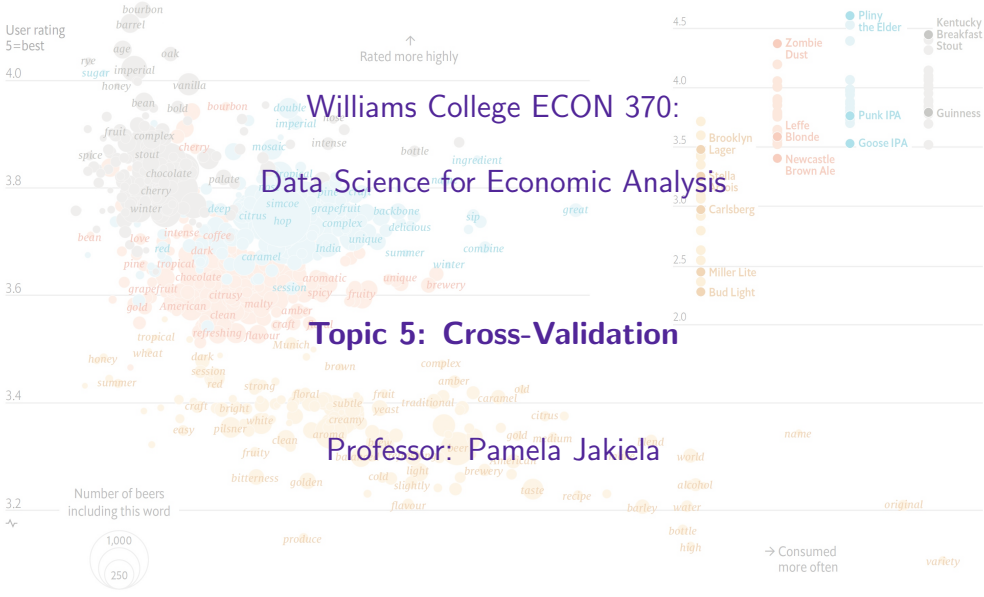


Williams College ECON 370:

Data Science for Economic Analysis

Topic 5: Cross-Validation

Professor: Pamela Jakiela

Outline

- What is over-fitting, and why is it a problem?
- Training data vs. test data and test mean squared error
- Using cross-validation to choose an OLS specification

Thought Experiment: 100 True Null Hypotheses

Let Y and $X_1, \ldots, X_{100}$ be column vectors containing $N = 20$ draws from iid standard normals.

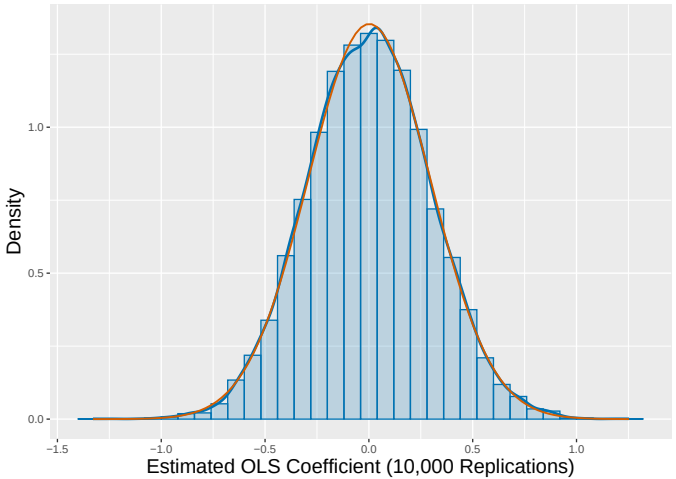
Suppose you run 100 regressions, regressing Y on each X_i variable (individually).

What is the expected value of $\hat{\beta}_{OLS}$?

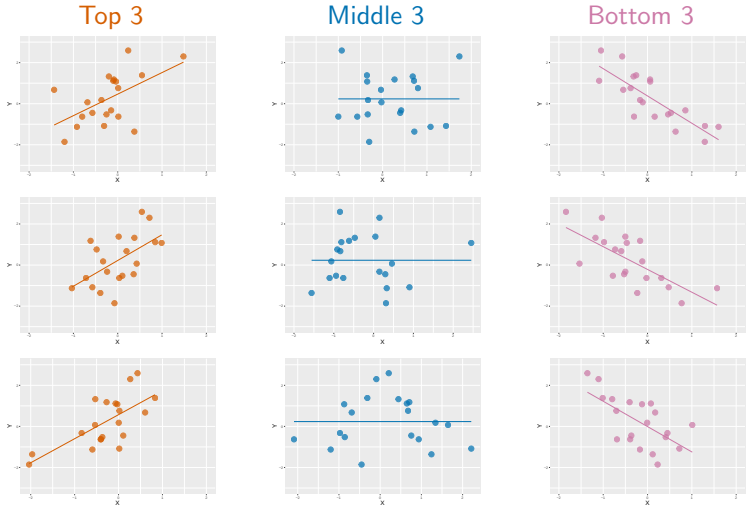
How many of the estimated OLS coefficients should we expect to be statistically significant?

What if we ran 10,000 regressions? How many significant X_i variables should we expect?

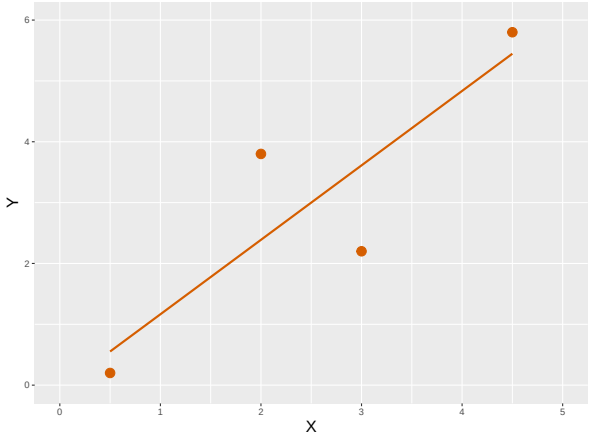
$\hat{\beta}_{OLS} \sim N(\beta_0, \tilde{\sigma}_{OLS})$: OLS Coefficients Are Normally Distributed About β_0



Strong Associations Occur by Chance (with 10,000 Replications)



OLS Is Fitting the Noise (and the True Model)



Over-Fitting

Why does it matter that OLS (or any other other statistical procedure) is fitting the noise?

- Inference: if you test a large number of hypotheses, some will be rejected (wrongly)
  - You may falsely conclude that two variables X and Y are correlated, or that policy X impacts outcome Y, or that outcome Y is different for two groups (e.g. men vs. women)
- Prediction: a less flexible model that ties our hands and prevents us from fitting the noise often provides better out-of-sample predictions than a more flexible (higher R^2) model
  - Using an overly complex or flexible model that ends up explaining both the underlying data-generating process and random error terms is often referred to as **over-fitting**

Over-fitting becomes a serious concern with many data science techniques that make use of very large data sets (including not just large Ns but many variables) and very flexible models

- (Aside about prediction vs. inference/causality and statistics/CS vs. economics here)

Another Thought Experiment: 100 True Null Hypotheses

Let Y and $X_1, \ldots, X_{100}$ be column vectors containing $N = 20$ draws from iid standard normals.

Split the data in half into Data Set 1 ($N = 10$) and Data Set 2 ($N = 10$).

Run 100 regressions, regressing Y on each X_i variable (individually) using only Data Set 1.

How many of the estimated OLS coefficients should we expect to be statistically significant?

How many of the X_i variables that are statistically significant predictors of Y in Data Set 1 will also be statistically significant predictors of Y in Data Set 2?

The Validation Set Approach to Specification Selection

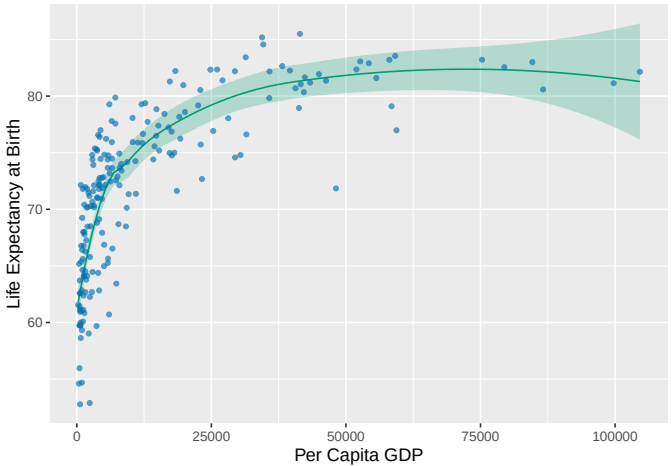
OLS minimizes the residual sum of squares (RSS) or, equivalently, the mean squared error (MSE), but if we care about prediction then we want to minimize the MSE in new data

- OLS will never tell you that you have included too many (non-collinear) variables

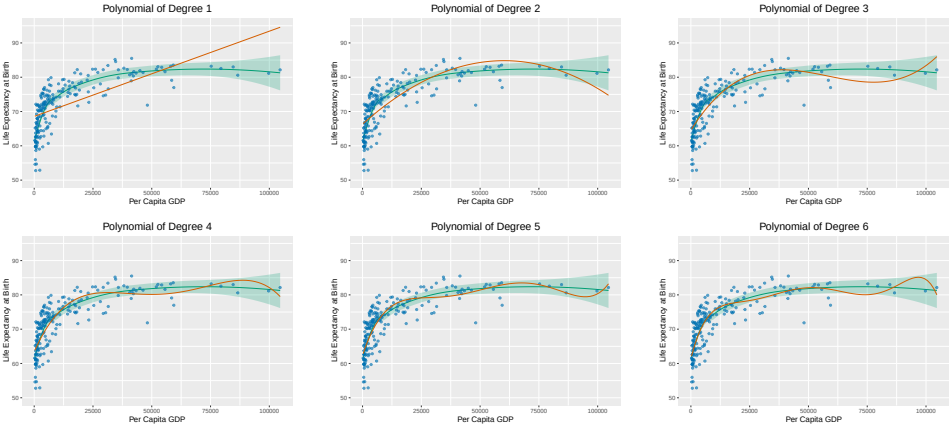
We don't always care about prediction, but when we do we need a strategy to avoid over-fitting

- One approach is to randomly partition a data into a **training** sample and a **test** sample
  - The training sample is used to choose or fit the model
  - The test sample is used to run the model, calculate the RSS or MSE
- The validation set approach builds on the intuition from the thought experiment: random noise in the training, test data sets is uncorrelated, so if we fit the model on the training data and then calculate the MSE on the test data, there is no (ok, much less) over-fitting

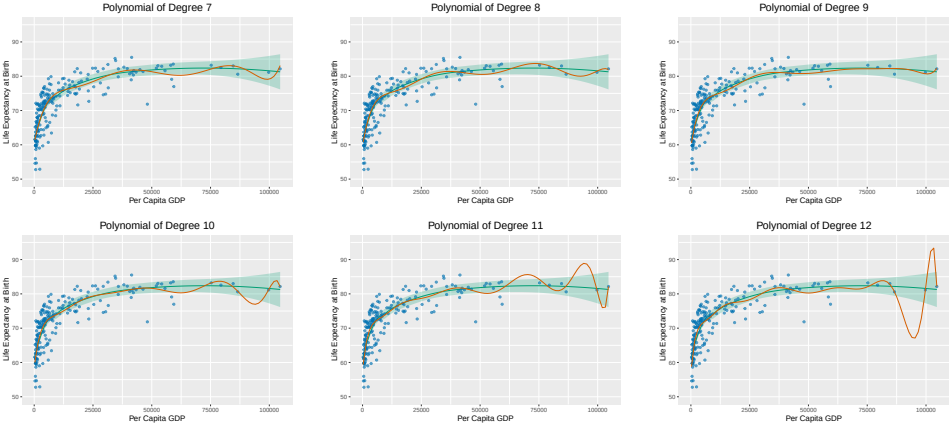
Example: The Preston Curve



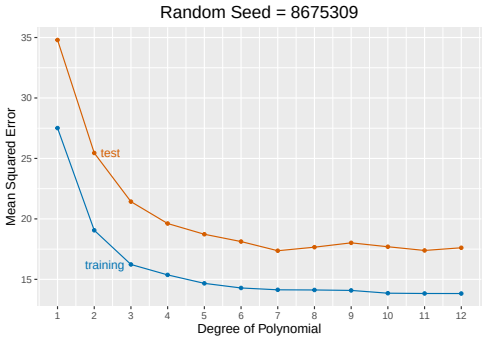
Over-Fitting the Preston Curve

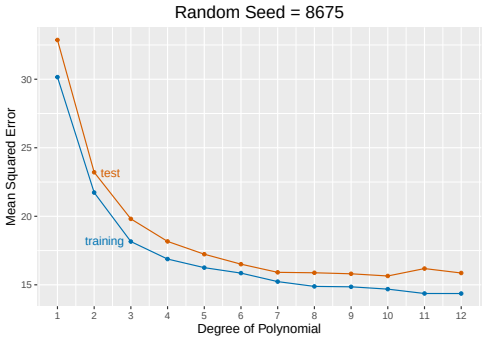


Over-Fitting the Preston Curve

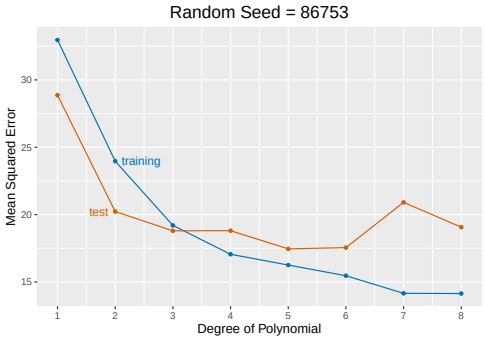


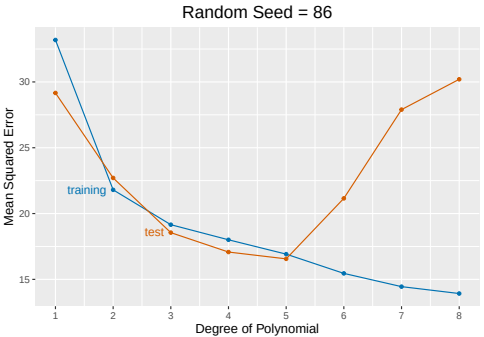
The Validation Set Approach in Practice



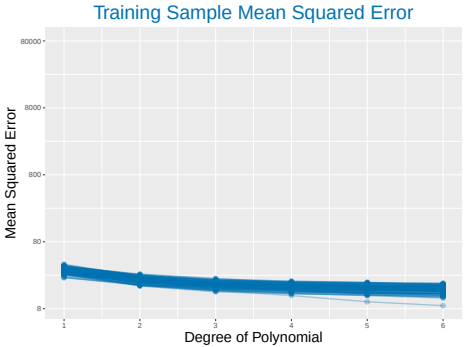


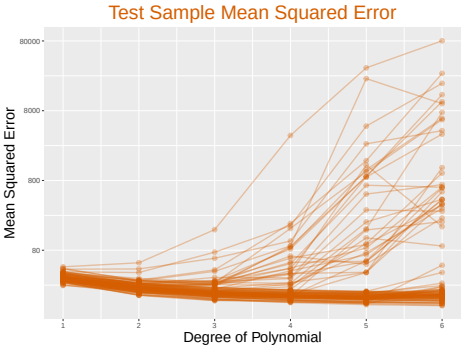
The Validation Set Approach in Practice



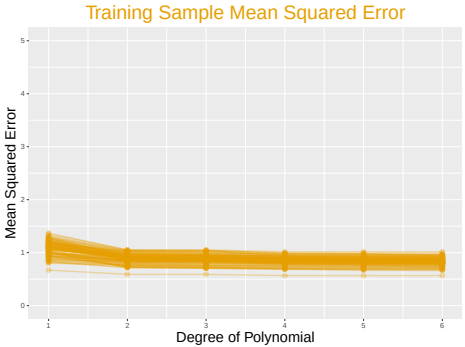


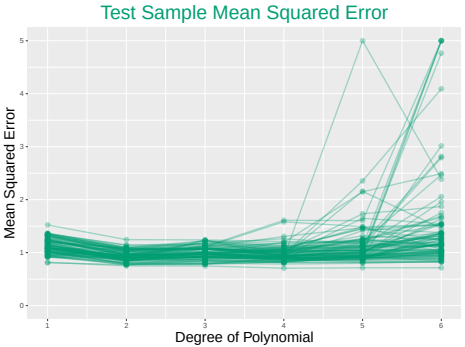
The Validation Set Approach in Practice: 100 Replications





Validation Set Approach Correct 43% of Time in Simulated Data*





data-generating process: $Y = 0.8X + 0.4X^2 + \varepsilon$ where $X \sim N(0,1)$, $\varepsilon \sim N(0,1)$, $N = 240$

*This is just one example, but the validation set approach is often noisy

Cross-Validation

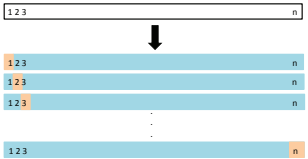
In **cross-validation**, we run the validation set approach multiple times and average the results

- Each observation only appears in the test data set once
- In each cut of the data, all the observations that are not test data are training data

Two types of cross-validation, defined by the way we construct the test data sets:

- Leave-One-Out Cross-Validation (LOOCV): each observation is a singleton test data set
- k-Fold Cross-Validation: partition the data into k **folds**, each a test data set

Leave-One-Out Cross-Validation

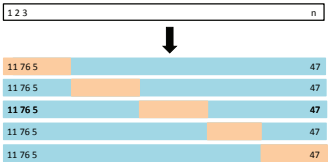


source: James et al. (2021)

Leave-One-Out Cross-Validation (LOOCV): each observation is a singleton test data set

- Run the validation set approach N times using all other observations as training data
- Calculate the squared prediction error for each observation, average to calculate MSE
- LOOCV may be computationally costly (when using models more complicated than OLS)

k-Fold Cross-Validation

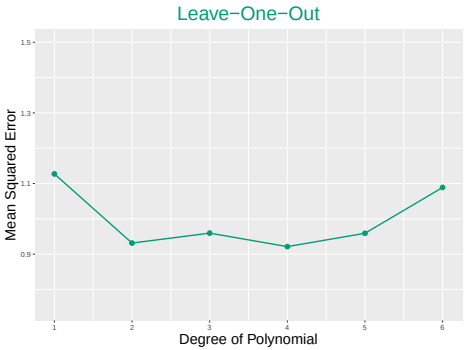


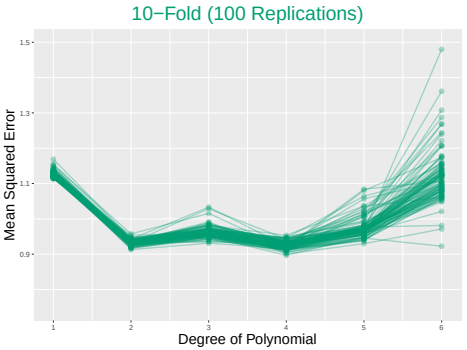
source: James et al. (2021)

k-Fold Cross-Validation:

- Randomly partition the sample into k distinct test data sets (aka folds)
- Run the validation set approach k times using the observations in fold k as the test data and the remainder of the data (i.e. everything but fold k) as the associated training data
- Calculate the mean squared prediction error in each old, average across folds

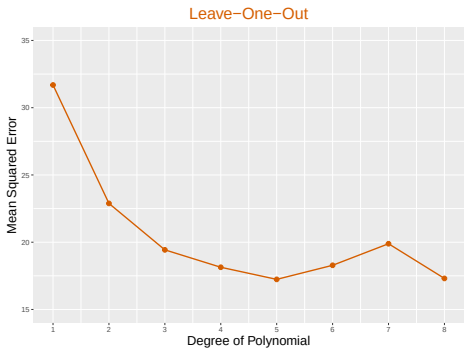
Comparing Test MSE: Leave-One-Out CV vs. k-Fold CV

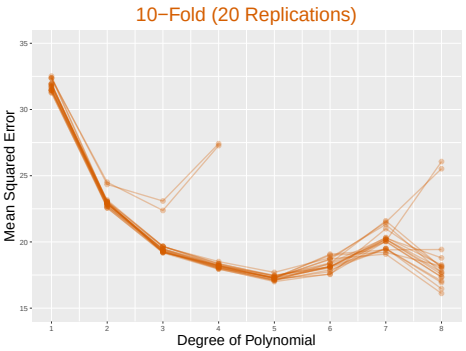




data-generating process: $Y = 0.8X + 0.4X^2 + \varepsilon$ where $X \sim N(0, 1)$, $\varepsilon \sim N(0, 1)$, $N = 240$

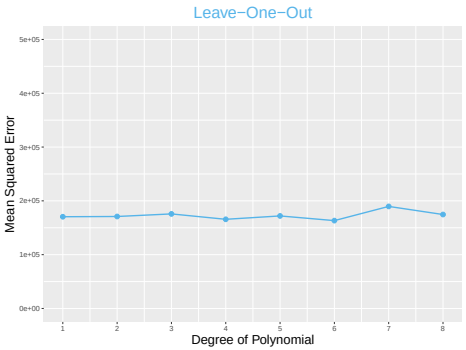
Comparing Test MSE: Leave-One-Out CV vs. k-Fold CV

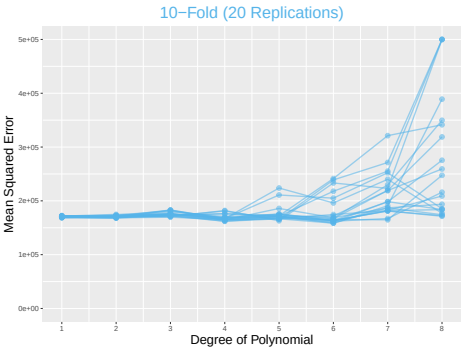




data source: World Bank WDI data on GDP per capita and life expectancy at birth

Comparing Test MSE: Leave-One-Out CV vs. k-Fold CV





data source: James et al. (2021) hitters data on runs and salary, a case where polynomial terms do not improve fit

Summary: Cross-Validation to Choose an OLS Specification

Cross-validation allows us to limit over-fitting when choosing a specification

- Randomize observations into k distinct groups (folds), make sure to set the seed
  - Skip this step in LOOCV
- For each of the k (in k-fold) or N (in LOOCV) partitions of the data into training vs. test:
  - Estimate each candidate specification j in the training data
  - Use the estimate coefficients to predict outcomes in test data in fold k
  - Calculate squared residuals, MSE in fold k for candidate specification j
- Calculate test MSE for specification j by averaging across folds (or test data sets)
- If graph of test MSEs has a flat bottom, tend toward most parsimonious specification

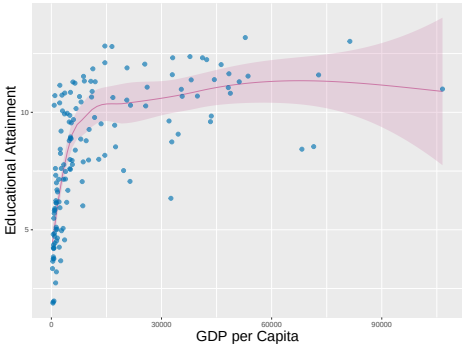
Lab #5

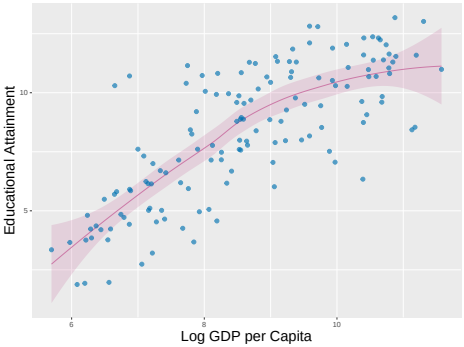
Objective: in this lab, you will implement k-fold cross-validation (with $k = 10$) using the data on GDP per capita and educational attainment from lab #1

You will find the polynomial function of log GDP that best predicts educational attainment

You will not use any packages or pre-built functions that implement cross-validation

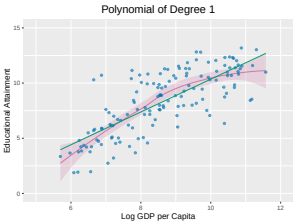
Lab #5

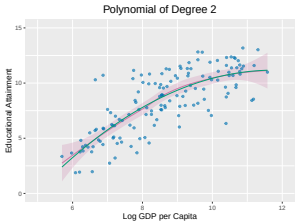


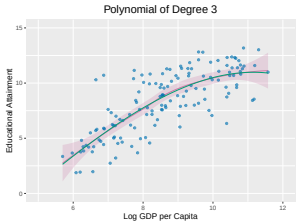


data source: merge data from lab #1 ($N = 143$), WDI data on GDP and Barro-Lee data on education

Lab #5







data source: merge data from lab #1 ($N = 143$), WDI data on GDP and Barro-Lee data on education

Lab #5: Outline of the Finished Product

1. Preliminaries: load the data from github, define key parameters and empty vectors to store output
2. Set the seed and split the data into `num_folds` randomly assigned folds
3. For each fold `n` in 1 to `num_folds`:
   - 3.1 Define the test (`fold == n`) and training (`fold != n`) data sets
   - 3.2 For each of the `max_order` possible polynomials of degree `i`:
     - 3.2.1 Fit the linear model
     - 3.2.2 Calculate the training data MSE
     - 3.2.3 Calculate the test data MSE
     - 3.2.4 Calculate a check MSE that that uses the formula from 3.2.3 on the data from 3.2.2
     - 3.2.5 Store your results in `train_mse`, `test_mse`, and `check_mse`
4. Average test MSE across folds and graph your results (test MSE as a function of polynomial degree)

Lab #5: Recommended Steps

1. Start with the content of the innermost loop: split the data into test and training samples, run OLS in the training sample, calculate the MSE in the training and test samples, check your results
2. Write a loop that does this for 1 up to `max_order` polynomial terms and store your results in empty `max_order` $\times$ 1 vectors `train_mse`, `test_mse`, and `check_mse` that you define
3. Wrap a second loop around your first loop that replicates the process for each of the folds

Hint: Setup

$\texttt{datasize} = 143$

$\texttt{max\_order} = 4$

$\texttt{num\_folds} = 3$

$$\texttt{fold\_id} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{bmatrix} \qquad \texttt{poly\_id} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} \qquad \texttt{test\_mse} = \begin{bmatrix} \\ \\ j \\ \\ \\ \end{bmatrix}$$

`j = max_order * (n - 1) + i`

where `n` is iteration of fold loop and

`i` is iteration of polynomial loop

Hint: Polynomial Regression

R:

```
ols <- lm(mean_edu ~ poly(log_gdp, 2, raw = TRUE), data = train_data)
   ...
test_xmat <- tibble(rep(1, length(test_data$mean_edu)), poly(test_data$log_gdp, 2, raw = TRUE))
```

Python:

```
from sklearn.preprocessing import PolynomialFeatures
   ...
poly = PolynomialFeatures(degree=2, include_bias=False)
X_train_poly = poly.fit_transform(train_data[['log_gdp']])
   ...
X_test_poly = poly.transform(test_data[['log_gdp']])
test_yhat = model.predict(sm.add_constant(X_test_poly))
```