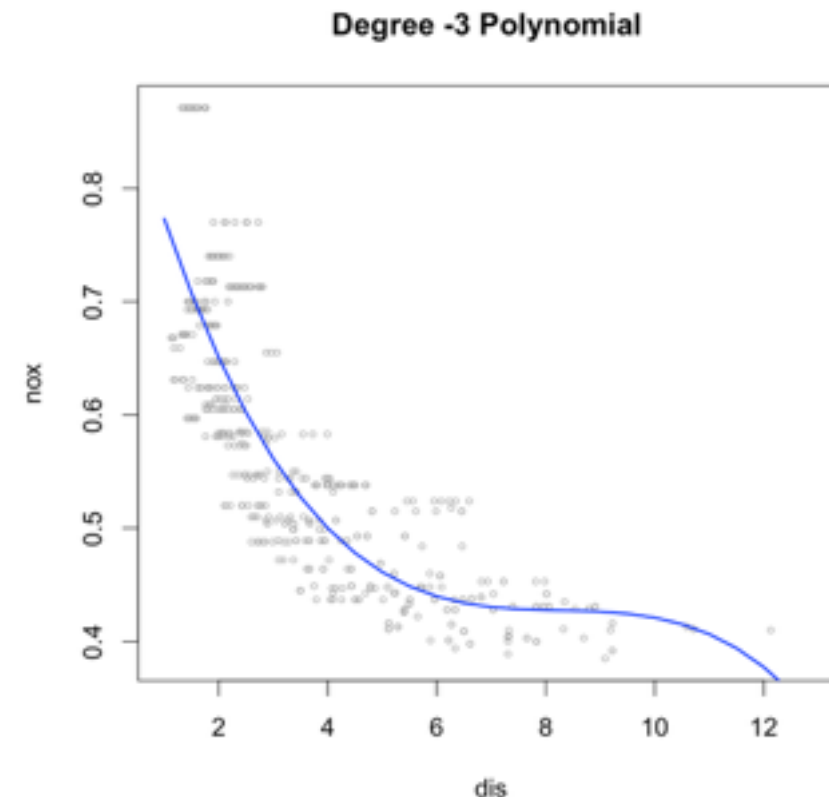
```
dis.grid = seq(round(range(dis)[1],0), round(range(dis)[2]+.5,0), .5)  
dis.range = c(dis.grid[1], dis.grid[length(dis.grid)])
```

```
# Plot the model results
```

```
preds=predict(fit,newdata=list(dis=dis.grid),se=TRUE)  
se.bands=cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit )
```

```
plot(dis,nox,xlim=dis.range ,cex=.5,col="darkgrey")  
title("Degree -3 Polynomial ")
```

```
lines(dis.grid,preds$fit,lwd=2,col="blue")
```



# Exercise

- Find the Training and Test MSE for the linear model to predict nox using dis.

# Linear Regression Lab - Auto dataset

- Use Auto data to fit a linear model (Predict mpg using horsepower)



# Linear Regression Lab - Auto dataset

```
#predict nox using dis - cubic polynomial regression and plot the line
```

```
# Get training dataset
```

```
library(ISLR)
```

```
attach(Auto)
```

```
Auto <- na.omit(Auto)
```

```
Auto$horsepower = as.numeric(as.character(Auto$horsepower))
```

```
# Build Linear Model
```

```
fit=lm(mpg~horsepower,data=Auto)
```

```
# View the Model
```

```
summary(fit)
```

```
# Plot the model results
```

