Lecture 10

Cross-validation

Julian Reif
Fall 2025

RStudio setup for this lecture

- Log into RStudio on your Amazon EC2 instance
  - Use AMI `FIN550-RStudio` with IAM role `BigDataEC2Role`

```
# Enter this command using RStudio Terminal
aws s3 cp --recursive s3://bigdata-fin550-reif/lecture-10 ~/fin550/lecture-10
```

Statistics review

R-squared is a measure of fit that ranges from 0 to 1

$$
\begin{aligned}
R^2 &= ESS/TSS \\
&= 1 - RSS/TSS
\end{aligned}
$$

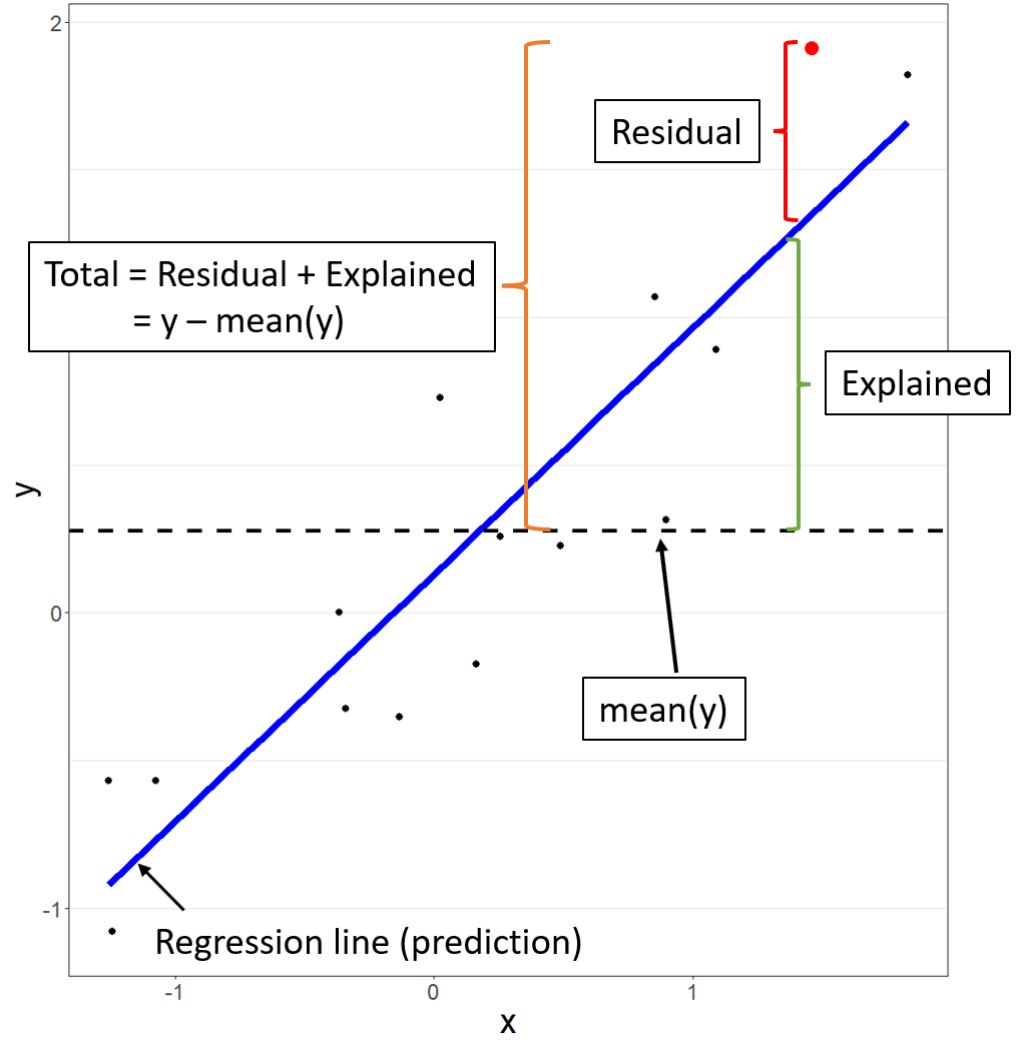
where:

- ESS: explained sum of squares
- TSS: total sum of squares
- RSS: residual sum of squares (squared prediction error)

- R^2 ranges from 0 to 1
  - $R^2 = 0$: model explains 0% of the variation in Y
  - $R^2 = 1$: model explains 100% of the variation in Y

Illustration of R-squared

- A high R^2 means:
  - Residuals are small
  - Explained variation is high



Linear regression and R-squared

- Recall: linear regression minimizes RSS (squared prediction error)

- Thus, $R^2 = 1 - RSS/TSS$ increases every time a predictor is added to a regression

