

# Linear regression



**Carl Freidrich Gauss**



**Andrey Markov**

*All models are wrong  
but some are useful*



George E.P. Box

- Regression analysis is used to:
  - Predict the value of a dependent variable based on the value of at least one independent variable
  - Explain the impact of changes in an independent variable on the dependent variable

**Dependent variable:** the variable we wish to predict or explain

**Independent variable:** the variable used to explain the dependent variable

*response =  $f(explanatory)$  + noise*

Y (response)- dependent variable

X (explanatory)– independent variable

## Simple Linear regression

*response = intercept + (slope \* explanatory) + noise*

$$Y = \beta_0 + \beta_1 * X + \epsilon \quad \epsilon \sim N(0, \sigma_\epsilon)$$

## Multiple linear regression

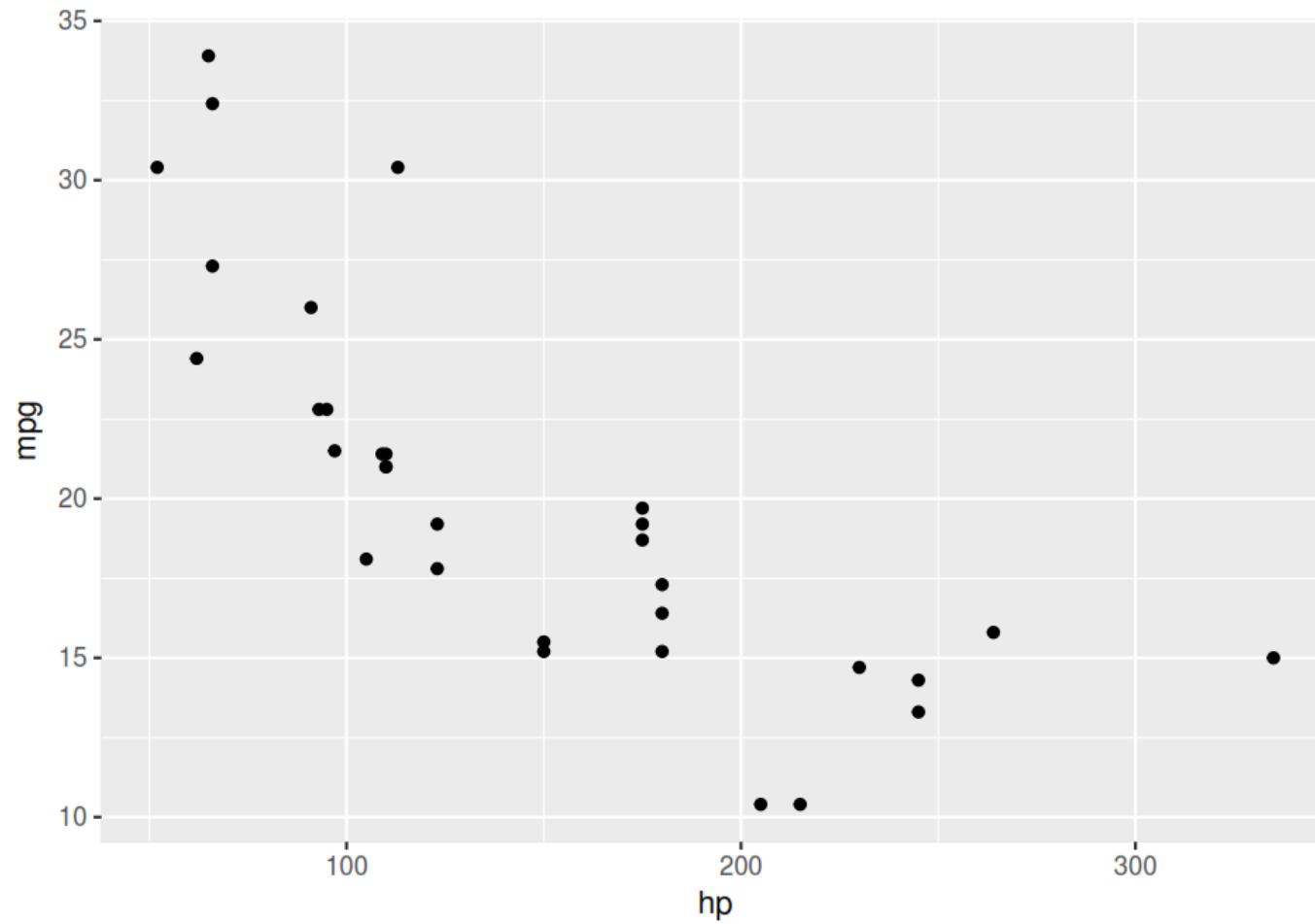
$$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \beta_3 * X_3 + \dots \epsilon$$

The goal of the modeling is to find optimal estimates for  $\beta_0, \beta_1 \dots, \beta_n$

$$\hat{Y} = \widehat{\beta_0} + \widehat{\beta_1} * X_1 + \widehat{\beta_2} * X_2 + \widehat{\beta_3} * X_3$$

# Scatterplot

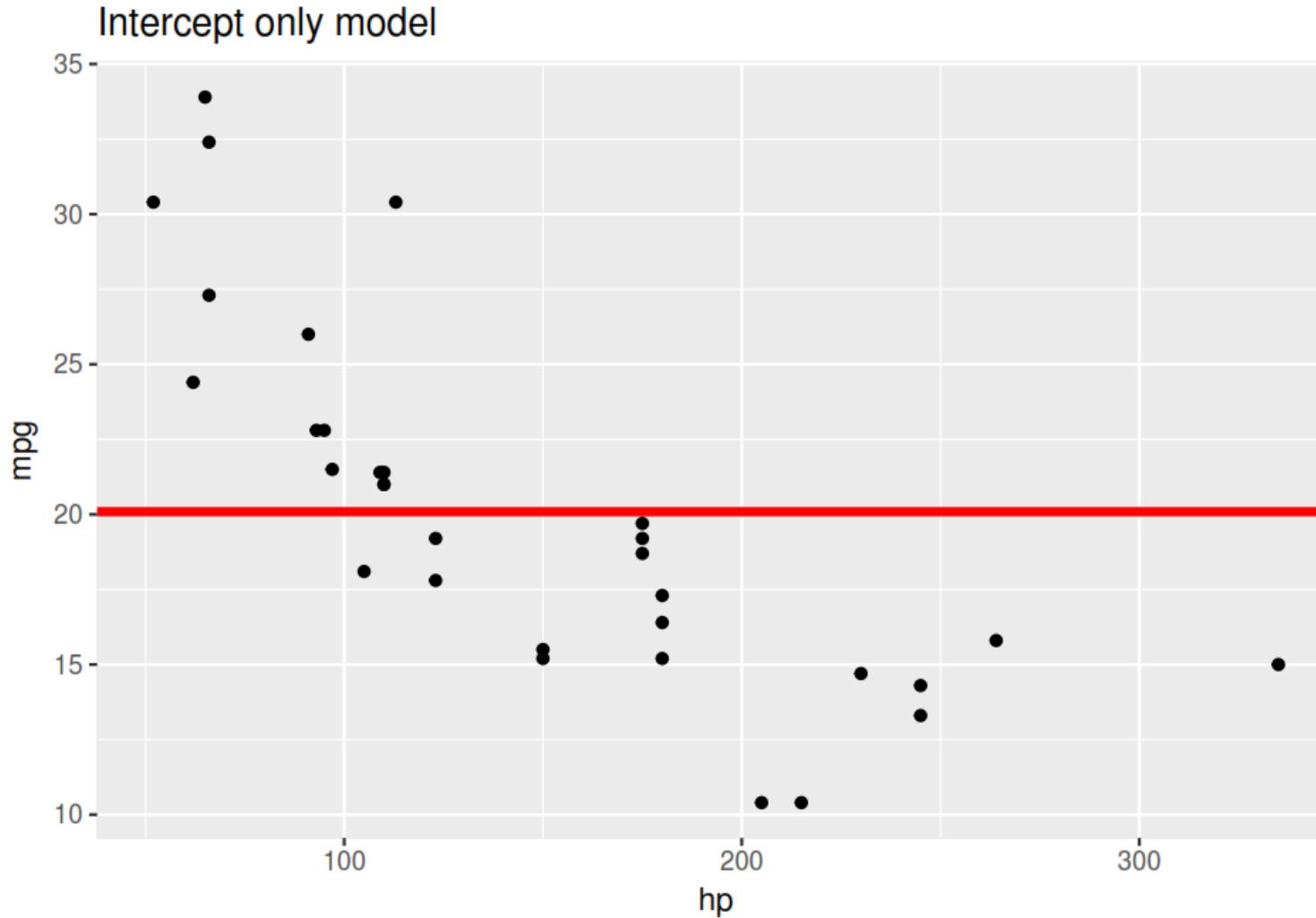
```
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point()
```



# Intercept only model

Intercept only model

```
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point() +  
  geom_hline(yintercept = mean(mtcars$mpg), col="red", size=1.5) +  
  ggtitle("Intercept only model")
```



- for linear regression we will use lm() function.
- The formula in R is defined with ~
- For regression
  - on the left side of ~ is your dependent variable,
  - on the right side of ~ are your independent variables

## Intercept only model

```
model<-lm(mpg~1, data=mtcars)
summary(model)

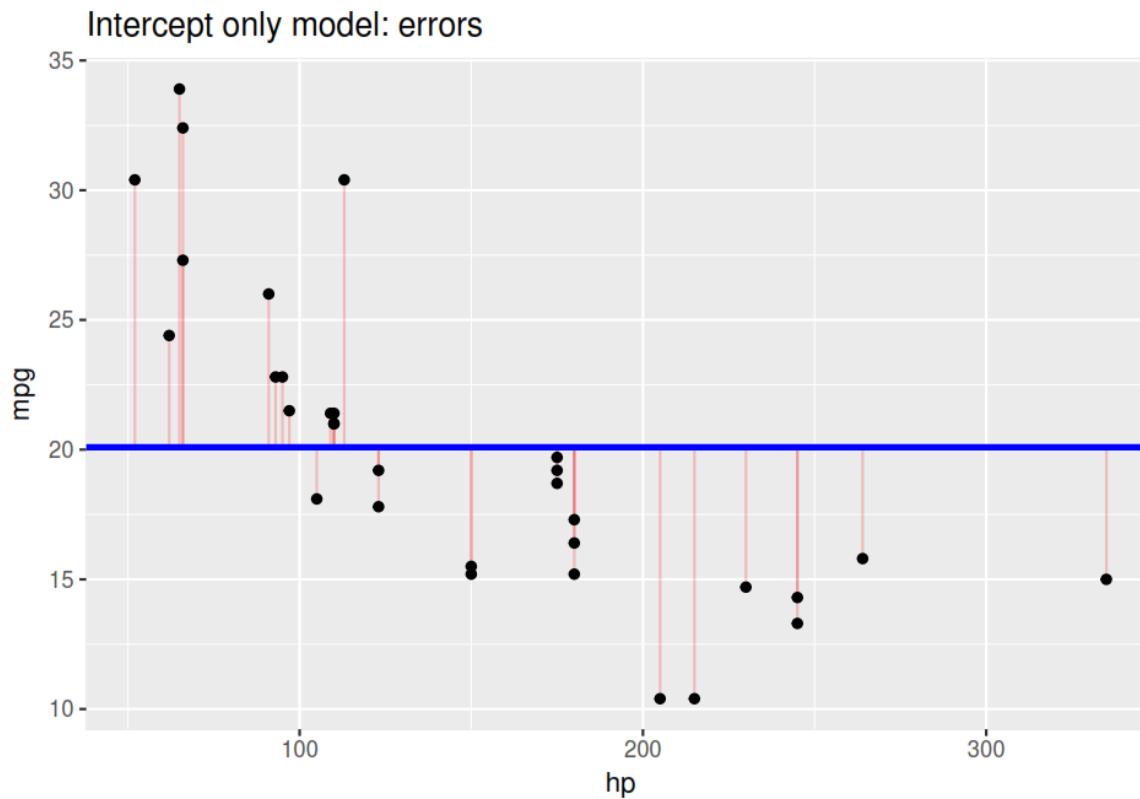
##
## Call:
## lm(formula = mpg ~ 1, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -9.6906 -4.6656 -0.8906  2.7094 13.8094 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 20.091     1.065   18.86 <2e-16 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 6.027 on 31 degrees of freedom
mean(mtcars$mpg)

## [1] 20.09062
```