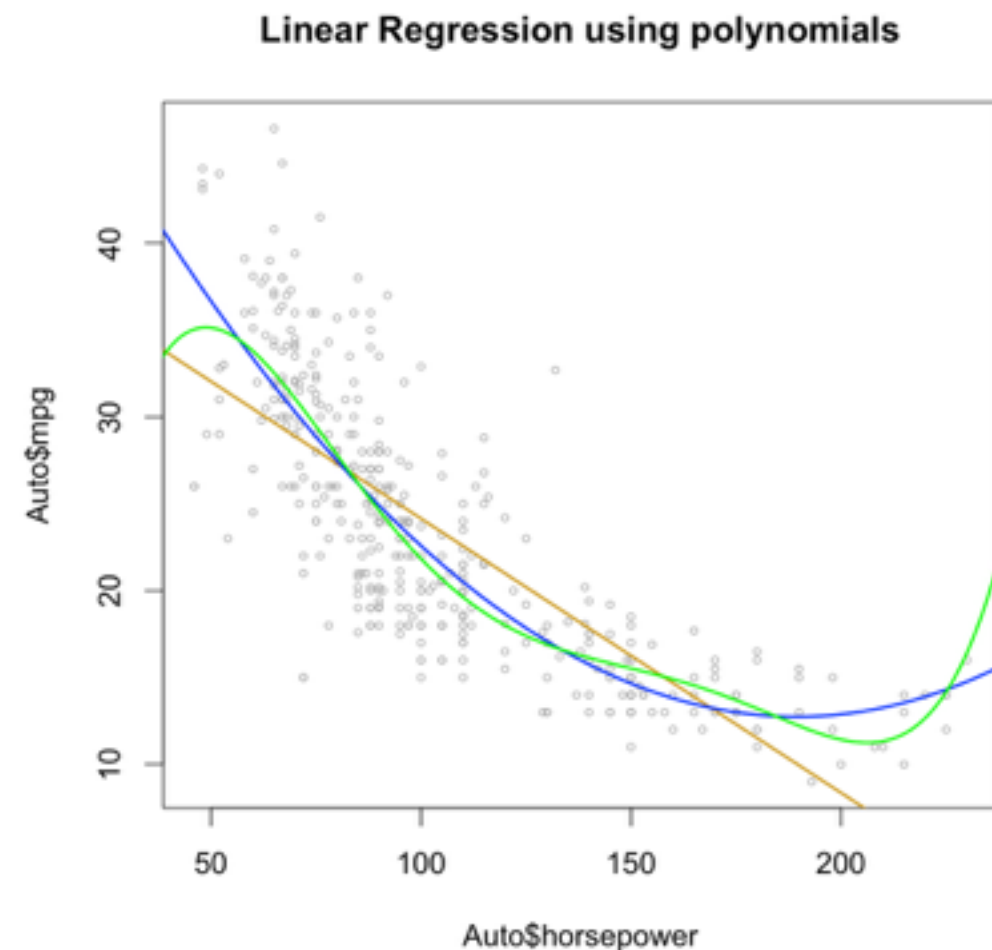
```
x.grid = seq(1:250)
```

```
plot(Auto$horsepower,Auto$mpg,cex=.5,col="darkgrey")
```

```
preds=predict(fit,newdata=list(horsepower=x.grid),se=TRUE)  
se.bands=cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit )
```

```
title("Linear Regression using polynomials")
```

```
lines(x.grid,preds$fit,lwd=2,col="goldenrod3")
```