- When the number of predictors equals the number of observations, $R^2 = 1$!
  - This is an example of "overfitting" your data

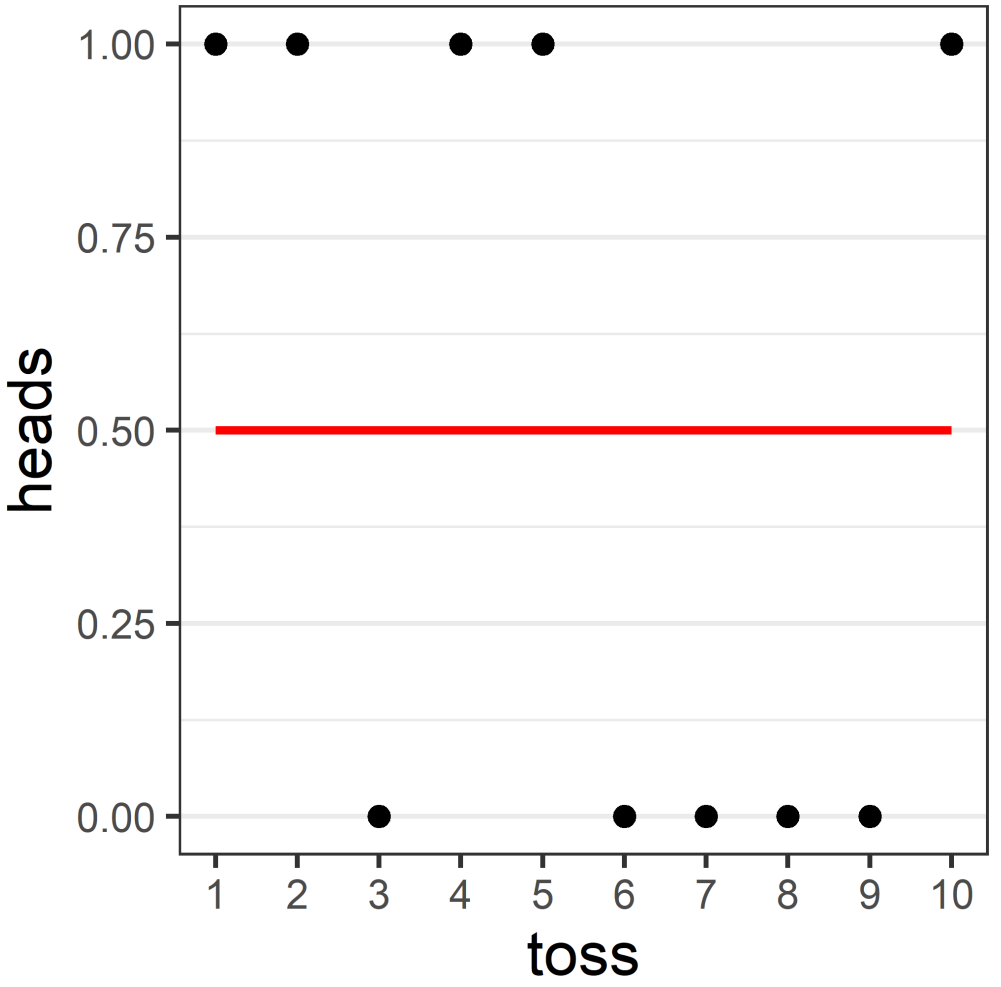
Flip a virtual coin 10 times

```
library(tidyverse)
set.seed(26) # Set seed for replicability

flips <- tibble(
  toss = (1:10),
  heads = sample(0:1,
                 size = length(toss),
                 replace = TRUE)
)

flips
```

```
# # A tibble: 10 × 2
#     toss heads
#    <int> <int>
#  1     1     1
#  2     2     1
#  3     3     0
#  4     4     1
#  5     5     1
#  6     6     0
#  7     7     0
#  8     8     0
#  9     9     0
# 10    10     1
```

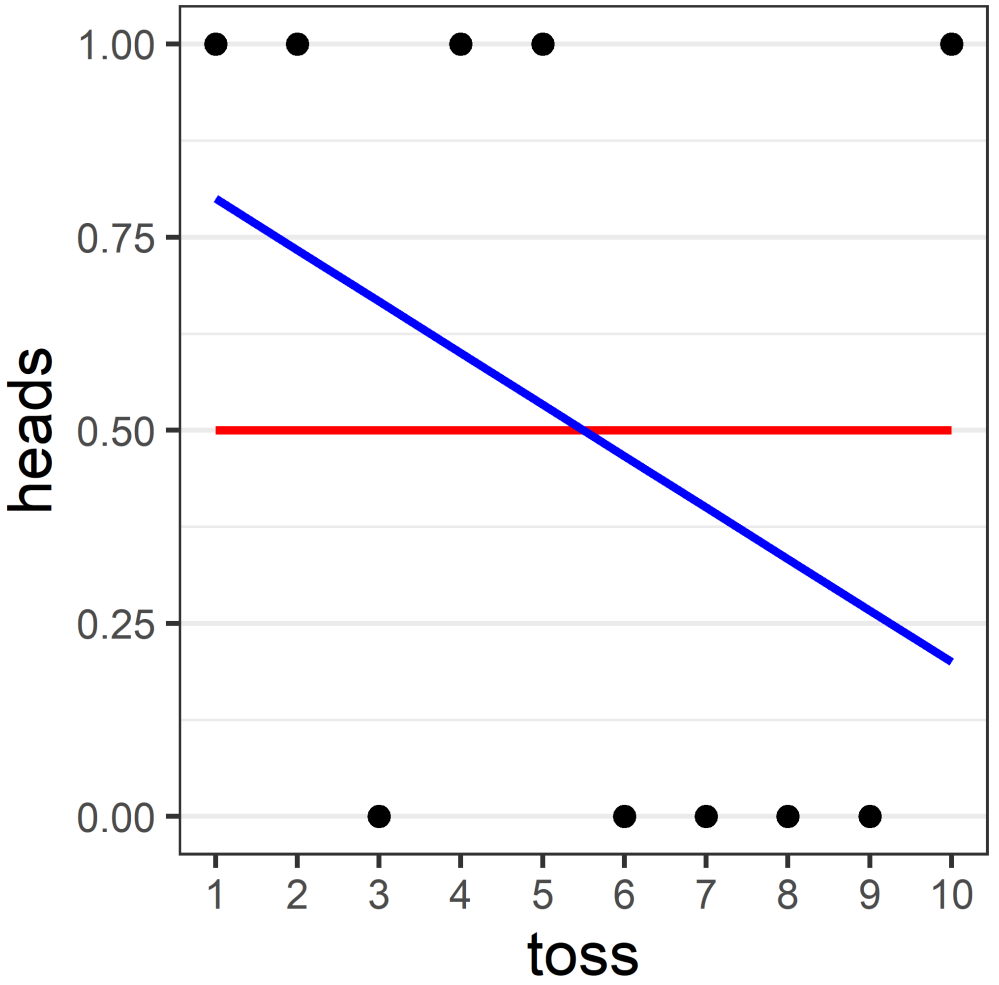
Model 1: no predictors

```
# Model 1: no predictors (intercept only)
fit1 <- lm(heads ~ 1,
           data = flips)


# Plot predicted versus actual outcomes
gg <- flips %>%
  ggplot(aes(x = toss, y = heads)) +
  geom_point(size=5) +
  geom_line(aes(y=fit1$fitted.values),
            color="red", linewidth=2) +
  scale_x_continuous(breaks = c(1:10))
gg
```



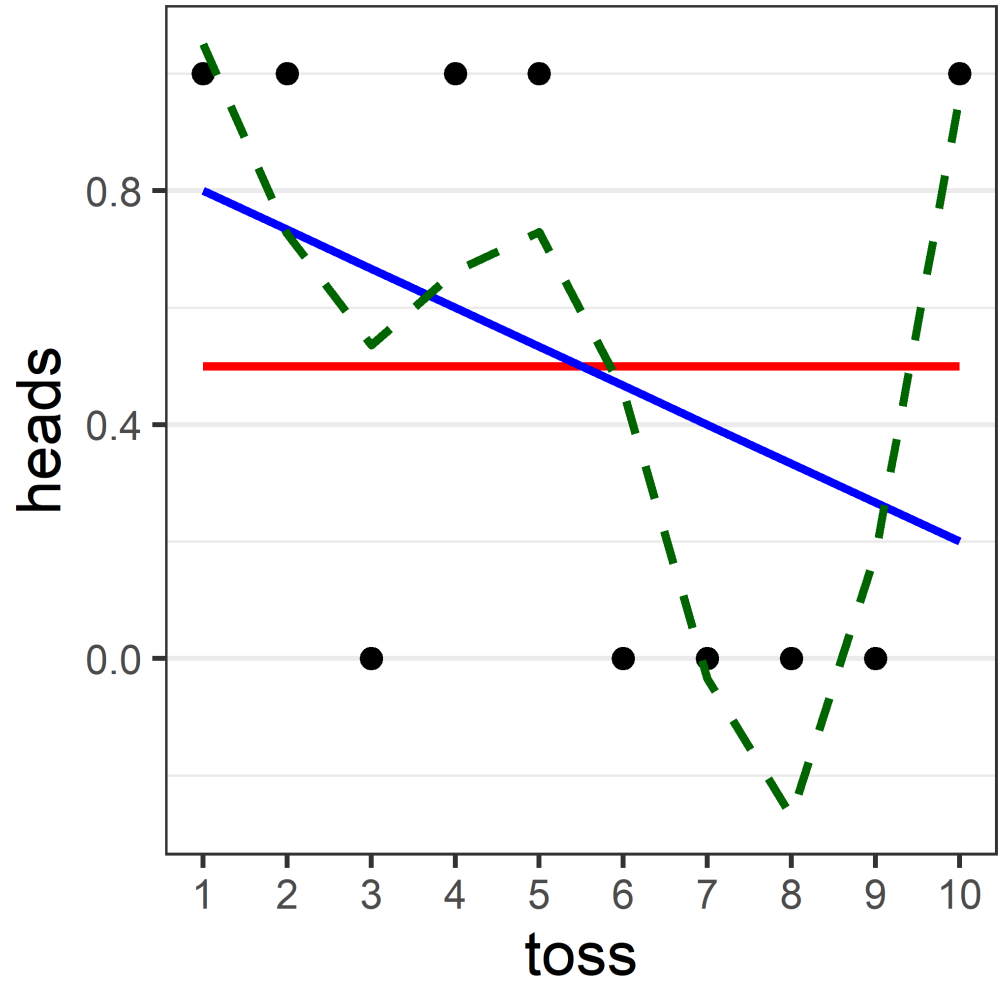
Model 2: toss order as predictor

```
# Model 2: toss order as predictor
fit2 <- lm(heads ~ toss,
           data = flips)

# Plot predicted versus actual outcomes
gg <- gg +
  geom_line(aes(y=fit2$fitted.values),
            color="blue", linewidth=2)
gg
```



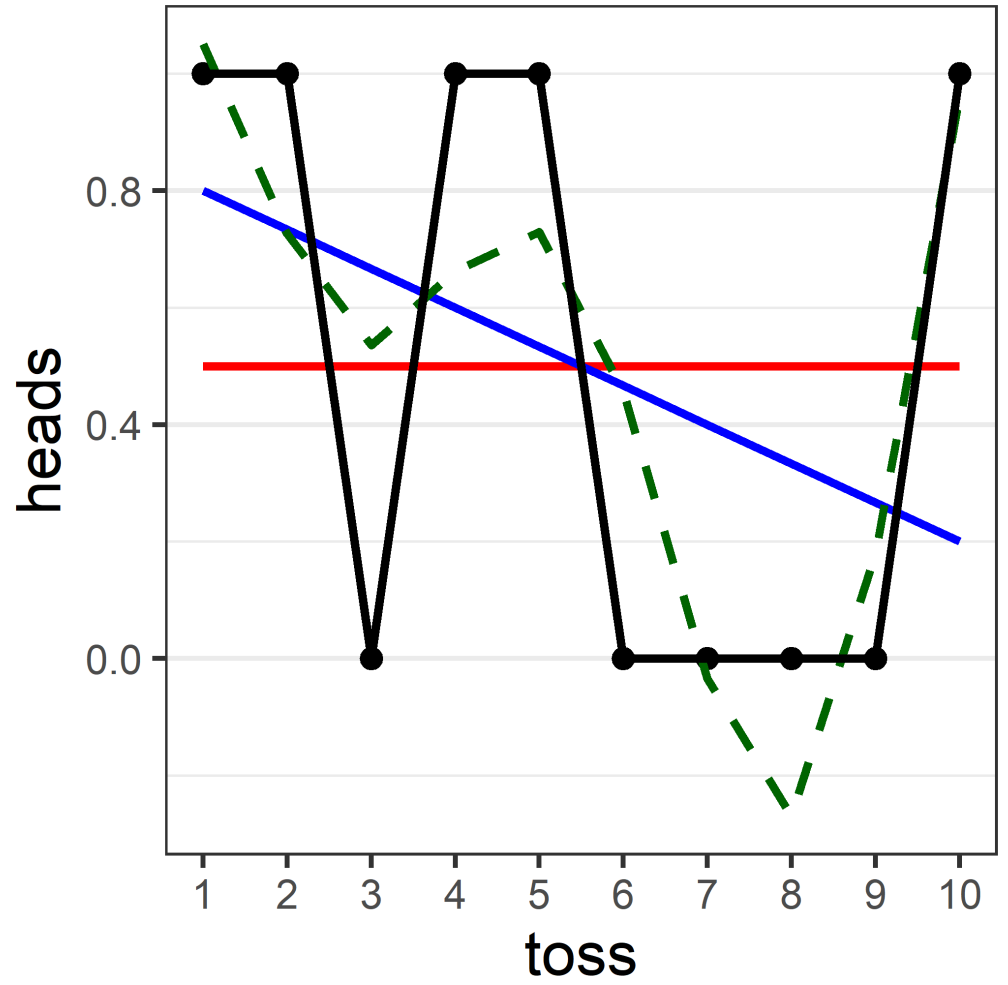
Model 3: polynomial in toss order (degree 6)

```
# Model 3: toss order poly(6) as predictor
fit3 <- lm(heads ~ poly(toss,6),
           data = flips)

# Plot predicted versus actual outcomes
gg <- gg +
  geom_line(aes(y=fit3$fitted.values),
            color="darkgreen",
            linetype="dashed",
            linewidth=2)
gg
```



Model 4: polynomial in toss order (degree 9)

```
# Model 4: toss order poly(9) as predictor
fit4 <- lm(heads ~ poly(toss,9),
           data = flips)

# Plot predicted versus actual outcomes
gg <- gg +
  geom_line(aes(y=fit4$fitted.values),
            color="black", linewidth=2)
gg
```



R-squared and MSE for the four models

```r
models <- list(fit1, fit2, fit3, fit4)

# Loop over each model and report its MSE & R^2
for(x in 1:4) {

  r2 <- round(summary(models[[x]])$r.squared, 3)

  p <- predict.lm(models[[x]])
  MSE <- round(mean((flips$heads - p)^2), 3)

  print(paste("R2  for model", x, "is:", r2))
  print(paste("MSE for model", x, "is:", MSE))
  cat("\n")
}
```

```
# [1] "R2  for model 1 is: 0"
# [1] "MSE for model 1 is: 0.25"
#
# [1] "R2  for model 2 is: 0.147"
# [1] "MSE for model 2 is: 0.213"
#
# [1] "R2  for model 3 is: 0.652"
# [1] "MSE for model 3 is: 0.087"
#
# [1] "R2  for model 4 is: 1"
# [1] "MSE for model 4 is: 0"
```

Cross-validation

Training data vs test data

- Training data: used to estimate the model ($\hat{f}$)
- Test data: used to evaluate the model
- A model that fits the training data well might not fit the test data well

Create test data for coin-flipping example

```
set.seed(25) # Note: different seed from training data

flips_test_data <- tibble(
  toss = (1:10),
  heads = sample(0:1,
                 size = length(toss),
                 replace = TRUE)
)
```

Compute mean-squared error (MSE) using the test data

```
models <- list(fit1, fit2, fit3, fit4)

# How do the models with lots of predictors perform?
for(x in 1:4) {
  p <- predict.lm(models[[x]], newdata = flips_test_data)
  MSE <- round(mean((flips_test_data$heads - p)^2), 3)
  print(paste("Test data MSE for model", x, "is", MSE))
}
```

```
# [1] "Test data MSE for model 1 is 0.25"
# [1] "Test data MSE for model 2 is 0.3"
# [1] "Test data MSE for model 3 is 0.544"
# [1] "Test data MSE for model 4 is 0.7"
```

Types of cross-validation

- Validation set approach

- Leave-one-out cross-validation (LOOCV)

- k-fold cross-validation

Validation sets

1. Randomly divide the data into two "folds"
    - One fold is used to **train** the model
    - Second fold is held out and later used to **test** the model

2. Fit the model on the training data

3. Use fitted model from step 2 to predict outcomes in the test data

4. Error rate is the MSE of the predictions in step 3

Key drawback: training data only includes half the observations

Leave-one-out cross-validation (LOOCV)

1. Choose 1 observation from the data to be the test data
    - Remaining $n - 1$ observations used to train the model

2. Fit the model on the training data

3. Use fitted model from step 2 to predict outcomes in test data (which has 1 data point)

4. Repeat steps 1-3 for each observation in the data
    - Error rate is the average MSE across all test datasets

Key drawback: computationally intensive, especially for large datasets (why?)

k-fold cross-validation offers a balance

1. Randomly divide the data into k folds
    - Select one fold to **test** the model
    - Remaining $k-1$ folds are used to **train** the model

2. Fit the model on the training data ($k-1$ folds)

3. Use fitted model to predict outcomes in the test data (1 fold)

4. Repeat steps 1-3, using a different fold to test the model each time
    - Error rate is the average MSE across all k folds

Note: LOOCV is special case of k-fold cross-validation with $k = n$

Which procedure works best for estimating test error?

- How to choose k, the number of folds?

- We face a bias-variance tradeoff in the cross-validated error estimate:
  - LOOCV $(k = n)$
    - Has more data in each training dataset $\rightarrow$ lower bias
    - But, those training datasets are nearly identical $\rightarrow$ higher variance
  - Small k (e.g., 2-fold)
    - Training sets are much smaller $\rightarrow$ CV error is biased upward
    - Error estimates are averaged over larger test folds $\rightarrow$ lower variance

- In practice, $k = 5$ or $k = 10$ is usually a good compromise, delivering the best estimate

Cross-validation in R

Try it: load libraries, turn off scientific notation

```
library(tidyverse)
library(ggplot2)

# scipen: a penalty for deciding whether to print fixed or exponential notation.
# Positive values encourage fixed notation, negative encourage scientific notation
options(scipen=999)
```

Try it: load and inspect the Boston housing dataset

```
# Housing information for 506 areas around Boston
housing <- read_csv("lecture-10-housing.csv")
nrow(housing)
ncol(housing)
```

```
# [1] 506
# [1] 14
```

Variables come from the 1970 Census

Variable name	Definition
MEDV	Median value of owner-occupied homes in $1000's
CRIM	Per capita crime rate by census tract
ZN	Proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	Proportion of non-retail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX	Nitric oxides concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	Pupil-teacher ratio by town
LSTAT	% lower status of the population

Model housing value as function of the number of rooms

- `MEDV`: median housing value in the area in 1970
- `RM`: average number of rooms per house in the area

```
summary(housing$MEDV)
cat("\n")
summary(housing$RM)
```

```
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#    5.00   17.02   21.20   22.53   25.00   50.00
#
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   3.561   5.886   6.208   6.285   6.623   8.780
```

Employ a polynomial in the number of rooms

- We will model value as function of j-degree polynomial in the number of rooms

$$MEDV = \beta_0 + \sum_{i=1}^{j} \beta_i RM^i + \epsilon$$

- What degree (j) should we use?

- Use cross-validation to decide

Try it: validation set approach

```
# Set seed (why is this needed?)
set.seed(1)
n_rows <- nrow(housing)

# Randomly select half the observations
train <- sample(n_rows, n_rows/2)

# RM degree 1
# 1. Estimate model using training observations only by specifying `subset` option
lm_1 <- lm(MEDV ~ RM, data = housing, subset = train)

# 2. Calculate MSE using test data
lm_1_mse <- mean((housing$MEDV - predict(lm_1, housing))[-train]^2)
lm_1_mse
```

```
# [1] 49.65573
```

Validation set approach: higher degrees

```
# RM degree 2
lm_2 <- lm(MEDV ~ poly(RM, 2), data = housing, subset = train)
lm_2_mse <- mean((housing$MEDV - predict(lm_2, housing))[-train]^2)
lm_2_mse

# RM degree 3
lm_3 <- lm(MEDV ~ poly(RM, 3), data = housing, subset = train)
lm_3_mse <- mean((housing$MEDV - predict(lm_3, housing))[-train]^2)
lm_3_mse
```

```
# [1] 50.59916
# [1] 48.63831
```

This code is getting repetitive....

- Write a function!

- Function will calculate the MSE for different models
  - Different training samples
  - Different polynomial orders

Try it: write the function

```
f_mse <- function(order=10, seed=NULL) {
  mse_lm <- rep(0,order)
  set.seed(seed)
  n_rows <- nrow(housing)
  train <- sample(n_rows, n_rows/2)

  for (j in 1:order){
    # Estimate training model where Y = MEDV and X = polynomial in rooms of order j
    lm <-

    # Calculate the MSE using test data (hint: look at code on previous slide)
    mse_lm[j] <-
  }

  return(mse_lm)
}
```

Write the function

```
f_mse <- function(order=10, seed=NULL) {
  mse_lm <- rep(0,order)
  set.seed(seed)
  n_rows <- nrow(housing)
  train <- sample(n_rows, n_rows/2)

  for (j in 1:order){
    # Estimate training model where Y = MEDV and X = polynomial in rooms of order j
    lm <- lm(MEDV ~ poly(RM, j), data = housing, subset = train)

    # Calculate the MSE using test data (hint: look at code on previous slide)
    mse_lm[j] <- mean((housing$MEDV - predict(lm, housing))[-train]^2)
  }

  return(mse_lm)
}
```

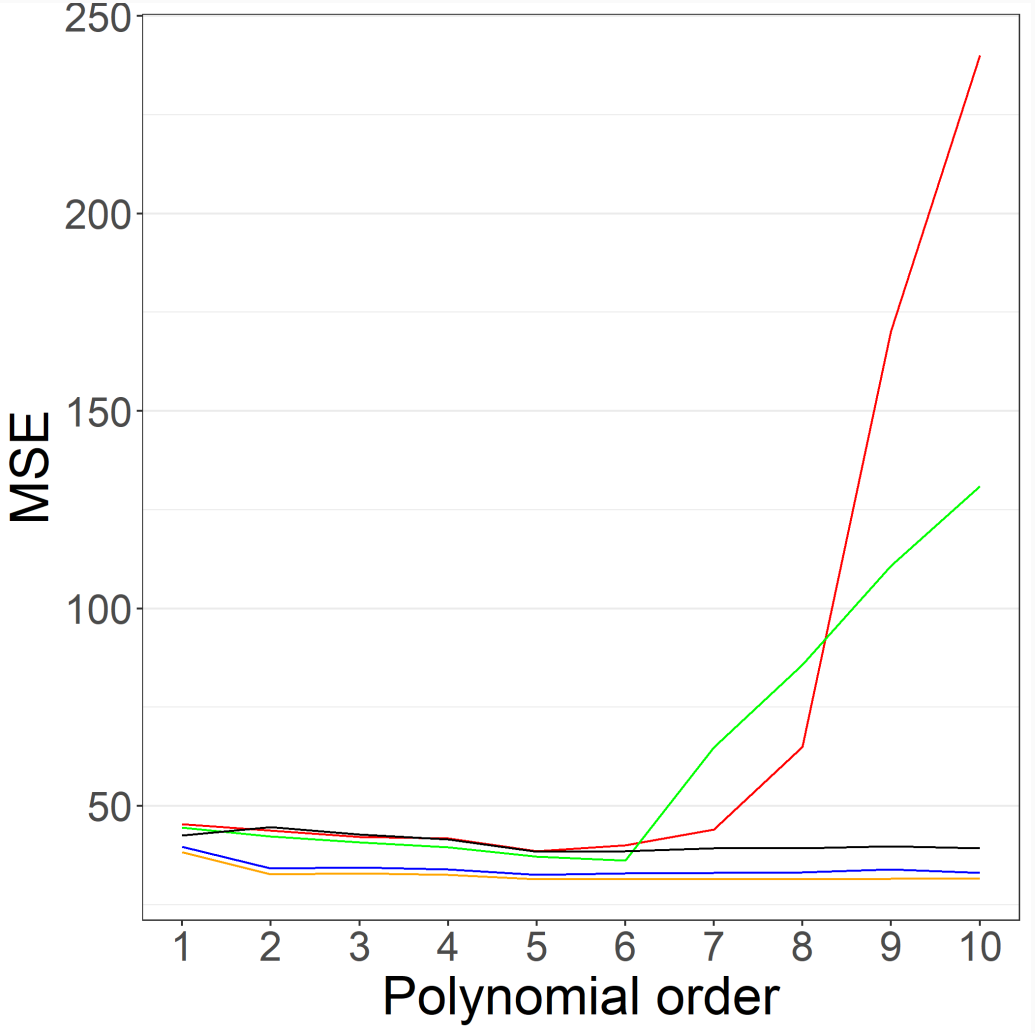
Try it: call the function using different seeds

```
# Confirm that we get the same results as previous slide (RM degree 1, 2, 3)
f_mse(3,1)

mse_1 <- f_mse(10, 891)
mse_2 <- f_mse(10, 892)
mse_3 <- f_mse(10, 893)
mse_4 <- f_mse(10, 894)
mse_5 <- f_mse(10, 895)

# Build a data frame with the results
mse <- data.frame(mse_1, mse_2, mse_3, mse_4, mse_5) %>%
  mutate(order = row_number())
```

```
# [1] 49.65573 50.59916 48.63831
```

Try it: plot our results

```
# Show MSE as a function of the polynomial degree
#   for the 5 different seeds we tried
ggplot(mse, aes(x=order)) + theme_bw() +
  geom_line(aes(y = mse_1), color = "red") +
  geom_line(aes(y = mse_2), color = "blue") +
  geom_line(aes(y = mse_3), color = "green") +
  geom_line(aes(y = mse_4), color = "orange") +
  geom_line(aes(y = mse_5), color = "black") +
  scale_x_continuous(breaks = c(1:10)) +
  ylab("MSE") + xlab("Polynomial order") +
  theme(
    axis.title = element_text(size = 26),
    axis.text = element_text(size = 20),
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank()
  )
```



Leave-One-Out Cross-Validation (LOOCV)

```r
library(boot) # Calculate LOOCV using cv.glm(), part of `boot` library

# RM degree 1. No need to set a seed with LOOCV (why not?)
glm_1 <- glm(MEDV ~ RM, data = housing) # GLM default is regression
cv_err_1 <- cv.glm(housing, glm_1)      # cv.glm default is LOOCV

# Value of K (LOOCV is k-fold with k = n)
cv_err_1$K

# Cross-validation estimate of prediction error
# We care about the first number: raw prediction error (not adjusted)
cv_err_1$delta[1]
```

```
# [1] 506
# [1] 44.21666
```

Estimate LOOCV for multiple polynomials (slow)

```r
for (j in 1:12) {
  glm_j <- glm(MEDV ~ poly(RM,j), data = housing)
  cv_err_j <- cv.glm(housing, glm_j)$delta[1]
  print(paste("MSE for order",j,"is",round(cv_err_j,2)))
}
```

```
# [1] "MSE for order 1 is 44.22"
# [1] "MSE for order 2 is 39.13"
# [1] "MSE for order 3 is 38.24"
# [1] "MSE for order 4 is 38.5"
# [1] "MSE for order 5 is 36.15"
# [1] "MSE for order 6 is 37.05"
# [1] "MSE for order 7 is 37.67"
# [1] "MSE for order 8 is 37.99"
# [1] "MSE for order 9 is 67.3"
# [1] "MSE for order 10 is 36.63"
# [1] "MSE for order 11 is 80.48"
# [1] "MSE for order 12 is 1898.7"
```

k-fold cross-validation

```r
set.seed(1)

## RM degree 1
glm_1 <- glm(MEDV ~ RM, data = housing)

# 10-fold cross-validation
cv_err_k10_1 <- cv.glm(housing, glm_1, K = 10)
cv_err_k10_1$K

# Cross-validation estimate of prediction error
cv_err_k10_1$delta[1]
```

```
# [1] 10
# [1] 44.49213
```

Estimate k-fold cross-validation for multiple polynomials

```r
for (j in 1:12) {
  glm_j <- glm(MEDV ~ poly(RM,j), data = housing)
  cv_err_j <- cv.glm(housing, glm_j, K=10)$delta[1]
  print(paste("MSE for order", j, "is", round(cv_err_j,2)))
}
```

```
# [1] "MSE for order 1 is 44.16"
# [1] "MSE for order 2 is 39.44"
# [1] "MSE for order 3 is 38.02"
# [1] "MSE for order 4 is 38.37"
# [1] "MSE for order 5 is 36.29"
# [1] "MSE for order 6 is 37.25"
# [1] "MSE for order 7 is 38.07"
# [1] "MSE for order 8 is 37.63"
# [1] "MSE for order 9 is 69.27"
# [1] "MSE for order 10 is 51.33"
# [1] "MSE for order 11 is 108.76"
# [1] "MSE for order 12 is 2319.54"
```

Compare validation set, LOOCV, and 10-fold CV

```
set.seed(2)
order <- 12
mse_vs    <- rep(0,order)
mse_loocv <- rep(0,order)
mse_k10   <- rep(0,order)

for (j in 1:order) {

  train <- sample(n_rows, n_rows/2)
  lm_vs <- lm(MEDV ~ poly(RM,j), data = housing, subset = train)
  mse_vs[j] <- mean((housing$MEDV - predict(lm_vs, housing))[-train]^2)

  lm <- glm(MEDV ~ poly(RM,j), data = housing)
  mse_loocv[j] <- cv.glm(housing, lm)$delta[1]
  mse_k10[j]   <- cv.glm(housing, lm, K = 10)$delta[1]
}
```

Table of comparisons

```
mat <- matrix(c(mse_vs, mse_loocv, mse_k10),
                    nrow = order, ncol = 3)
rownames(mat) <- paste("Degree", rep(1:order))
colnames(mat) <- c("Valid. Set","LOOCV","k-fold")
kable(mat, digits=1, align="c")
```

- Which polynomial achieves the lowest MSE?
- Do validation set, LOOCV, and k-fold agree?

	Valid. Set	LOOCV	k-fold
Degree 1	39.8	44.2	44.0
Degree 2	34.3	39.1	39.3
Degree 3	45.7	38.2	38.8
Degree 4	38.7	38.5	41.6
Degree 5	40.1	36.1	37.3
Degree 6	1984.8	37.0	36.5
Degree 7	39.0	37.7	38.1
Degree 8	32.5	38.0	38.1
Degree 9	3149.9	67.3	77.5
Degree 10	69.9	36.6	36.6
Degree 11	52.8	80.5	96.1
Degree 12	11003.4	1898.7	2269.8

Summary

- Model fit needs to be assessed using a test dataset

- This assessment is accomplished using cross-validation
  - Validation sets
  - Leave-one-out cross-validation
  - k-fold cross-validation

- k-fold cross-validation with 5 or 10 folds generally performs best

- Lab-10 due Sunday at 11:59pm
- Make sure to **stop your instance** when you are done working