# Intercept only model: errors

Errors with Intercept only model

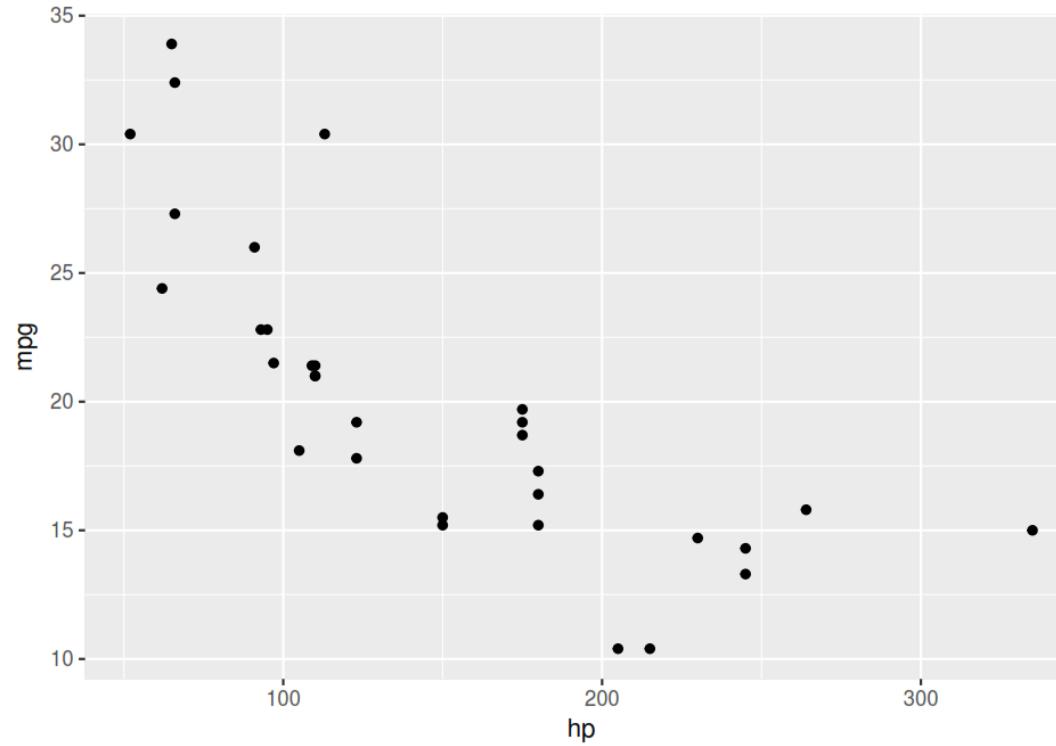
```
ggplot(mtcars, aes(x = hp, y = mpg)) +  
  geom_segment(aes(xend = hp, yend = mean(mtcars$mpg)), alpha = .2, col="red") +  
  geom_point() +  
  geom_hline(yintercept=mean(mtcars$mpg), col="blue", size=1.2) +  
  ggtitle("Intercept only model: errors")
```



# Try to find the best line

```
ggplot(mtcars, aes(x=hp, y=mpg))+geom_point()+
  geom_abline(intercept=, slope=)
```

```
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x=hp, y=mpg))+geom_point()
```



# Regression

```
mod3<-lm(mpg~hp, data=mtcars)
names(mod3)

## [1] "coefficients"   "residuals"      "effects"        "rank"
## [5] "fitted.values" "assign"         "qr"             "df.residual"
## [9] "xlevels"        "call"          "terms"          "model"
```

# Regression

```
summary(mod3)

##
## Call:
## lm(formula = mpg ~ hp, data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.7121 -2.1122 -0.8854  1.5819  8.2360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.09886   1.63392 18.421 < 2e-16 ***
## hp          -0.06823   0.01012 -6.742 1.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.863 on 30 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5892
## F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07

coefficients(mod3)

## (Intercept)      hp
## 30.09886054 -0.06822828
```

## The regression formula

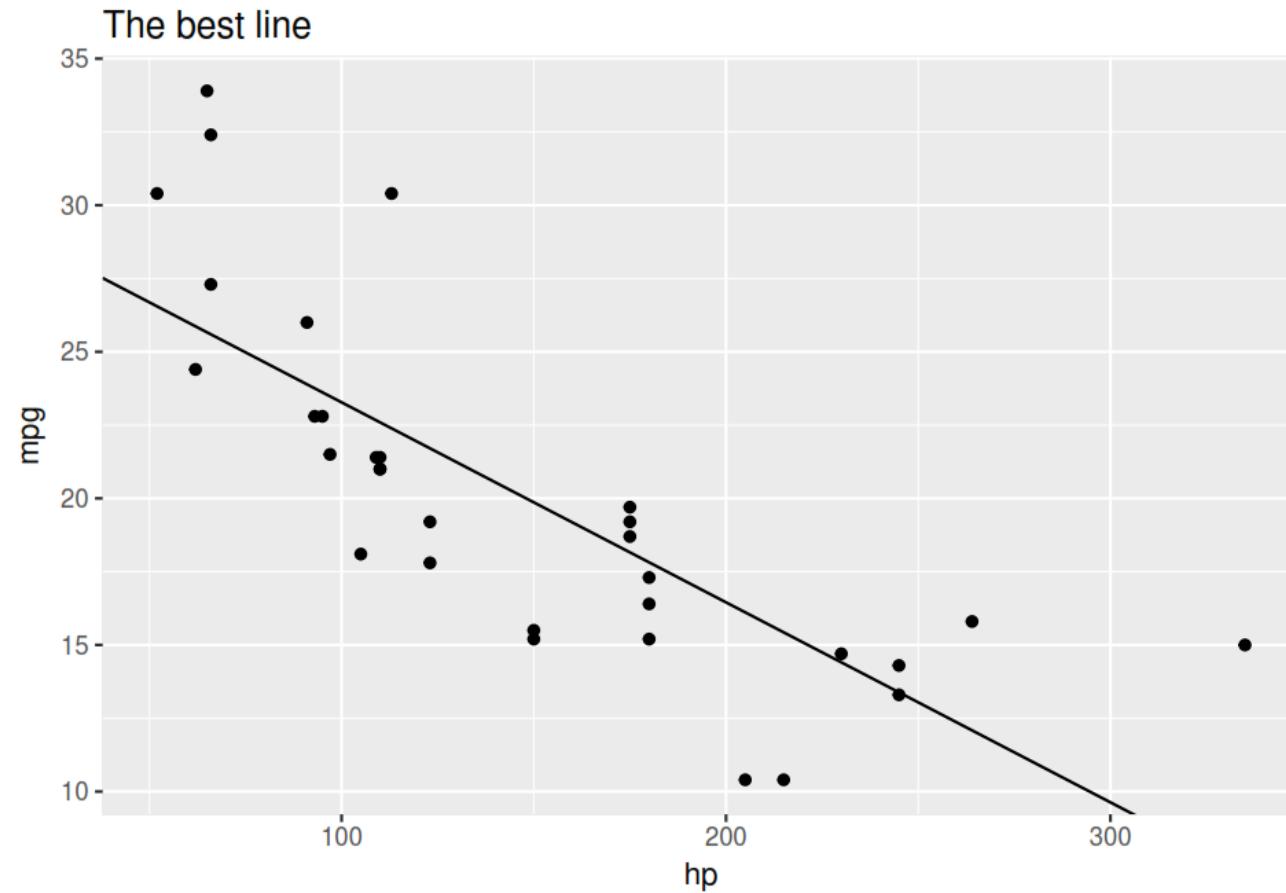
```
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 30.09886   1.63392 18.421 < 2e-16 ***  
## hp          -0.06823   0.01012 -6.742 1.79e-07 ***
```

$$mpg = 30.9886 + (-0.06823) * hp$$

# The best line

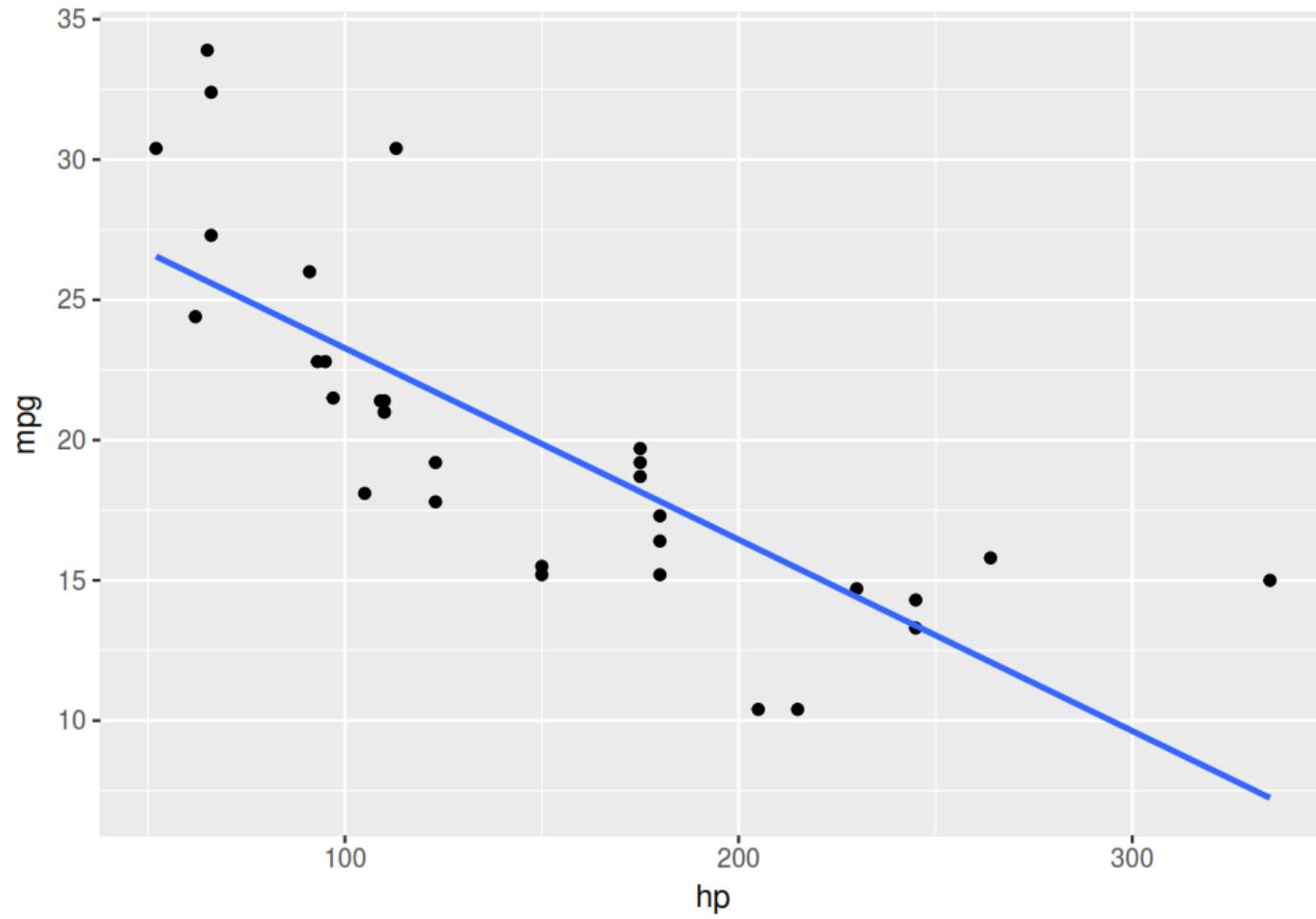
The best line

```
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point() +  
  geom_abline(intercept=30.09886054, slope=-0.06822828) +  
  ggtitle("The best line")
```



## The best fit

```
ggplot(mtcars, aes(x=hp, y=mpg)) + geom_point() +  
  geom_smooth(method="lm", se=F)
```



## The fitting process

$$Y = \beta_0 + \beta_1 * X + \epsilon$$

The goal of the modeling is to find optimal estimates for  $\beta_0$ ,  $\beta_1$

$$\hat{Y} = \widehat{\beta}_0 + \widehat{\beta}_1 * X$$

$\hat{Y}$  is called fitted or predicted value

## Residual

$$e = Y - \hat{Y}$$

## Fitting procedure

- Given  $n$  observations of pairs  $(x_i, y_i)$
- Find  $\widehat{\beta}_0, \widehat{\beta}_1$  that minimize  $\sum_{i=1}^n e_i^2$

## Least Squares method

- Easy, unique solution
- Residuals sums to zero
- Line must pass through  $(\bar{X}, \bar{Y})$
- There are several other assumptions as well

- **Linearity**
  - The underlying relationship between X and Y is linear
- **Independence of Errors**
  - Error values are statistically independent
- **Normality of Error**
  - Error values ( $\varepsilon$ ) are normally distributed for any given value of X
- **Equal Variance (Homoscedasticity)**
  - The probability distribution of the errors has constant variance

## How good is the model ?

- Inference about the slope: Is there a significant linear relationship between independent and dependent variables?
- How good is the model in explaining the relationship ?