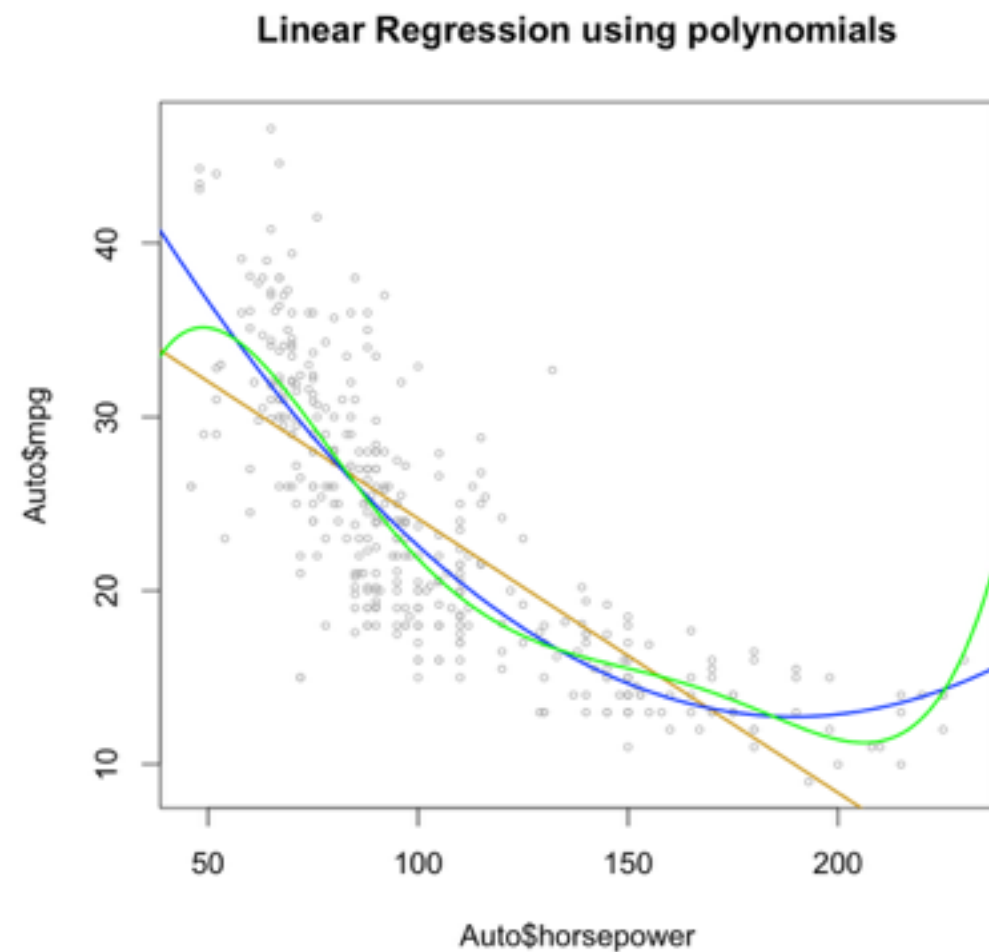
# Linear Regression Lab - Auto dataset

```
# Build Linear Model (polynomial - 2)
```

```
fit=lm(mpg~poly(horsepower,2),data=Auto)
preds=predict(fit,newdata=list(horsepower=x.grid),se=TRUE)
se.bands=cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit )
lines(x.grid,preds$fit,lwd=2,col="blue")
```

```
# Build Linear Model (polynomial - 5)
```

```
fit=lm(mpg~poly(horsepower,5),data=Auto)
preds=predict(fit,newdata=list(horsepower=x.grid),se=TRUE)
se.bands=cbind(preds$fit+2*preds$se.fit,preds$fit-2*preds$se.fit )
lines(x.grid,preds$fit,lwd=2,col="green")
```



# Exercise

- Plot Test MSE for models using polynomials 1 thru 5. Which model would you recommend?

# Exercise

Using Boston dataset, form a new outcome variable  $Y$  which equals one if per capita crime rate by town (crim) is greater than or equal to the overall median and zero otherwise. Fit a logistic regression model to  $Y$  and report the training and test misclassification rates and the most important predictors. Compare the results of this analysis to that of the linear regression approach.

# Logistic Regression

# Lab - Predict Default

- Use Default data to fit a logistic regression model (Predict Probability to Default based on Balance)

# Lab

```
library(ISLR)

# Analyze Default Data
par(mfrow=c(1,2))

boxplot(Default$balance~Default$student, xlab="Student", ylab="Balance", col=c("blue","brown"))
boxplot(Default$income~Default$student, xlab="Student", ylab="Income", col=c("blue","brown"))

par(mfrow=c(1,1))

# Plot training data: Default vs. Balance
default.prob <- ifelse(Default$default=="Yes", 1, 0)
plot(Default$balance, default.prob, col="goldenrod3", pch="'", xlab="Balance", ylab="Probability to Default")

# Fit logit model: Default vs. Balance
glm.fit <- glm(default~balance,data=Default,family=binomial)

# Predict default for balances 1 to 2600
glm.probs=predict(glm.fit,newdata=data.frame(balance=seq(1:2600)),type="response")

# Draw default prediction
lines(seq(1:2600),glm.probs, cex=.2, col="blue")

# Calculate confusion matrix
glm.pred=rep("No", nrow(Default))
glm.probs=predict(glm.fit,type="response")
glm.pred[glm.probs > .5]="Yes"
table(glm.pred,Default$default)
```

# Lab

```
# Plot training data: Default vs. Balance and Student
default.prob <- ifelse(Default$default=="Yes", 1, 0)
plot(Default$balance, default.prob, col="goldenrod3", pch="", xlab="Balance", ylab="Probability to Default")

# Fit logit model: Default vs. Balance
glm.fit2 <- glm(default~balance+student,data=Default,family=binomial)

# Predict default for balances 1 to 2600 and student=Yes
glm.probs=predict(glm.fit2,newdata=data.frame(balance=seq(1:2600),student='Yes'),type="response")

# Draw default prediction for student
lines(seq(1:2600),glm.probs, cex=.2, col="blue")

# Predict default for balances 1 to 2600 and student=No
glm.probs=predict(glm.fit2,newdata=data.frame(balance=seq(1:2600),student='No'),type="response")

# Draw default prediction for non-student
lines(seq(1:2600),glm.probs, cex=.2, col="green")

# Calculate confusion matrix
glm.pred=rep("No", nrow(Default))
glm.probs=predict(glm.fit2,type="response")
glm.pred[glm.probs > .5]="Yes"
table(glm.pred,Default$default)
```



# Exercise

- a. Change the threshold to predict default from .5 to .4, .3, .2, .1  
Analyze the results in prediction accuracy.

# Lab - Blood Screening Data

## Description

The erythrocyte sedimentation rate and measurements of two plasma proteins (fibrinogen and globulin).

## Format

A data frame with 32 observations on the following 3 variables.

fibrinogen the fibrinogen level in the blood.

globulin the globulin level in the blood.

ESR the erythrocyte sedimentation rate, either less or greater 20 mm / hour.

## Details

The erythrocyte sedimentation rate (ESR) is the rate at which red blood cells (erythrocytes) settle out of suspension in blood plasma, when measured under standard conditions. If the ESR increases when the level of certain proteins in the blood plasma rise in association with conditions such as rheumatic diseases, chronic infections and malignant diseases, its determination might be useful in screening blood samples taken from people suspected to being suffering from one of the conditions mentioned. The absolute value of the ESR is not of great importance rather it is whether it is less than 20mm/hr since lower values indicate a healthy individual.

The question of interest is whether there is any association between the probability of an ESR reading greater than 20mm/hr and the levels of the two plasma proteins. If there is not then the determination of ESR would not be useful for diagnostic purposes.

## Source

<https://cran.r-project.org/web/packages/HSAUR>

# Lab

```
#install.packages("HSAUR")

data("plasma", package = "HSAUR")
plasma_glm_1 <- glm(ESR ~ fibrinogen, data = plasma,
                  family = binomial())
layout(matrix(1:2, ncol = 2))

head(plasma)

par(mfrow=c(1,2))

cdplot(ESR ~ fibrinogen, data = plasma)
cdplot(ESR ~ globulin, data = plasma)

par(mfrow=c(1,1))

plasma_glm_1 <- glm(ESR ~ fibrinogen, data = plasma,
                  family = binomial())

plasma_glm_2 <- glm(ESR ~ fibrinogen + globulin, data = plasma,
                  family = binomial())

summary(plasma_glm_1)
summary(plasma_glm_2)

anova(plasma_glm_1, test = "Chisq")

anova(plasma_glm_2, test = "Chisq")

prob <- predict(plasma_glm_1, type = "response")

glm.pred=rep("ESR < 20p", nrow(plasma))
glm.pred[prob > .4]="ESR > 20p"

table(glm.pred, plasma$ESR)
```