## Is there a significant linear relationship

- t test for a population slope
  - Is there a linear relationship between X and Y?
- Null and alternative hypotheses
  - $H_0: \beta_1 = 0$  (no linear relationship)
  - $H_1: \beta_1 \neq 0$  (linear relationship does exist)

Testing: look at the p-value. If it is small (less than 0.05) then the relationship is significant

## Hypothesis testing for the slope

```
...  
## Coefficients:  
##             Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 30.09886   1.63392 18.421 < 2e-16 ***  
## hp          -0.06823   0.01012 -6.742 1.79e-07 ***  
## ---
```

2e-16 is a scientific notation

This means  $2 * 10^{-16} = \frac{2}{10^{16}}$

Very very small number

Total variation is made up of two parts:

$$\text{SST} = \text{SSR} + \text{SSE}$$

Total Sum of Squares

Regression Sum of Squares

Error Sum of Squares

$$SST = \sum (Y_i - \bar{Y})^2$$

$$SSR = \sum (\hat{Y}_i - \bar{Y})^2$$

$$SSE = \sum (Y_i - \hat{Y}_i)^2$$

where:

$\bar{Y}$  = Average value of the dependent variable

$Y_i$  = Observed values of the dependent variable

$\hat{Y}_i$  = Predicted value of Y for the given  $X_i$  value

## How good is the model ?

- $SST = \text{total sum of squares}$ 
  - Measures the variation of the  $Y_i$  values around their mean  $\bar{Y}$
- $SSR = \text{regression sum of squares}$ 
  - Explained variation attributable to the relationship between  $X$  and  $Y$
- $SSE = \text{error sum of squares}$ 
  - Variation attributable to factors other than the relationship between  $X$  and  $Y$

## R square

$$SSR = \sum_{i=1}^n (\hat{y} - \bar{y})^2$$

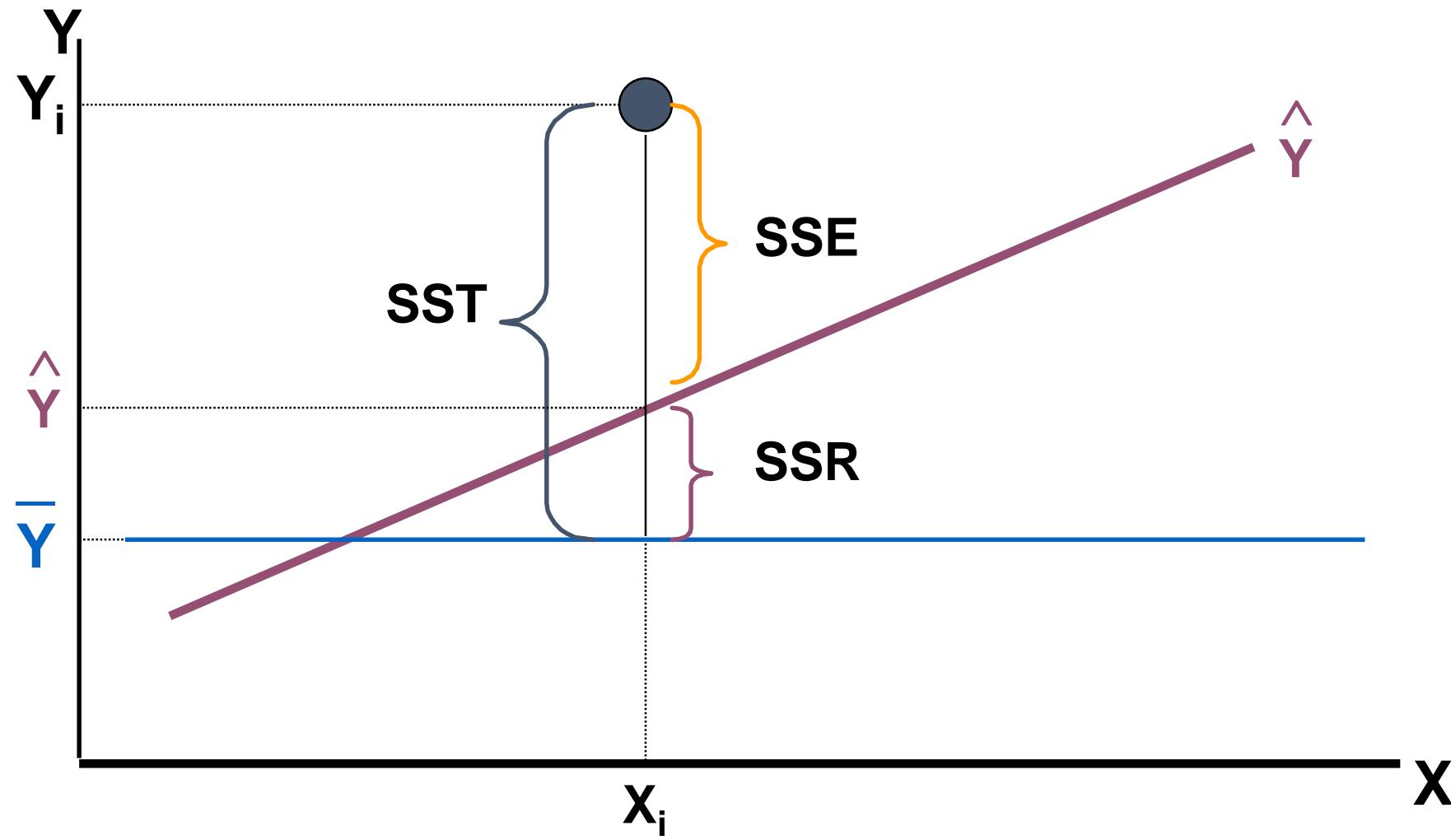
$$SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$SST = \sum_{i=1}^n (y_i - \bar{y})^2$$

$$R^2 = \frac{SSR}{SST}$$

$$R^2 = 1 - \frac{SSE}{SST}$$

## Measures of Variation



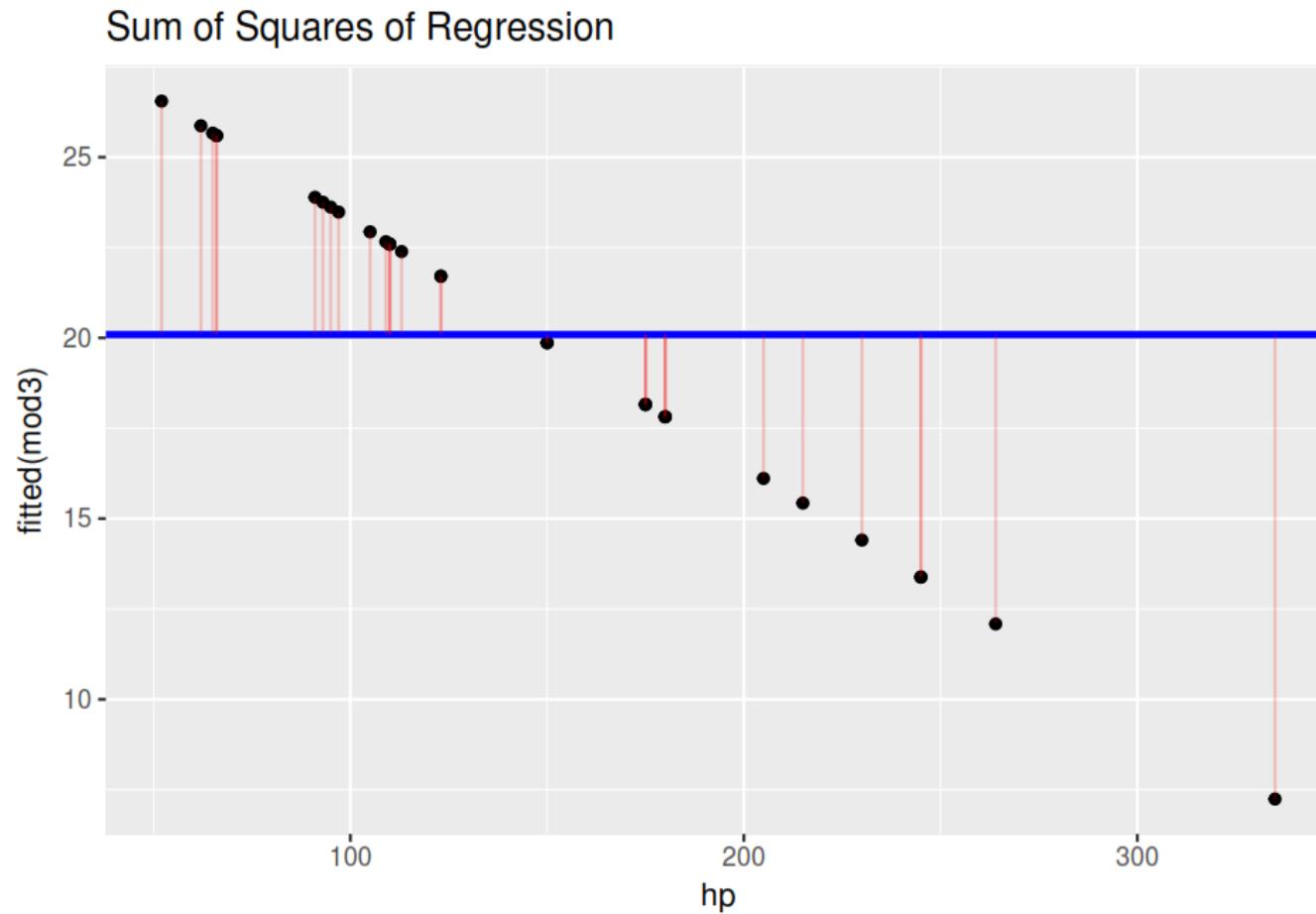
# Sum of Squares of Total (intercept only model)

```
ggplot(mtcars, aes(x = hp, y = mpg)) +  
  geom_segment(aes(xend = hp, yend = mean(mtcars$mpg)), alpha = .2, col="red") +  
  geom_point() +  
  geom_hline(yintercept=mean(mtcars$mpg), col="blue", size=1.2) +  
  ggtitle("Intercept only model: errors")
```



# Sum of Squares of Regression

```
ggplot(mtcars, aes(x = hp, y = fitted(mod3))) + geom_point()+
  geom_hline(yintercept=mean(mtcars$mpg), col="blue", size=1.2) +
  geom_segment(aes(xend = hp, yend = mean(mtcars$mpg)), alpha = .2, col="red") +
  ggtitle("Sum of Squares of Regression")
```



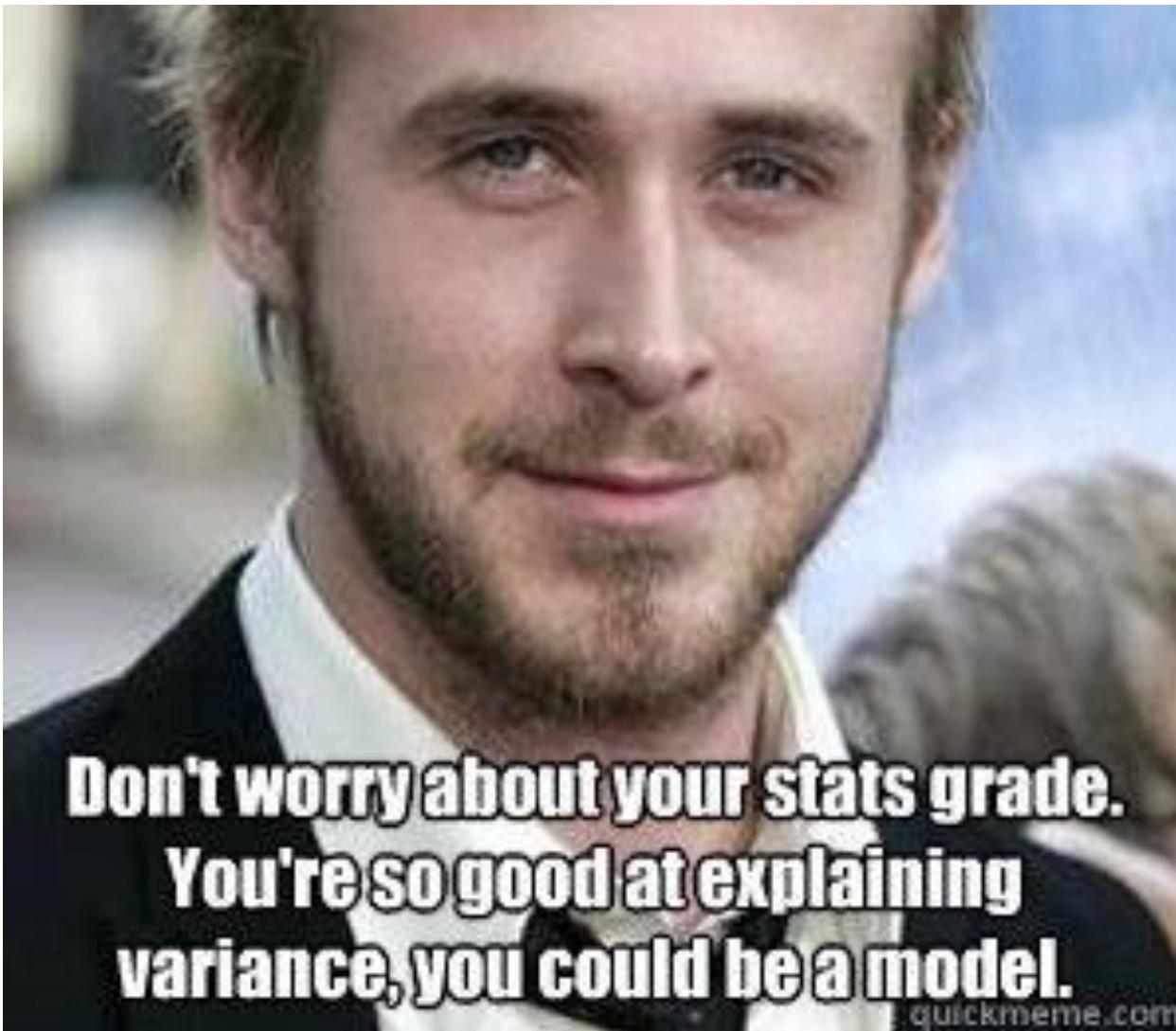
# Sum of Squares of Errors

```
ggplot(mtcars, aes(x = hp, y = mpg)) +  
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  
  geom_segment(aes(xend = hp, yend = fitted(mod3)), alpha = .2, col="red") +  
  geom_point() + geom_point(aes(y = fitted(mod3)), shape = 1) +  
  ggtitle("Sum of Squares of errors")
```



$$0 \leq R^2 \leq 1$$

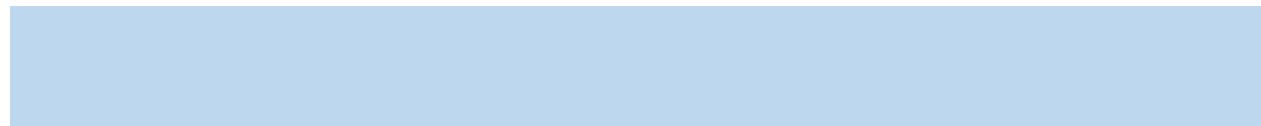
- $R_{square}$  The percentage of variation in Y (dependent variable) that is explained by the X (independent variable).
- Closer to 1, better is the model
- Is showing how good is the model doing compared to baseline (intercept only model).
- For the simple linear regression model (1 independent variable) is the square of correlation coefficient.



**Don't worry about your stats grade.  
You're so good at explaining  
variance, you could be a model.**

quickmeme.com

# Multiple regression model



## Multiple regression

$$Y = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \beta_3 * X_3 + \dots \epsilon$$

The goal of the modeling is to find optimal estimates for  $\beta_0, \beta_1 \dots, \beta_n$

$$\hat{Y} = \widehat{\beta_0} + \widehat{\beta_1} * X_1 + \widehat{\beta_2} * X_2 + \widehat{\beta_3} * X_3$$

```
serieia <- read.csv("serieia.csv")
str(serieia)

## 'data.frame': 8775 obs. of 15 variables:
## $ POS      : Factor w/ 3 levels "D","F","M": 1 1 3 3 3 1 1 1 3 3 ...
## $ Name     : Factor w/ 2817 levels "Aaron Mattia Tabacchi",...: 1753 946 1859 853 455 1125 26...
## $ Age      : int 27 29 23 20 18 26 31 25 26 23 ...
## $ APP      : int 32 34 20 1 30 25 8 27 18 27 ...
## $ SUBIN    : int 0 0 6 1 9 2 1 2 1 6 ...
## $ G         : int 2 6 3 0 1 0 0 0 0 2 ...
## $ A         : int 0 0 0 0 0 0 0 0 0 0 ...
## $ SH        : int 0 0 0 0 0 0 0 0 0 0 ...
## $ ST        : int 0 0 0 0 0 0 0 0 0 0 ...
## $ FC        : int 0 0 0 0 0 0 0 0 0 0 ...
## $ FA        : int 0 0 0 0 0 0 0 0 0 0 ...
## $ YC        : int 8 1 1 0 7 2 0 8 2 5 ...
## $ RC        : int 0 0 0 0 1 0 0 0 0 0 ...
## $ team_id   : int 116 116 116 116 116 117 117 117 117 117 ...
## $ Season    : int 2002 2002 2002 2002 2002 2002 2002 2002 2002 2002 ...
```

## Variables:

POS: Players position: D-Defender, F-Forward, M-Midfielder

Name: Players name

Age: Player age at that season

APP: Number of appearances

SUBIN: Substitute Appearances

G: Goals

A: Assists

SH: Shots

ST: Shots on target

FC: Fouls committed

FA: Fouls Suffered

YC: Yellow cards

RC: red cards

- First we run regression model with G – number of goals as dependent variable and POS position of the player as an independent variable
- Note that POS is a categorical variable

```

model1 <- lm(G~POS, data=seriea)
summary(model1)

##
## Call:
## lm(formula = G ~ POS, data = seriea)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.863 -1.324 -0.527  0.473 32.137
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.52731   0.05395  9.775  <2e-16 ***
## POSF        3.33551   0.08654 38.543  <2e-16 ***
## POSM        0.79699   0.07351 10.842  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.019 on 8772 degrees of freedom
## Multiple R-squared:  0.1497, Adjusted R-squared:  0.1495
## F-statistic: 771.9 on 2 and 8772 DF,  p-value: < 2.2e-16

```

- POS is a categorical variable with 3 levels, D,F,M
- D is the reference level, as it comes alphabetically first
- POSF is the coefficient for F, POSM is the coefficient for M

```
coef(model1)
```

```
## (Intercept)      POSF      POSM
## 0.5273076    3.3355065   0.7969946
```

- Interpretations:
  - The average number of goals for Forwards is by 3.3355 more than the average for **Defenders**.
  - The average number of goals for Midfielders is by 0.79699 more than the average for **Defenders**.

- Calculate the average number of the goals for Forwards, Midfielders and Defenders
- You can see that the average for Defenders is the same as our intercept

```
F <- mean(seriea$G[seriea$POS=='F'])  
D <- mean(seriea$G[seriea$POS=='D'])  
M <- mean(seriea$G[seriea$POS=='M'])
```

```
F
```

```
## [1] 3.862814
```

```
D
```

```
## [1] 0.5273076
```

```
M
```

```
## [1] 1.324302
```

F

```
## [1] 3.862814
```

D

```
## [1] 0.5273076
```

M

```
## [1] 1.324302
```

```
coef(model1)
```

```
## (Intercept)      POSF      POSM  
## 0.5273076  3.3355065  0.7969946
```

POSF = 3.335

Mean(Forwards) =  $0.5273076 + 3.3355065 = 3.862814$

- The resulting object from `lm()` function is a list
- It contains different information about our model and can be assessed by names as well

```
typeof(model1)

## [1] "list"

names(model1)

##  [1] "coefficients"    "residuals"        "effects"          "rank"
##  [5] "fitted.values"   "assign"           "qr"               "df.residual"
##  [9] "contrasts"       "xlevels"          "call"              "terms"
## [13] "model"
```

```
model1$coefficients

## (Intercept)      POSF      POSM
## 0.5273076  3.3355065  0.7969946

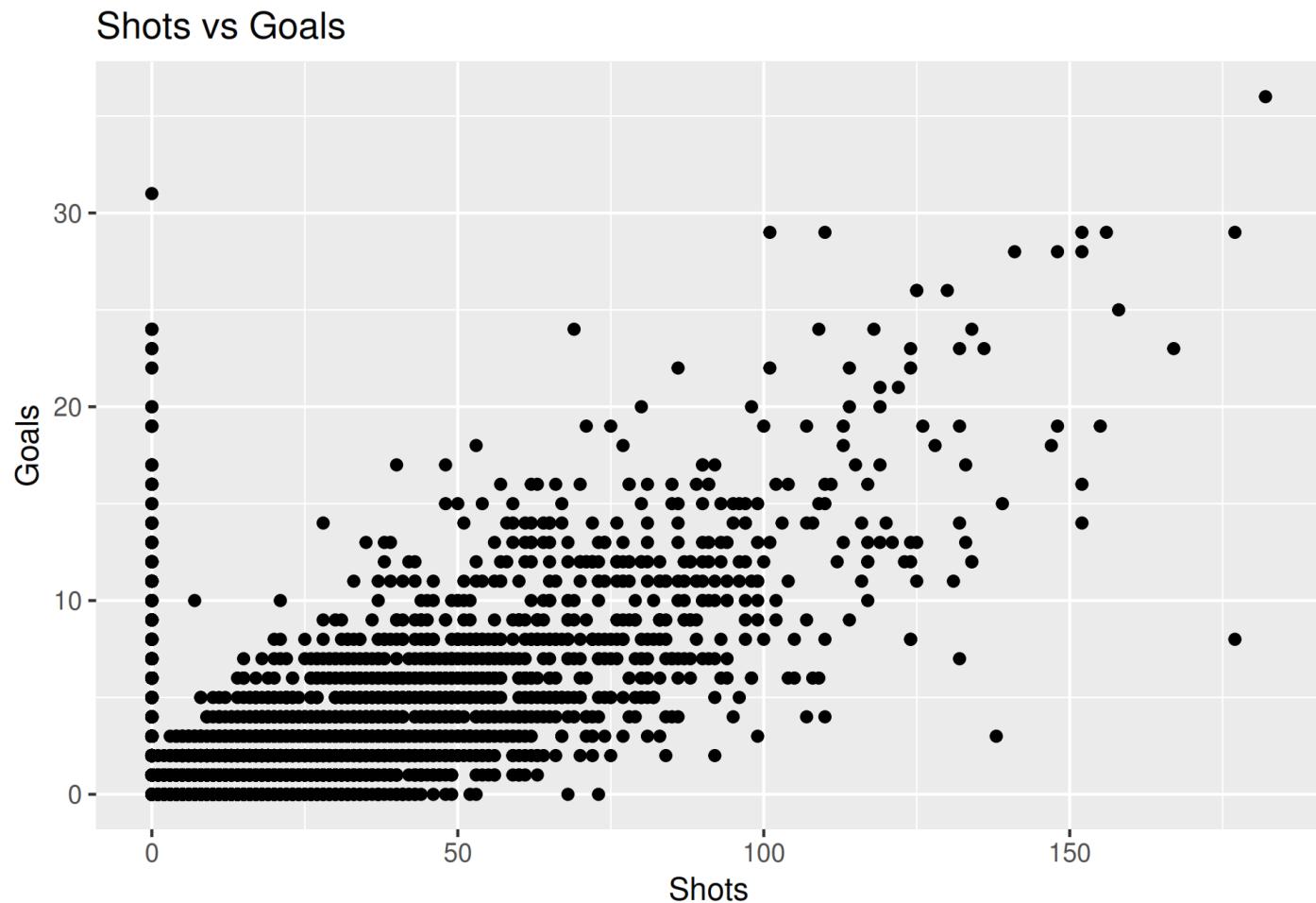
model1$residuals[1:20]

##          1          2          3          4          5          6
## 1.4726924 5.4726924 1.6756979 -1.3243021 -0.3243021 -0.5273076
##          7          8          9         10         11         12
## -0.5273076 -0.5273076 -1.3243021  0.6756979 -1.3243021  2.6756979
##          13         14         15         16         17         18
## 5.6756979 -1.3243021 3.1371859 -0.5273076 -0.5273076  0.4726924
##          19         20
## 0.4726924  0.4726924
```

Now lets add more variables to our model

Variable SHOTS on goal, fist lets visualize the data, do you think the relationship is linear? anything strange ?

```
ggplot(seriea, aes(x=SH, y=G))+ geom_point()+
  labs(x="Shots", y="Goals", title="Shots vs Goals")
```



## Lets run the model

```
model2 <- lm(G~POS+SH, data=seriea)
summary(model2)

##
## Call:
## lm(formula = G ~ POS + SH, data = seriea)
##
## Residuals:
##       Min     1Q Median     3Q    Max 
## -12.4015 -0.9138 -0.0026  0.0977 29.4301 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -0.003864   0.039370  -0.098   0.9218    
## POSF         1.573751   0.065446   24.047 <2e-16 ***  
## POSM        -0.093801   0.053959  -1.738   0.0822    
## SH           0.100229   0.001115  89.881 <2e-16 ***  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 2.178 on 8771 degrees of freedom
## Multiple R-squared:  0.5574, Adjusted R-squared:  0.5572 
## F-statistic: 3681 on 3 and 8771 DF,  p-value: < 2.2e-16
```

- How good is the model ?

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.178 on 8771 degrees of freedom
## Multiple R-squared:  0.5574, Adjusted R-squared:  0.5572
## F-statistic:  3681 on 3 and 8771 DF,  p-value: < 2.2e-16
```

~56% of the variance explained

- Are the variables significant ?

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.003864  0.039370 -0.098  0.9218
## POSF         1.573751  0.065446 24.047 <2e-16 ***
## POSM        -0.093801  0.053959 -1.738  0.0822 .
## SH          0.100229  0.001115 89.881 <2e-16 ***
## ---
```

Look at the p-values

- SH is significant ( $\alpha = 0.05$ )
- Categorical variable is treated as significant if at least one of its categories is significant

## Interpret the coefficient for SH (SHOT)

```
###  
## Coefficients:  
##             Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -0.003864  0.039370 -0.098  0.9218  
## POSF         1.573751  0.065446 24.047 <2e-16 ***  
## POSM        -0.093801  0.053959 -1.738  0.0822 .  
## SH           0.100229  0.001115 89.881 <2e-16 ***  
## ---
```

## Regression equation

coefficients(model2)

```
##   (Intercept)      POSF      POSM       SH  
## -0.003864495  1.573751402 -0.093801396  0.100228996
```

$$G = \hat{\beta}_0 + \hat{\beta}_1 POSF + \hat{\beta}_2 POSM + \hat{\beta}_3 SH$$

$$G = -0.0038 + 1.573 * POSF - 0.0938POSM + 0.1002SH$$

Predict the number of the goals for a Forward who made 35 shots on goal

$$G = -0.0038 + 1.573 * POSF - 0.0938 * POSM + 0.1002 * SH$$

POSF=1

POSM=0

SH=35

$$G = -0.0038 + 1.573 * 1 - 0.0938 * 0 + 0.1002 * 30 = 4.57$$

Predict the number of the goals for a defender who made 20 shots on goal

$$G = -0.0038 + 1.573 * POSF - 0.0938 * POSM + 0.1002 * SH$$

$$POSF=0$$

$$POSM=0$$

$$SH=20$$

$$G = -0.0038 + 1.573 * 0 - 0.0938 * 0 + 0.1002 * 20 = 2.0002$$

Predict the number of the goals for a Forward who made 35 shots on goal

$$G = -0.0038 + 1.573 * POSF - 0.0938 * POSM + 0.1002 * SH$$

POSF=1

POSM=0

SH=35

$$G = -0.0038 + 1.573 * 1 - 0.0938 * 0 + 0.1002 * 30 = 4.57$$

Lets build another model now with ST (Shots on target).

Are the variables significant?

How good is the model ?

```
model3 <- lm(G~POS+ST, data=seriea)
summary(model3)

##
## Call:
## lm(formula = G ~ POS + ST, data = seriea)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -11.3920 -0.6871 -0.1113  0.0361 29.6404 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 0.111263   0.036359   3.060  0.00222 ** 
## POSF        1.248364   0.061358  20.346 < 2e-16 ***  
## POSM       -0.011135   0.049855  -0.223  0.82328    
## ST          0.287939   0.002774 103.815 < 2e-16 ***  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 2.022 on 8771 degrees of freedom
## Multiple R-squared:  0.6185, Adjusted R-squared:  0.6183 
## F-statistic: 4739 on 3 and 8771 DF,  p-value: < 2.2e-16
```

Add both variables SH and ST to model.

Anything strange ?

```
model4 <- lm(G~POS+ST+SH, data=seriea)
summary(model4)

##
## Call:
## lm(formula = G ~ POS + ST + SH, data = seriea)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -11.3358  -0.7131  -0.1903   0.1475  29.6211 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 0.167984  0.036621  4.587 4.56e-06 ***
## POSF        1.210879  0.061146 19.803 < 2e-16 ***
## POSM        0.048768  0.049960  0.976   0.329    
## ST          0.379793  0.009752 38.945 < 2e-16 ***
## SH         -0.035746  0.003640 -9.820 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.011 on 8770 degrees of freedom
## Multiple R-squared:  0.6226, Adjusted R-squared:  0.6224 
## F-statistic: 3617 on 4 and 8770 DF,  p-value: < 2.2e-16
```

The sign of the variable SH changes from + to – when we add variable ST to the model

```
coef(model2)
```

```
## (Intercept)      POSF      POSM      SH
## -0.003864495  1.573751402 -0.093801396  0.100228996
```

```
coef(model4)
```

```
## (Intercept)      POSF      POSM      ST      SH
##  0.16798351  1.21087861  0.04876766  0.37979301 -0.03574640
```

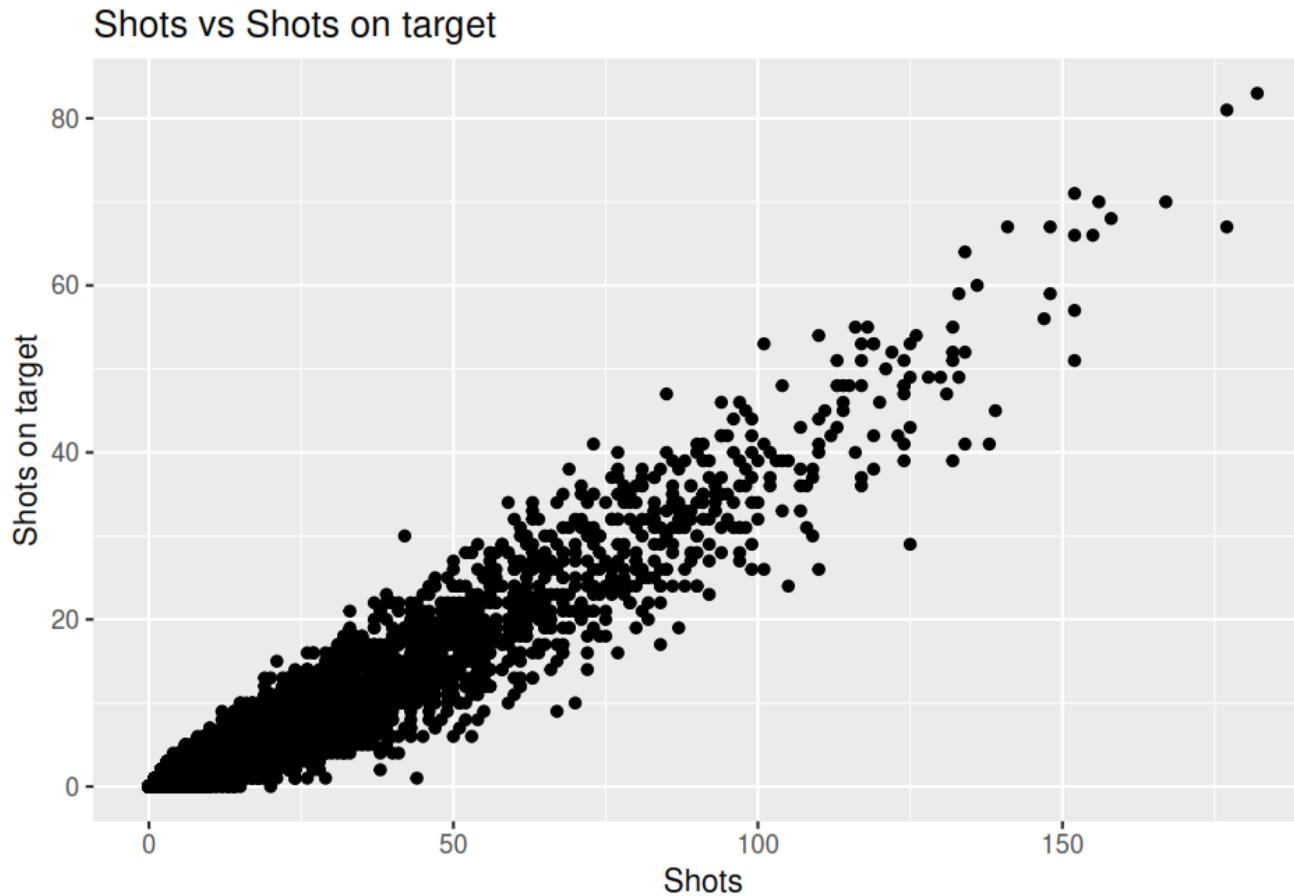
This is called a multicollinearity problem

Multicollinearity is present when independent x variables are highly correlated with each other

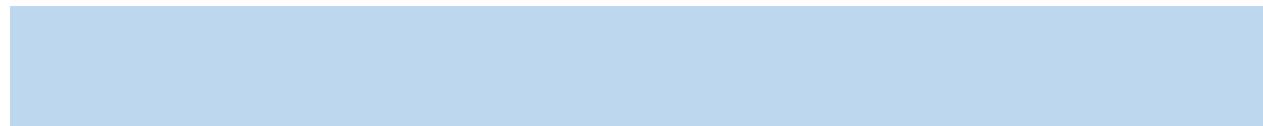


Look at the correlation coefficient first

```
cor(seriea$SH, seriea$ST)  
## [1] 0.9616804  
  
ggplot(seriea, aes(x=SH, y=ST))+geom_point() +  
  labs(x="Shots", y="Shots on target",  
    title="Shots vs Shots on target")
```



# Model Selection



# Learning process: Supervised Learning

Problem

$$x^2 - 10x + 24 = 0 \longrightarrow (6,4)$$

$$x^2 - 2x - 8 = 0 \longrightarrow (4,-2)$$

$$x^2 - 4x - 21 = 0 \longrightarrow (7,-3)$$

$$x^2 - 6x + 5 = 0 \longrightarrow (5,1)$$

$$x^2 - 5x + 4 = 0 \longrightarrow (4,1)$$

Solution

# Learning process: Unsupervised Learning

## Problem

$$x^2 - 10x + 24 = 0$$

$$x^2 - 2x - 8 = 0$$

$$x^2 - 4x - 21 = 0$$

$$x^2 - 6x + 5 = 0$$

$$x^2 - 5x + 4 = 0$$

# Learning process: How to know if one has really learned how to solve the problems?

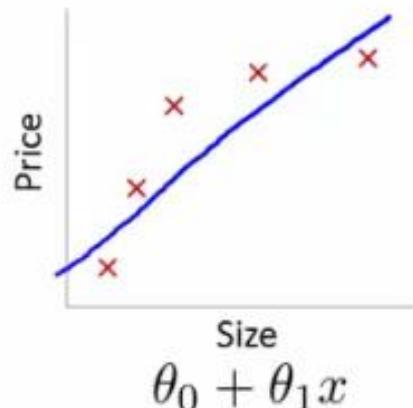
Problem	Solution
$x^2 - 10x + 24 = 0$	$(6,4)$
$x^2 - 2x - 8 = 0$	$(4,-2)$
$x^2 - 4x - 21 = 0$	$(7,-3)$
$x^2 - 6x + 5 = 0$	$(5,1)$
$x^2 - 5x + 4 = 0$	$(4,1)$

**Give another set of problems, with no solutions**

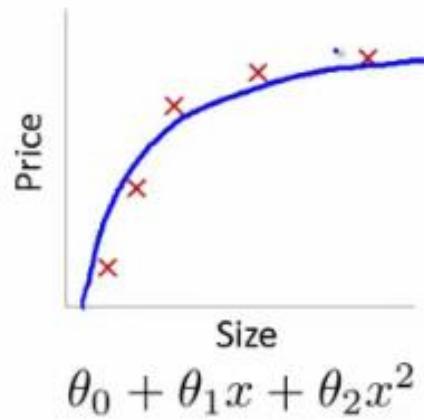
$$\text{Mean Error} = \text{Bias}^2 + \text{Variance} + \sigma^2$$

- The bias is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- The variance is error from sensitivity to small fluctuations in the training set. High variance can cause overfitting: modeling the random noise in the training data, rather than the intended outputs.

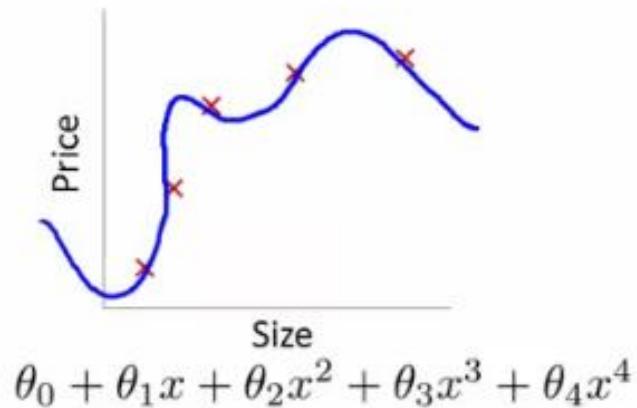
# Overfitting and Underfitting



High bias  
(underfit)

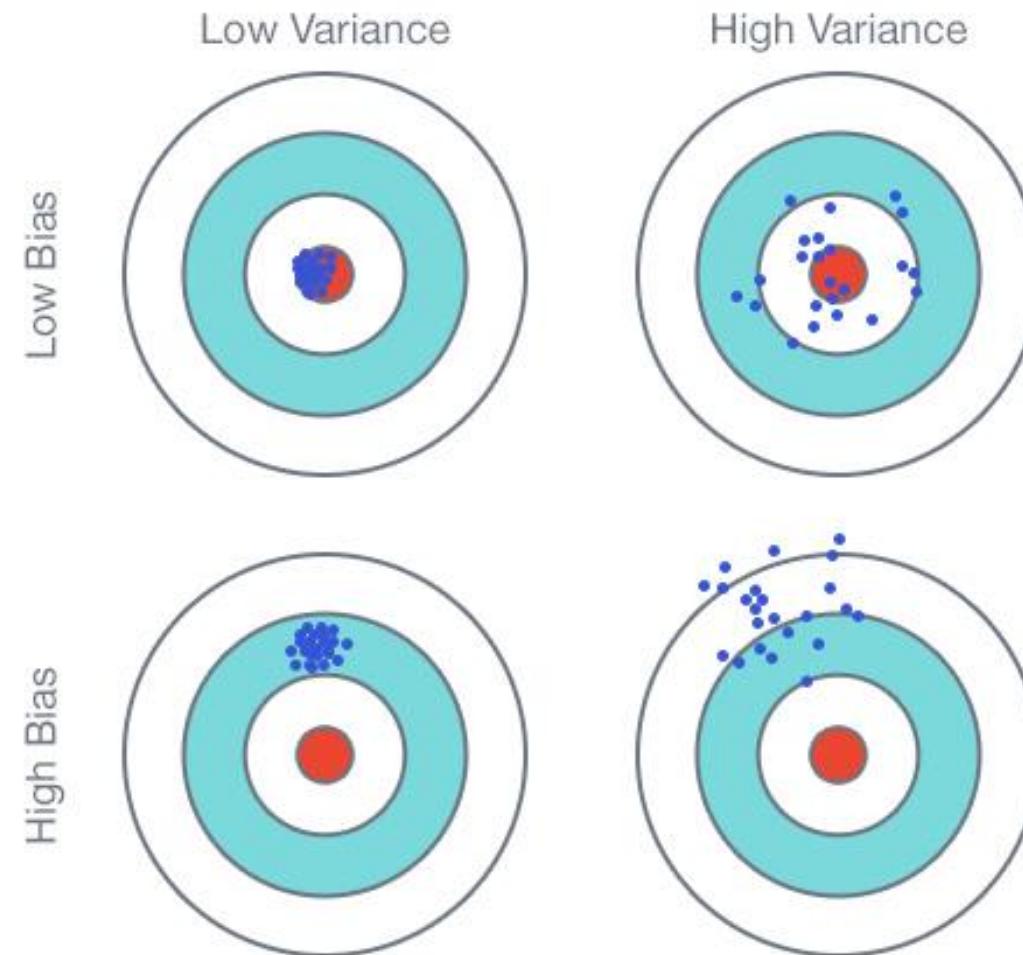


“Just right”



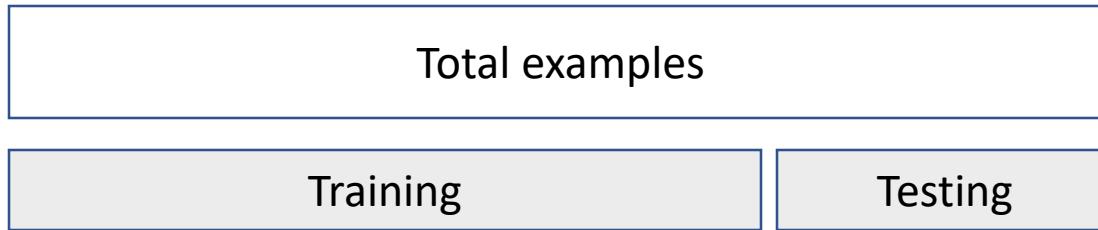
High variance  
(overfit)

# Bias vs Variance trade-off



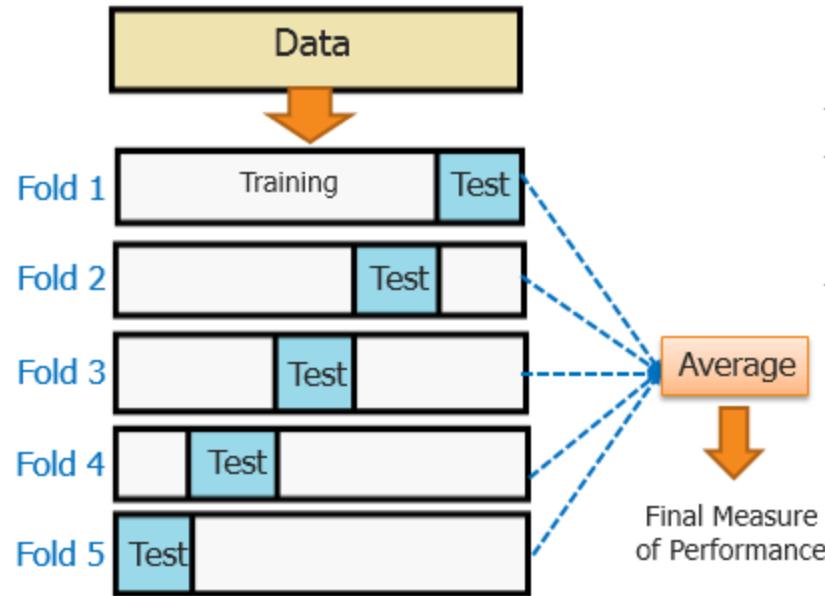
### Holdout

- Reserve 2/3 for training and 1/3 for testing



Use model evaluation metric, such as RMSE, on Testing dataset. Choose the model with the optimum evaluation metric.

# Cross Validation



- Technique to validate models/classifiers
- Method to estimate how accurately the model generalizes to unseen data i.e., how well it performs/predicts
- K-fold CV
  - » Most popular
  - » k is typically set to 10
  - » Every sample/record is used both in training and test sets

### Root Mean Square of Errors

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y} - y)^2}{n}}$$

### Mean Absolute Error

$$MAE = \frac{\sum_{i=1}^n |\hat{y} - y|}{n}$$

## Bias vs Variance tradeoff

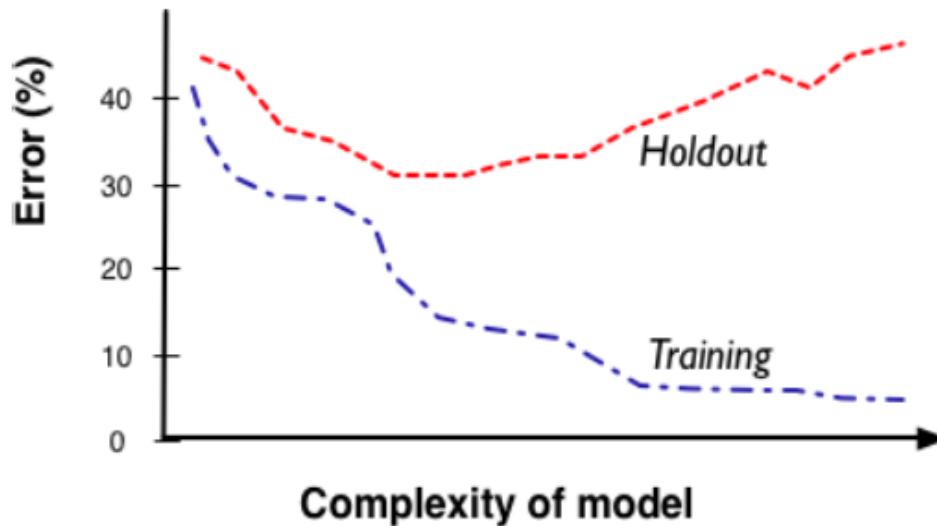
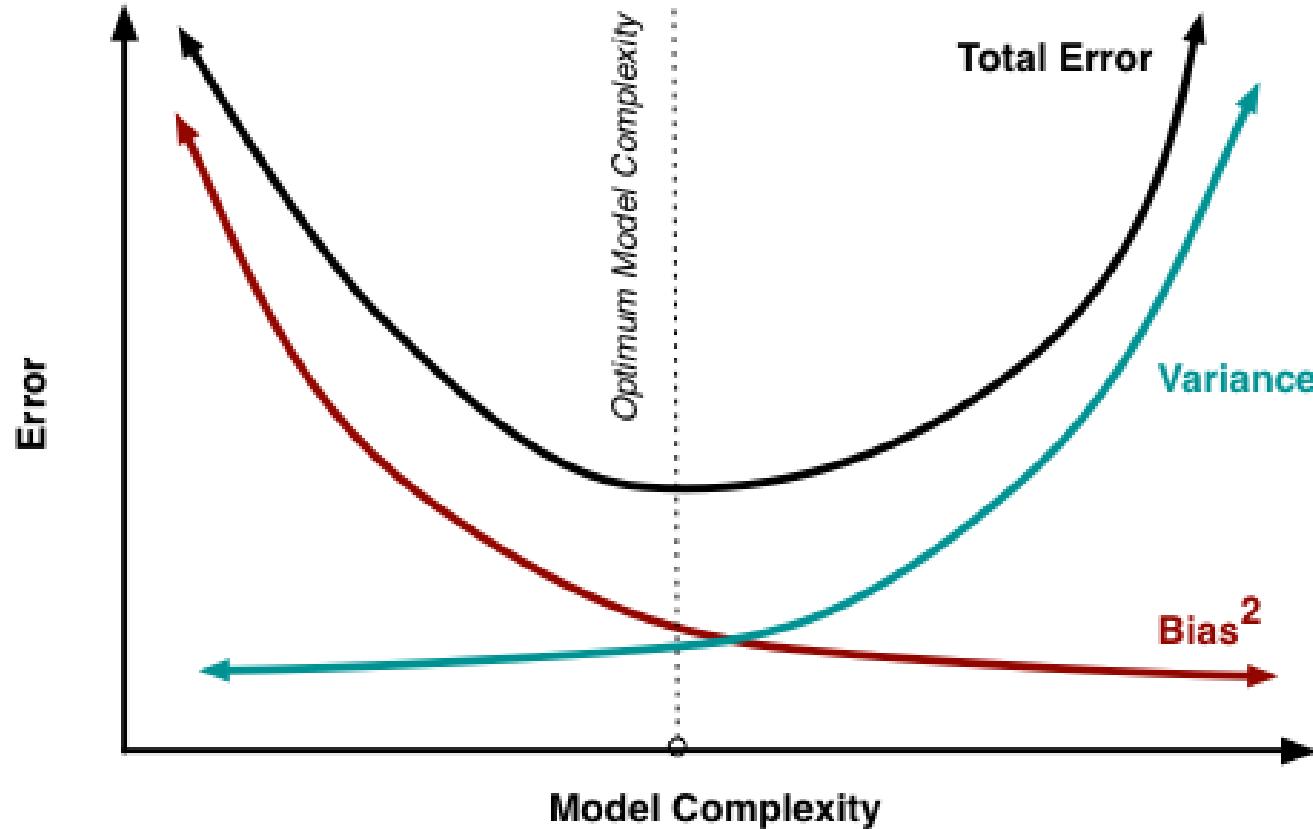
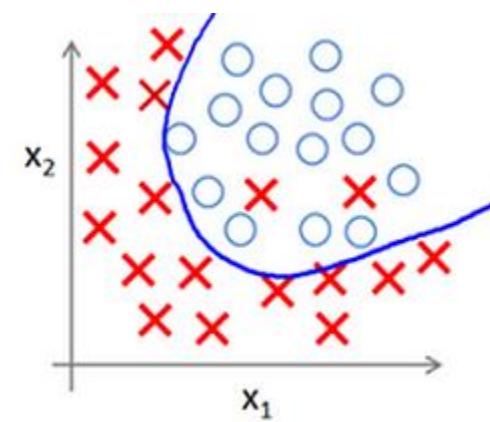
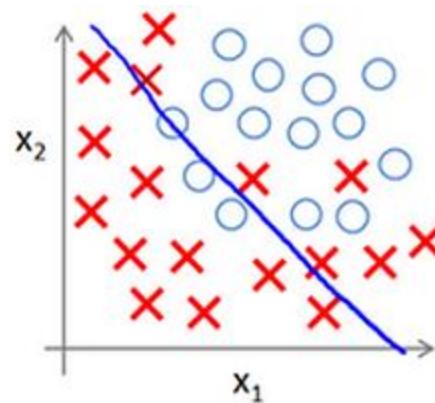
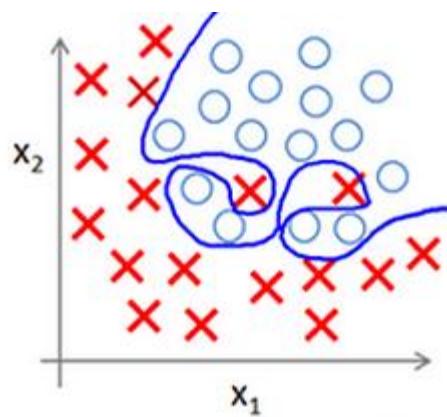


Figure 5-1. A typical fitting graph. Each point on a curve represents an accuracy estimation of a model with a specified complexity (as indicated on the horizontal axis). Accuracy estimates on training data and testing data vary differently based on how complex we allow a model to be. When the model is not allowed to be complex enough, it is not very accurate. As the models get too complex, they look very accurate on the training data, but in fact are overfitting—the training accuracy diverges from the holdout (generalization) accuracy.

## Bias-Variance tradeoff



# Where is what ?



In 1838 Charles Darwin was considering to propose to his cousin Emma Wedgwood. He listed all pros and cons of the marriage

### Pros

- Children
- companionship
- charms of music
- female chit-chat

### Cons

- Terrible loss of time
- Lack of freedom to go where I want
- the burden of visiting relatives
- the expense and anxiety provoked by children
- Maybe she will not like London
- Less money to spend on books

What you think is Darwin doing in terms of machine learning ?

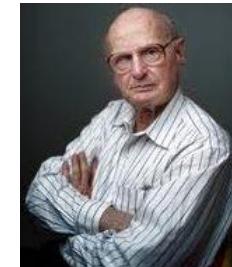


## What is Markowitz doing ?

The economist Harry Markowitz won the 1990 Nobel Prize in Economics for developing modern portfolio theory: his groundbreaking “mean-variance portfolio optimization” showed how an investor could make an optimal allocation among various funds and assets to maximize returns at a given level of risk.

So when it came time to invest his own retirement savings, it seems like Markowitz should have been the one person perfectly equipped for the job. What did he decide to do?

*I should have computed the historical covariances of the asset classes and drawn an efficient frontier. Instead, I visualized my grief if the stock market went way up and I wasn't in it—or if it went way down and I was completely in it. My intention was to minimize my future regret. So I split my contributions **fifty-fifty** between bonds and equities*



- Compare your model with the baseline accuracy
  - What will be your prediction and error if you haven't done any modeling?
- Compare the performance of different models on the testing data set. Choose the one with highest performance.

Are people happy at marriage ?

Average happiness by year in marriage

create the dataframe

```
marriage <- data.frame(Happy = c(7.8, 7.6, 7.4, 7.6, 7.65, 7.4, 7.35, 7.3, 7.4, 7.5),  
                        Year = 1:10)
```

Plot the data using ggplot

2 new layers

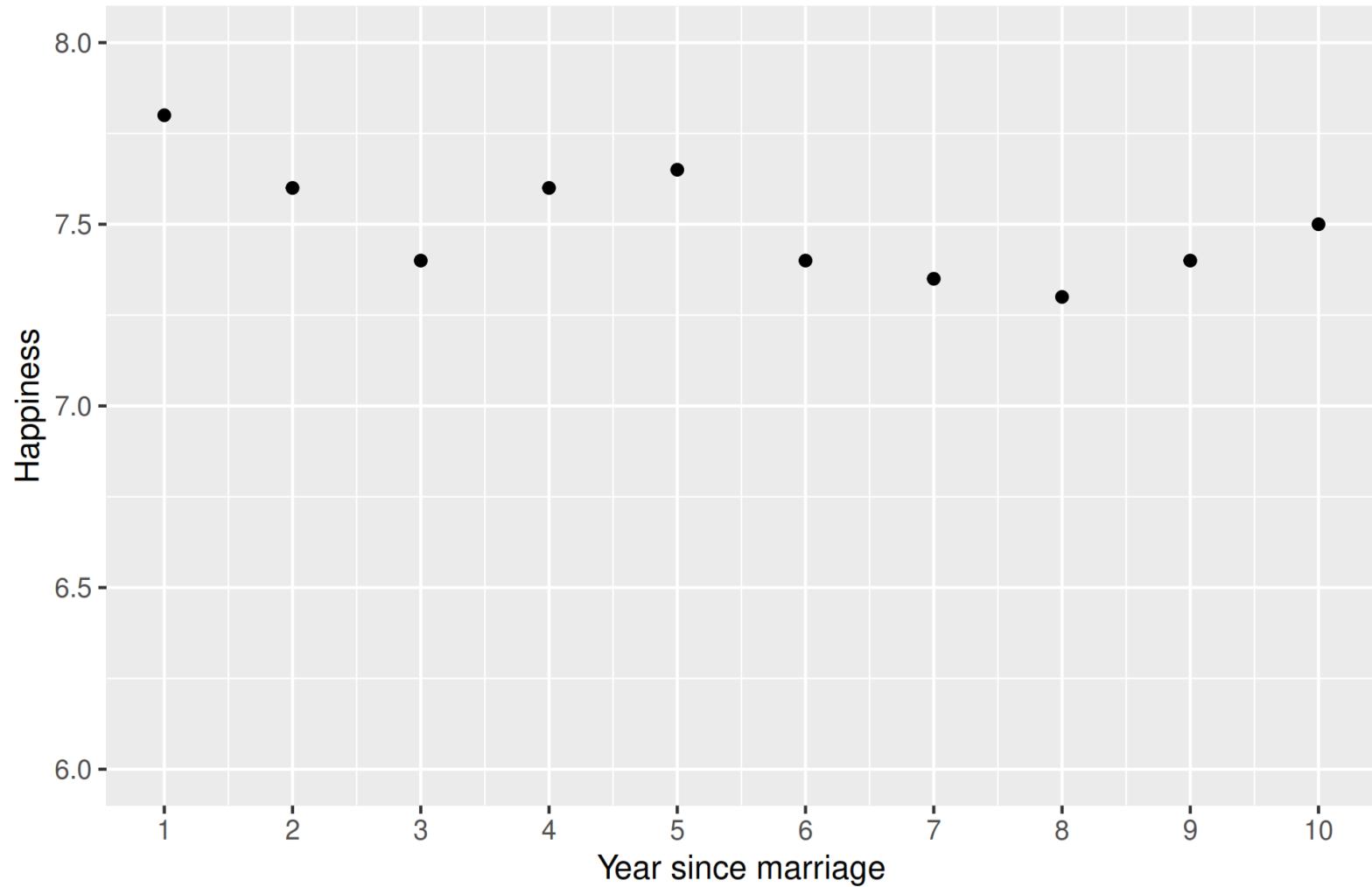
- `ylim` – defines limits of y axis
- `scale_x_continuous` – sets the breaks on the ax axis

To see better what these layers are doing try to do the plotting without them and compare

```
ggplot(marriage, aes(x = Year, y = Happy)) + geom_point() + ylim(c(6, 8)) +
  scale_x_continuous(breaks = 1:10) + labs(x = "Year since marriage", y = "Happiness",
  title = "Average happiness by year in marriage")
```

```
ggplot(marriage, aes(x = Year, y = Happy)) + geom_point() + ylim(c(6, 8)) +
  scale_x_continuous(breaks = 1:10) + labs(x = "Year since marriage", y = "Happiness",
  title = "Average happiness by year in marriage")
```

Average happiness by year in marriage



## Fit the regression line

```
ggplot(marriage, aes(x=Year, y=Happy))+geom_point()+
  ylim(c(6,8))+scale_x_continuous(breaks=1:10)+
  labs(x="Year since marriage", y="Happiness",
       title="Average happiness by year in marriage")+
  geom_smooth(method='lm', se=FALSE)
```



```
mar1 <- lm(Happy~Year, data=marriage)
summary(mar1)

##
## Call:
## lm(formula = Happy ~ Year, data = marriage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18485 -0.09508  0.00000  0.11205  0.15273
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.68667   0.08707  88.279 3.03e-13 ***
## Year        -0.03394   0.01403  -2.419   0.0419 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1275 on 8 degrees of freedom
## Multiple R-squared:  0.4224, Adjusted R-squared:  0.3502
## F-statistic: 5.849 on 1 and 8 DF,  p-value: 0.04194
```

- Fit polynomial regression model of second order – quadratic regression

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2$$

- Polynomial regression model of third order – cubic regression

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

```
mar2 <- lm(Happy~poly(Year,2), data=marriage)
summary(mar2)

##
## Call:
## lm(formula = Happy ~ poly(Year, 2), data = marriage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.16932 -0.05184 -0.01659  0.05818  0.19515 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 7.50000   0.03744 200.310 2.04e-14 ***
## poly(Year, 2)1 -0.30827   0.11840  -2.604   0.0352 *  
## poly(Year, 2)2  0.17843   0.11840   1.507   0.1755  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1184 on 7 degrees of freedom
## Multiple R-squared:  0.5639, Adjusted R-squared:  0.4392 
## F-statistic: 4.525 on 2 and 7 DF,  p-value: 0.05479
```

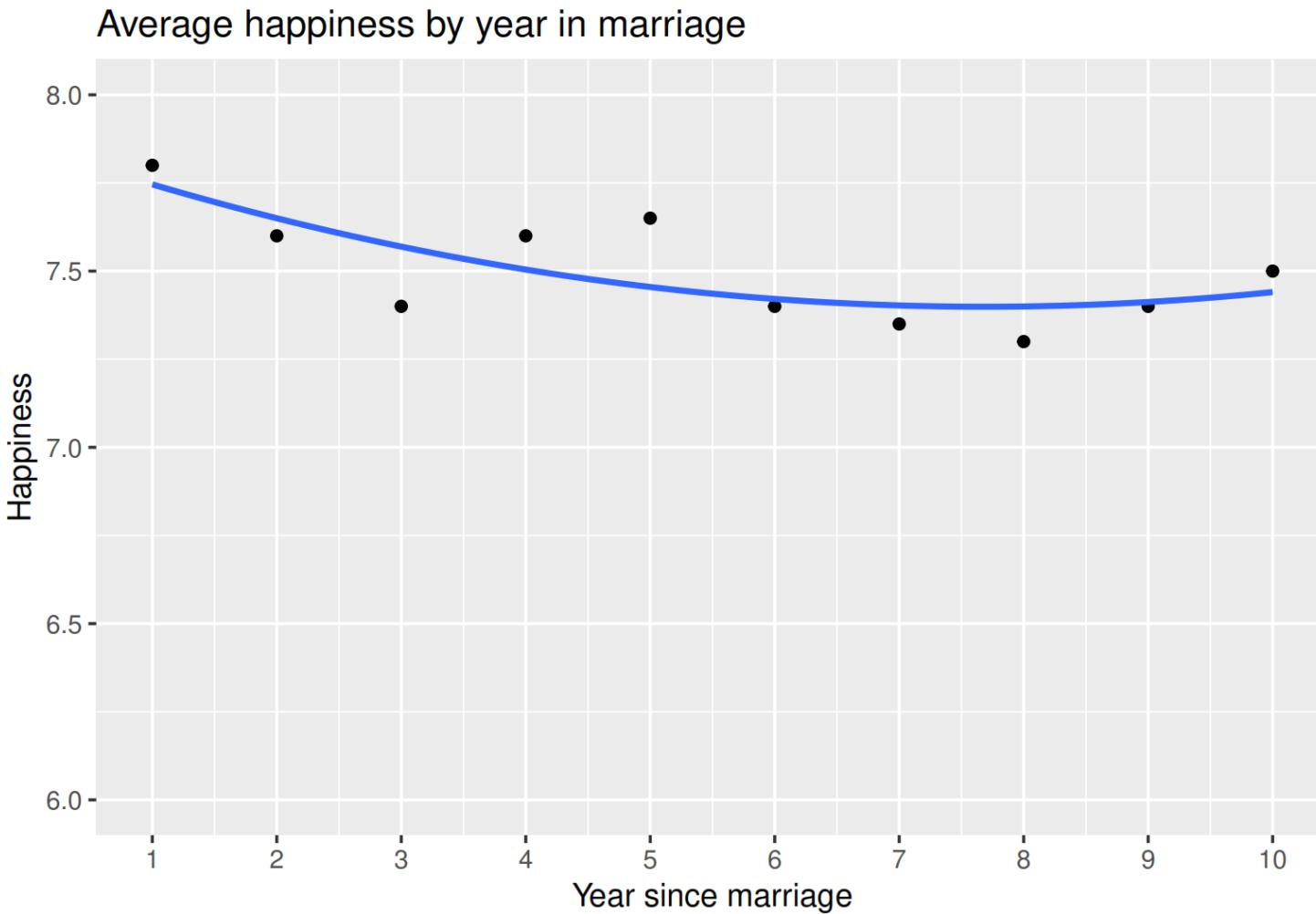
## Plot regression line using stat\_smooth() layer

```
ggplot(marriage, aes(x=Year, y=Happy)) + geom_point() +  
  ylim(c(6,8)) + scale_x_continuous(breaks=1:10) +  
  stat_smooth(method='lm', formula = y~poly(x,2), se=FALSE) +  
  labs(x="Year since marriage", y="Happiness",  
       title="Average happiness by year in marriage")
```

```

ggplot(marriage, aes(x=Year, y=Happy)) + geom_point() +
  ylim(c(6,8)) + scale_x_continuous(breaks=1:10) +
  stat_smooth(method='lm', formula = y~poly(x,2), se=FALSE) +
  labs(x="Year since marriage", y="Happiness",
       title="Average happiness by year in marriage")

```



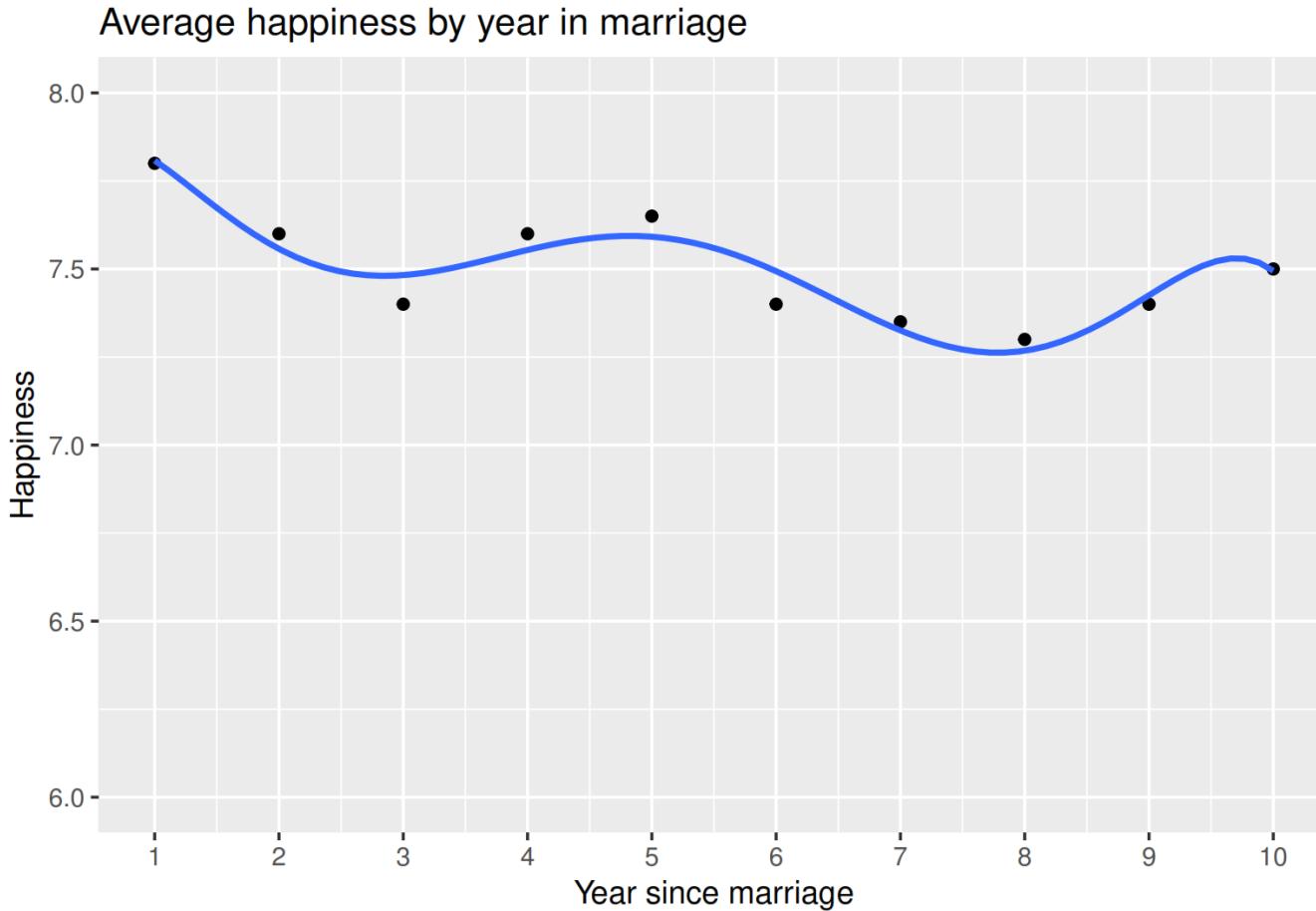
```
ggplot(marriage, aes(x=Year, y=Happy))+geom_point()+
  ylim(c(6,8))+scale_x_continuous(breaks=1:10)+  

  stat_smooth(method='lm', formula = y~poly(x,6), se=FALSE)+  

  labs(x="Year since marriage", y="Happiness",  

       title="Average happiness by year in marriage")
```

## Sixth order polynom



```

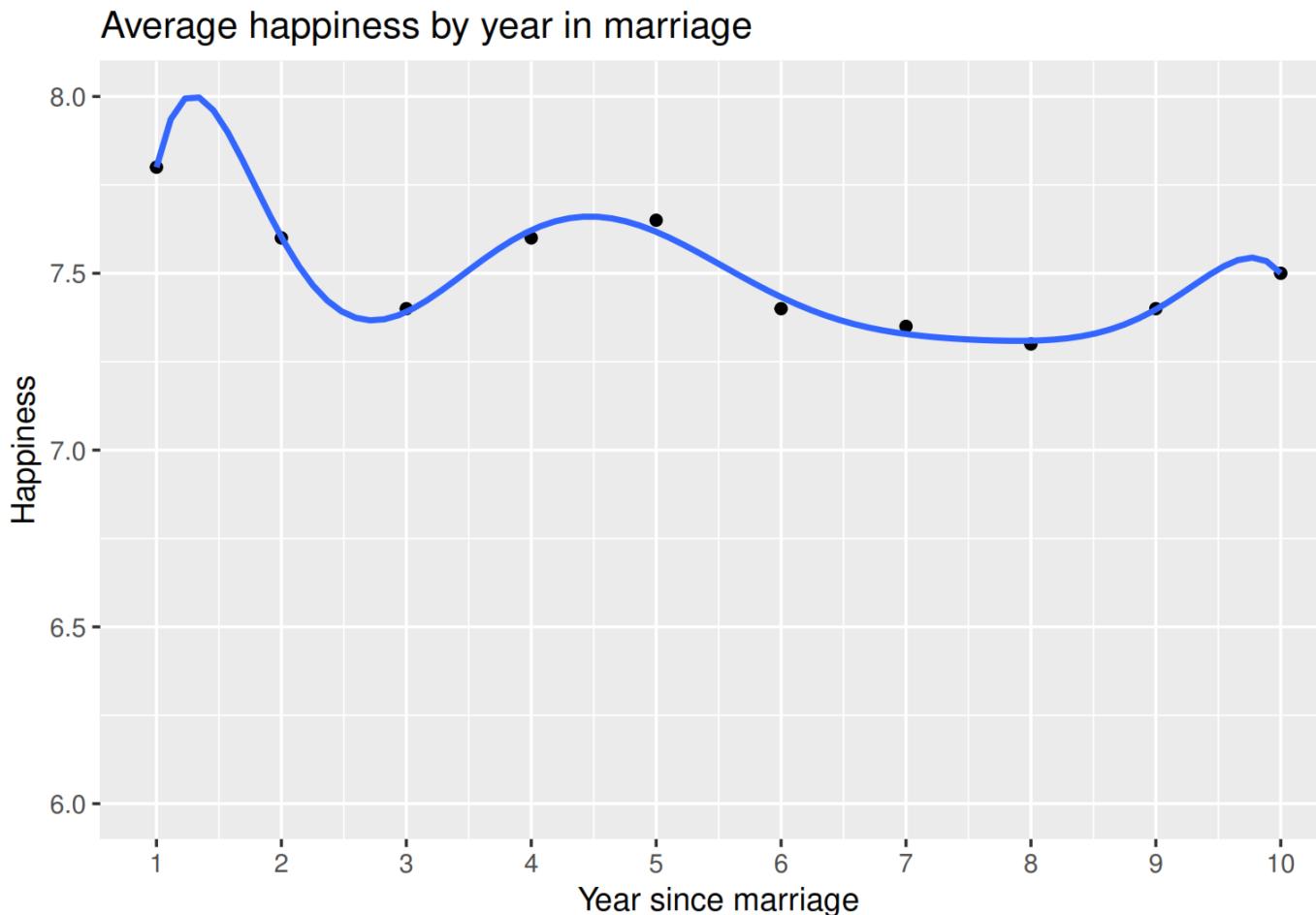
mar6 <-lm(Happy~poly(Year,6), data=marriage)
summary(mar6)

##
## Call:
## lm(formula = Happy ~ poly(Year, 6), data = marriage)
##
## Residuals:
##       1        2        3        4        5        6        7
## -0.007972  0.042890 -0.082471  0.046014  0.058415 -0.093660  0.024476
##       8        9       10
##  0.032121 -0.025268  0.005455
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.50000   0.02904 258.235 1.28e-07 ***
## poly(Year, 6)1 -0.30827   0.09184 -3.356  0.0438 *
## poly(Year, 6)2  0.17843   0.09184  1.943  0.1473
## poly(Year, 6)3  0.04804   0.09184  0.523  0.6371
## poly(Year, 6)4  0.21036   0.09184  2.290  0.1059
## poly(Year, 6)5 -0.11995   0.09184 -1.306  0.2827
## poly(Year, 6)6 -0.10899   0.09184 -1.187  0.3208
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09184 on 3 degrees of freedom
## Multiple R-squared:  0.8875, Adjusted R-squared:  0.6626
## F-statistic: 3.946 on 6 and 3 DF,  p-value: 0.1436

```

```
ggplot(marriage, aes(x=Year, y=Happy))+geom_point()+
  ylim(c(6,8))+scale_x_continuous(breaks=1:10)+
  stat_smooth(method='lm', formula = y~poly(x,8), se=FALSE)+
  labs(x="Year since marriage", y="Happiness",
       title="Average happiness by year in marriage")
```

8<sup>th</sup> degree  
polynom

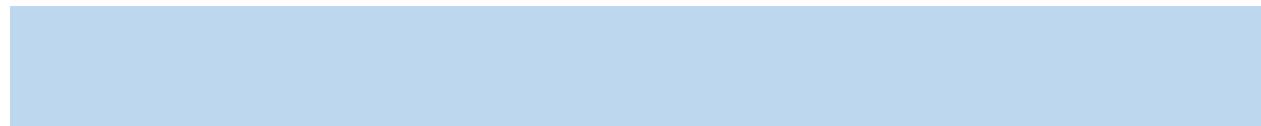


```
mar8 <- lm(Happy~poly(Year,8), data=marriage)
summary(mar8)

##
## Call:
## lm(formula = Happy ~ poly(Year, 8), data = marriage)
##
## Residuals:
##       1        2        3        4        5        6
## 0.0002592 -0.0023324  0.0093295 -0.0217688  0.0326532 -0.0326532
##       7        8        9       10
## 0.0217688 -0.0093295  0.0023324 -0.0002592
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.50000   0.01807 415.048  0.00153 ***
## poly(Year, 8)1 -0.30827   0.05714  -5.395  0.11668
## poly(Year, 8)2  0.17843   0.05714   3.123  0.19731
## poly(Year, 8)3  0.04804   0.05714   0.841  0.55495
## poly(Year, 8)4  0.21036   0.05714   3.681  0.16886
## poly(Year, 8)5 -0.11995   0.05714  -2.099  0.28303
## poly(Year, 8)6 -0.10899   0.05714  -1.907  0.30742
## poly(Year, 8)7  0.13232   0.05714   2.316  0.25953
## poly(Year, 8)8 -0.06732   0.05714  -1.178  0.44808
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05714 on 1 degrees of freedom
## Multiple R-squared:  0.9855, Adjusted R-squared:  0.8694
## F-statistic: 8.488 on 8 and 1 DF,  p-value: 0.2597
```

What if you try 9<sup>th</sup>  
order polynomial  
regression ?

# Model selection in practice



## General approach

- Create Testing and Training datasets
- Build your models on Training data set
- Evaluate your model performance on Testing set
- Choose the model that does the best on testing set

- Model performance is always evaluated based on testing dataset
- For different types of models (regression, classification), different types of performance measures exists
- For regression analysis, we will use **Root Mean Squares of Errors (RMSE)**

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

- The goal is to predict median household value (medv)
- Dataset Boston from library MASS

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.1
```

```
data(Boston)
```

```
## ?MASS::Boston
```

- Training and testing sets need to be generated at random.
- There is no true randomness in computer science, so we use pseudorandom numbers
- When you use `set.seed(Number)` with the same Number as an argument you are going to get the same “random” output

```
set.seed(1)
sample <- sample(nrow(Boston), floor(nrow(Boston) * 0.7))
sample[1:5]

## [1] 135 188 289 457 102

length(sample)

## [1] 354
```

## Create training and testing datasets

- The vector sample contains rownumber that you want to sample
- Boston[sample,] will give a dataframe with given rownumbers
- Boston[-sample,] will give a dataframe with wverything else (that is not sample)

```
Train<-Boston[sample,]  
Test<-Boston[-sample,]
```

Here in the formula we use ~.

This means use medv as dependent variable and ALL other variables as independent

```
model <- lm(medv~., data=Train)
```

- To do predictions on the Test set, we will use generic function (method) predict.
- In case of linear regression model, it takes at least two
  - model
  - newdata

```
pred1 <- predict(model, newdata = Test)
pred1[1:10]
```

```
##          1         2         3        20        22        23        24        26
## 30.04244 25.08534 30.88295 18.65169 17.73878 15.75622 13.66044 13.48754
##          27         30
## 15.58626 21.03303
```

## Calculate Root Mean Square

```
sqrt(mean((Test$medv-pred1)^2))  
## [1] 4.256622
```

On average our model is wrong by 4.25

```

summary(model)

##
## Call:
## lm(formula = medv ~ ., data = Train)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -15.7559 -2.8044 -0.8952  1.6246 26.4958 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 34.693176   6.380806   5.437 1.04e-07 ***
## crim        -0.136079   0.040298  -3.377 0.000818 ***
## zn          0.025453   0.018090   1.407 0.160353    
## indus       -0.018390   0.076381  -0.241 0.809876    
## chas         2.598633   1.186039   2.191 0.029129 *  
## nox        -14.126596   4.771804  -2.960 0.003288 ** 
## rm          3.820468   0.534460   7.148 5.40e-12 ***
## age        -0.006563   0.016846  -0.390 0.697074    
## dis         -1.359358   0.249090  -5.457 9.33e-08 ***
## rad          0.321100   0.088730   3.619 0.000341 *** 
## tax        -0.013205   0.004912  -2.688 0.007537 ** 
## ptratio     -0.907350   0.164474  -5.517 6.86e-08 *** 
## black        0.009071   0.003450   2.630 0.008935 ** 
## lstat       -0.547723   0.063813  -8.583 3.35e-16 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 5.001 on 340 degrees of freedom
## Multiple R-squared:  0.7154, Adjusted R-squared:  0.7045 
## F-statistic: 65.74 on 13 and 340 DF,  p-value: < 2.2e-16

```

Variable indus is not significant, lets remove it from the model

Look at how the formula is specified:

We want all variables as independent variables except indus

```
model2 <- lm(medv~.-indus, data=Train)
pred2 <- predict(model2, newdata = Test)
sqrt(mean((Test$medv-pred2)^2))
```

```
## [1] 4.25144
```

We see slight improvement in RMSE

Remove variable age as well (another non-significant variable)

```
model3 <- lm(medv~.-indus-age, data=Train)
pred3 <- predict(model3, newdata = Test)
sqrt(mean((Test$medv-pred3)^2))

## [1] 4.242682
```

What if we take only two variables (2 that have highest correlation with the dependent variable)

```
model4 <- lm(medv~rm+lstat, data=Train)
pred4 <- predict(model4, newdata = Test)
sqrt(mean((Test$medv-pred4)^2))

## [1] 5.275777
```

What if we have used different Train and Test sets ?

```
sample1 <- sample(nrow(Boston), floor(nrow(Boston) * 0.7))
sample2 <- sample(nrow(Boston), floor(nrow(Boston) * 0.7))
sample3 <- sample(nrow(Boston), floor(nrow(Boston) * 0.7))
```

```
Train1 <- Boston[sample1,]
Test1 <- Boston[-sample1,]
```

```
Train2 <- Boston[sample2,]
Test2 <- Boston[-sample2,]
```

```
Train3 <- Boston[sample3,]
Test3 <- Boston[-sample3,]
```

```
pred1 <- predict(model1, newdata = Test1)
pred2 <- predict(model1, newdata = Test2)
pred3 <- predict(model1, newdata = Test3)

sqrt(mean((Test1$medv-pred1)^2))

## [1] 5.372828

sqrt(mean((Test2$medv-pred2)^2))

## [1] 4.76811

sqrt(mean((Test3$medv-pred3)^2))

## [1] 5.317002
```