# Lab

```
prob <- predict(plasma_glm_2, type = "response")

glm.pred=rep("ESR < 20p", nrow(plasma))
glm.pred[prob >.4]="ESR > 20p"

table(glm.pred, plasma$ESR)

threshold=seq(.05, .9, .01)
n=length(threshold)
TP = rep(n,0.0)
FP = rep(n,0.0)

plasma_glm_2 <- glm(ESR ~ fibrinogen + globulin, data = plasma,
                    family = binomial())

# draw ROC curve

prob <- predict(plasma_glm_2, type = "response")

for(i in 1:n){
  glm.pred=rep("ESR < 20p", nrow(plasma))
  glm.pred[prob >threshold[i]]="ESR > 20p"
  conf_tab <- table(glm.pred, plasma$ESR)
  FP[i]= conf_tab[2,1] / sum(conf_tab[,1])
  TP[i]= conf_tab[2,2] / sum(conf_tab[,2])
}

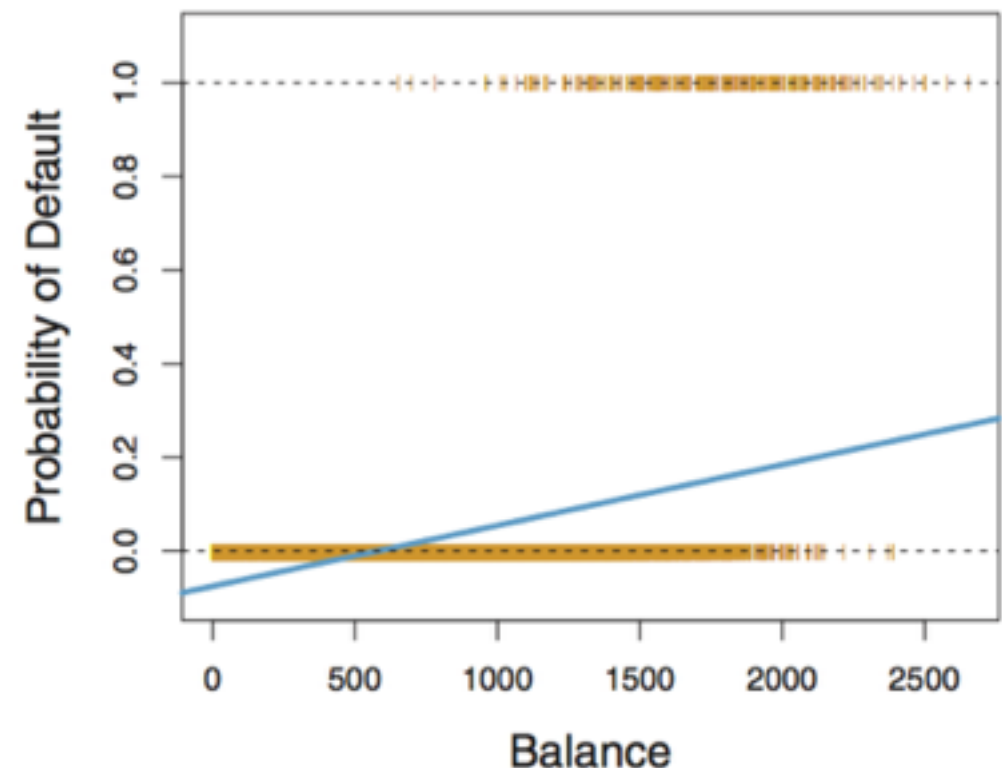
plot(FP,TP,type="l", xlab="False Positive", ylab="True Positive")
```

# Exercise

- a. Use `rpart` to build a classification tree. Compare the accuracy of the training data fit with logistic classification

# Exercise - Simple Linear Model

- a. Use Default data to fit a linear regression model (Predict Probability to Default based on balance)
- b. Draw probability estimate



# References

1. An Introduction to Statistical Learning  
with Applications in R

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani