Resampling

Class # 13

October 15, 2019

Announcements

- HW6 posted
- Reading for today: ISLR 5.1-5.2
- Reading for Thursday
  - ISLR 6.1-6.2
  - Metz (NYT 2019; see readme in github reading folder for this day)

Today's topic: Resampling

What you can do with resampling:

1. Model selection
2. Measuring the accuracy of parameter estimates
   - Parameters can be model coefficients
   - But they can also be other quantities you'd like to compute with the model.

When you'll (most likely) use which method:

- *Cross validation* for model selection.
  - This is a way to use your computer to do better than AIC.
- *Bootstrapping* for measuring parameter accuracy.
  - This is a way to use your computer to construct confidence intervals.

What you'll learn

- A few approaches to construct *test* mean square error estimates from your data

We have two terminology issues to deal with

First, “resampling”

- In ISLR, resampling means repeatedly pulling sub samples from your data.
  - If your data are a sample from the population, then we’re creating new samples from the original.
- But in pandas, resampling can also mean aggregating your data,
  - e.g. taking all data from a particular time window and averaging them.
  - We’ll see this in the notebook today.

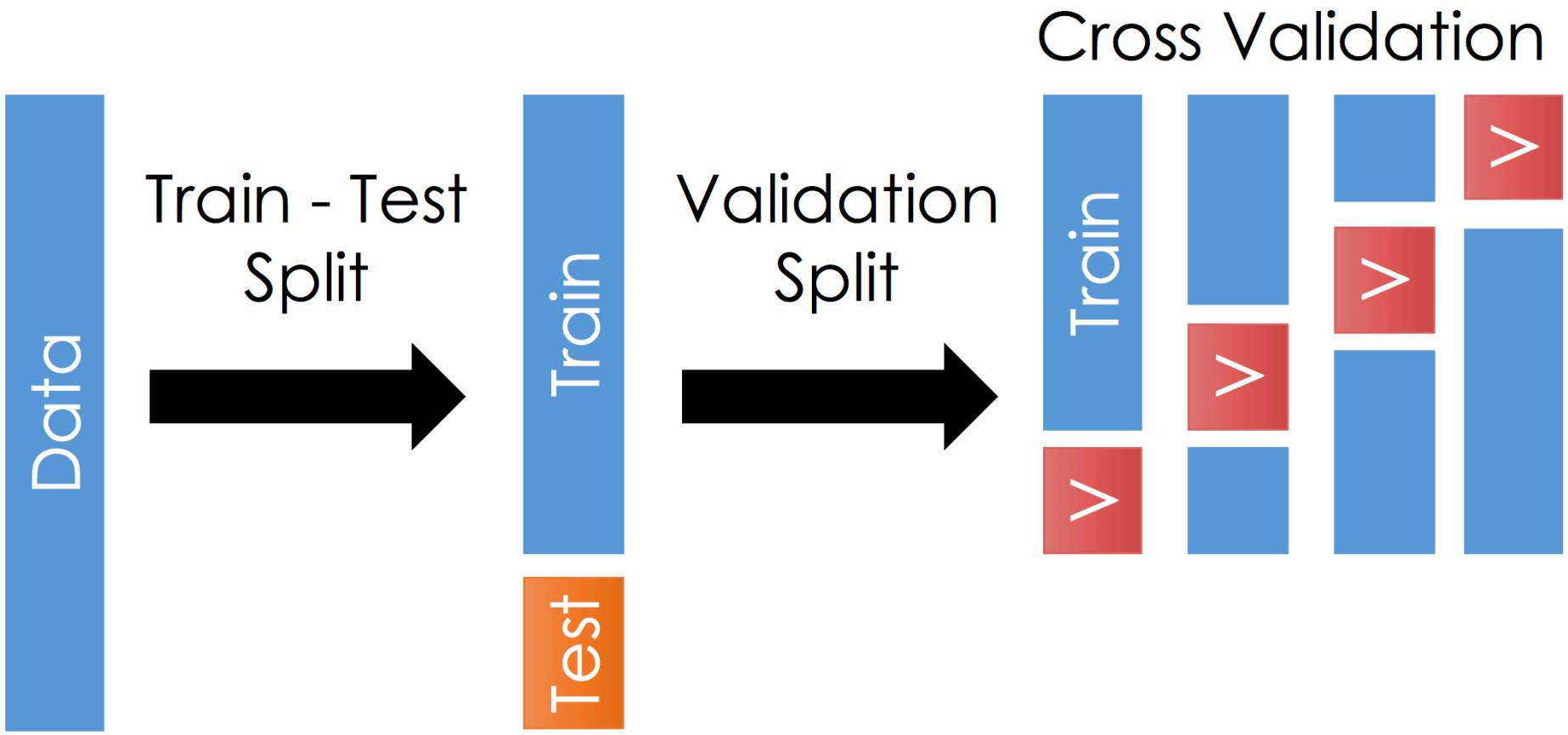
Second Terminology issue: train, validate, test

- **Training data**: Data used to fit a model's parameters
  - e.g. β values in a linear model
- **Validation data**: used to choose a model's hyperparameters.
  - We haven't learned about these yet, but we will soon.
  - For now: hyperparameters govern how flexible a model will be
  - They are critical for balancing the bias-variance tradeoff.
- **Testing data**
  - This is a fraction of data you withhold for final analysis
  - You DON'T use these data to fit any type of parameter
  - Compare test data error across modeling *strategies*, e.g. KNN vs OLS, choose best

Training, validation, testing…

- In cross-validation (what we're about to do)
  - Training data and validation data are pulled from the same block of data over and over again (more soon)
  - The testing data are only used for a single evaluation step.
- Ambiguity alert: "Test error" can refer to
  - The average error measured across all *validation* partitions of the data
  - The error obtained by evaluating your best possible model against the test data
  - **Take-home message: Test error need not refer to error obtained with the test data. Sigh!**
  - Instead it is any error other than training error.

Train, validate, test…



Let’s return to the task at hand…

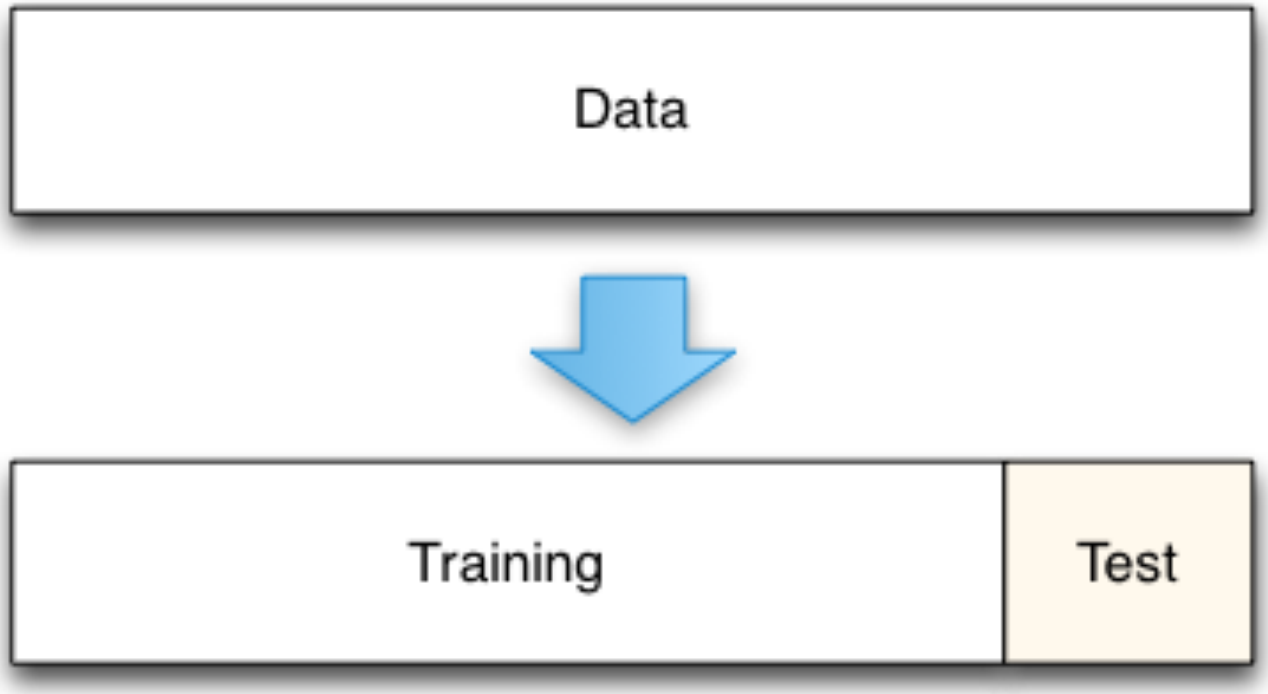
Model selection!

- So far we have mostly focused on model *identification,* or the process of choosing model parameters.
- But we also used AIC to do model *selection*, i.e. to learn which *type* of model we should use. For example:
  - How many features in my OLS model?
  - What size k for KNN?
- AIC works for OLS specifications.
- But what about KNN and all the other methods we’re about to learn?

What is an objective measure we can use for model selection, if AIC won't work for us?

- Hint: use data that the model has not been trained on
- Answer: mean squared *test* error, where test data were not used to train the model
- Why do this?
  - We avoid choosing a model that has *overfit* the data.
- Which is a symptom of overfit: bias or variance?
  - Variance!

Test and Train data

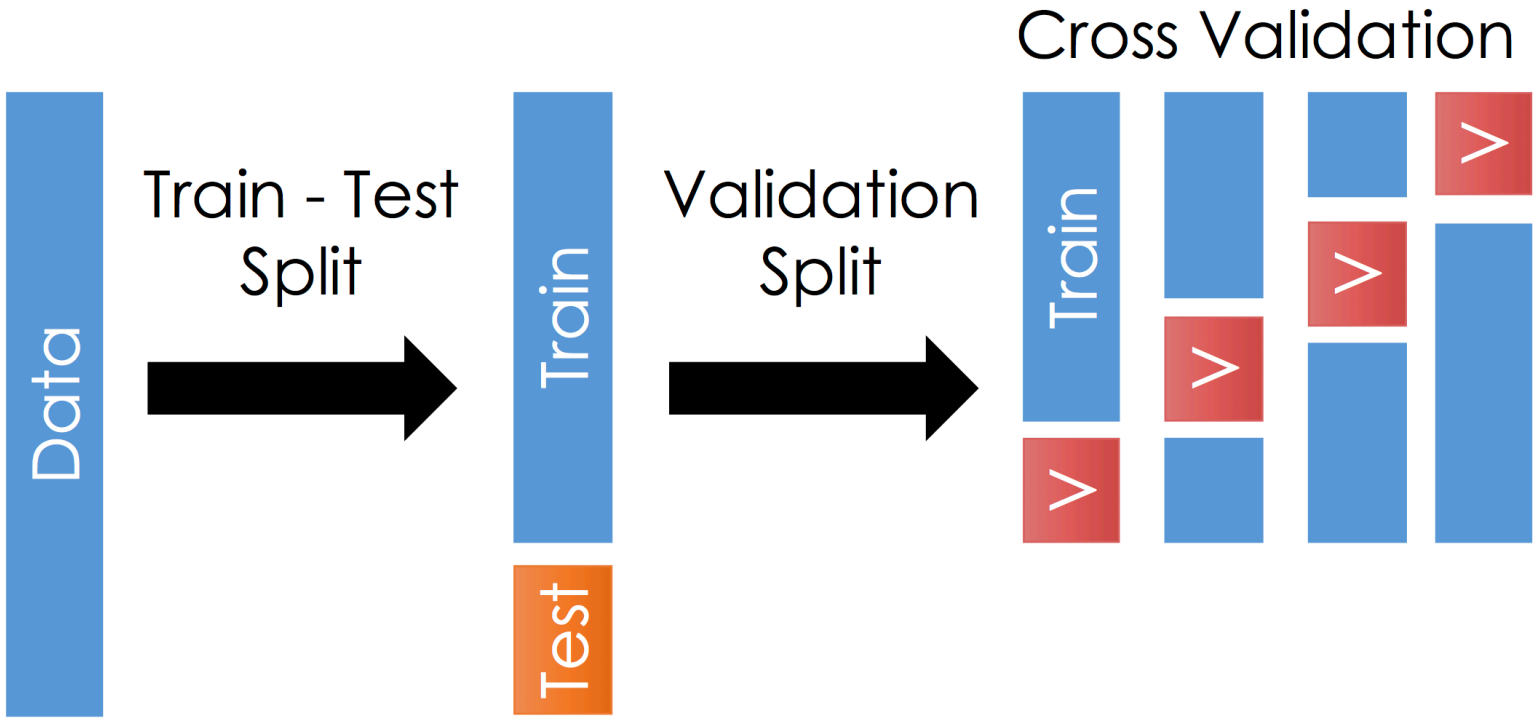


- If you just split the data once, you may get undesirable results:
  - Your error rate may depend strongly on the characteristics of the random split
  - You're only using a fraction of the data for validation.
  - You're only using a fraction of the data for training

Enter “cross validation”

Basic idea:

- Break your data up into groups. Give each group a chance to be the test data set, and use the remaining data for training

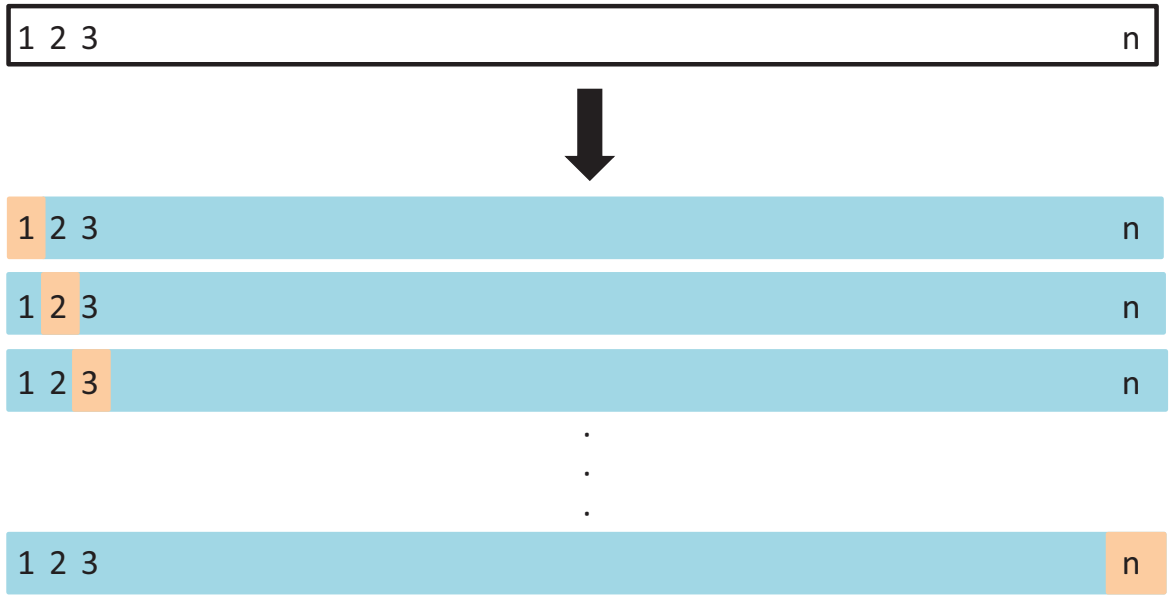


Two basic types of cross validation

- “Leave one out” cross validation
  - n observations means you’ll train n different models with (n-1) observations.
  - Compute the error for each withheld observation
  - We’ll talk about this first
- k-fold cross validation
  - k “folds” mean you train k < n different models.
  - Compute the error for each withheld “fold” of observations
  - We’ll talk about this second
- Remember, these are model selection strategies
  - They serve the same purpose as something like AIC
  - They work outside the confines of OLS

Leave One Out Cross validation…

- Allows you to use *all* the data for testing / validation
- Provides one estimate of the test error.
- The basic idea is to repeatedly split the data such that you “leave out” each observation once.
  - Use remaining observations to train a model
  - Use left-out observation to compute a test error
  - Do this *n* times!



Leave One Out Cross validation (LOOCV)

- For all *i*, pull the observation (x_i, y_i) out and fit the data to the remaining data.
  - Call the estimate of y_i from the "one-left-out" model $\hat{y}_i$
  - Then $MSE_i = (y_i - \hat{y}_i)^2$ (note this is an average of one value!)
- Then the error estimate for this "leave one out" process is

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i.$$

- …where the sum is over *all* n – i.e. n models were fit, each time fitting a different subset of the data.
- Note the subscript on CV is n because we split the data n times

LOOCV advantages to using just one split

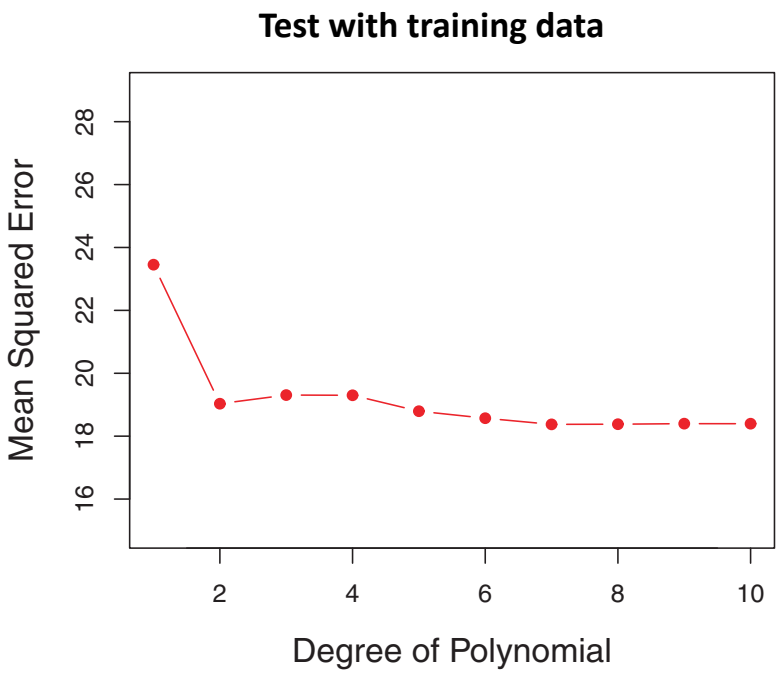
- Test error when you split just once tends to overestimate error more than LOOCV
  - This is because you use more data to fit each model with LOOCV
- By using every possible split, there is no variation in the estimate of error.
  - On the other hand, if you use just one validation set, the error depends strongly your random split.
- For some special cases you don’t even need to do the resampling
  - See Equation 5.2

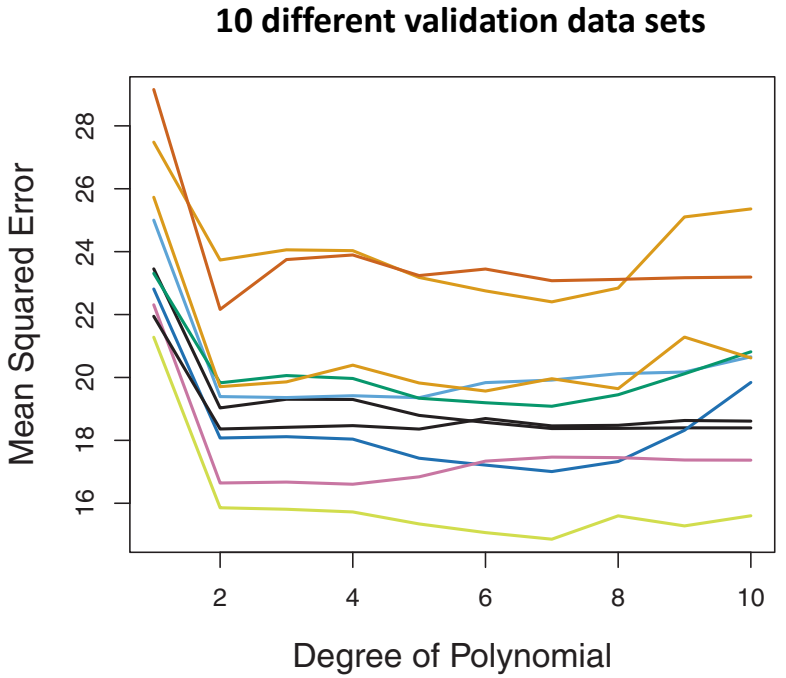
k-fold cross validation

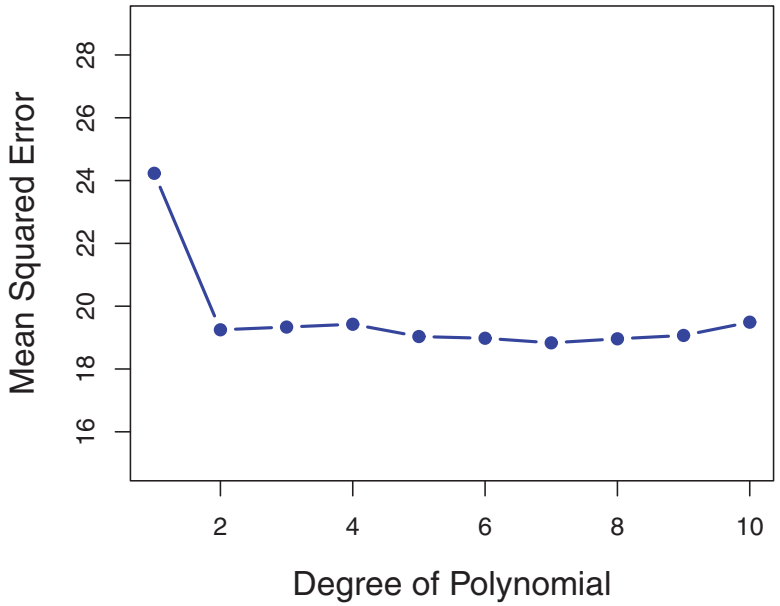
- *Randomly* divide the set into *k* equal groups, or "folds"
  - $k \leq n$
  - LOOCV is n-fold cross validation
  - Every observation gets assigned to a group to make sure we use all the data
- For each fold:
  - Withhold the fold and train a model with the remaining data
  - Compute MSE_i = the average of errors for *all* withheld observations in the fold.
- Then:

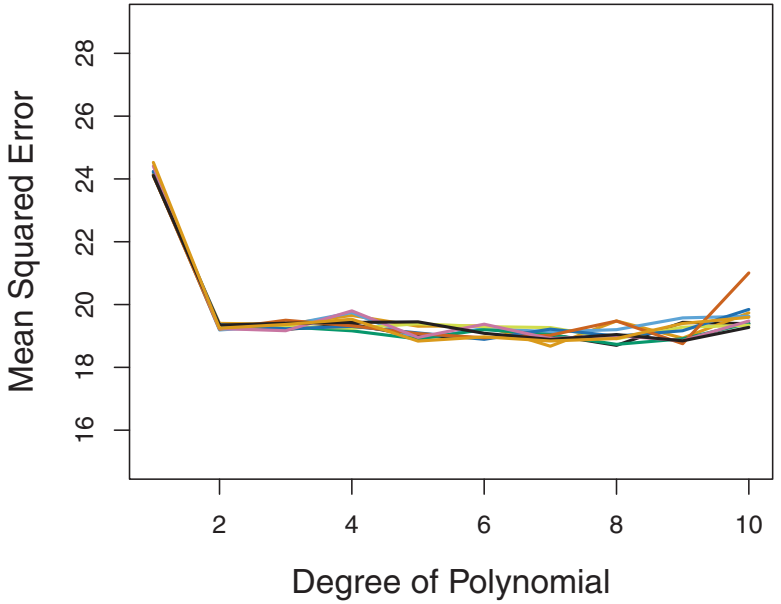
$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i$$

- Advantage of k-fold over LOOCV?
  - Only k folds (rather than n) – computationally easier









- Validation can be used for model selection.
- In these figures (Auto data set from the book) the minima are in different locations for different validation approaches
- 10-fold: each line is a different random split into 10 folds.

What should k be?

- Rule of thumb: k = 5 to 10.

Let’s look at today’s notebook.

The Bootstrap

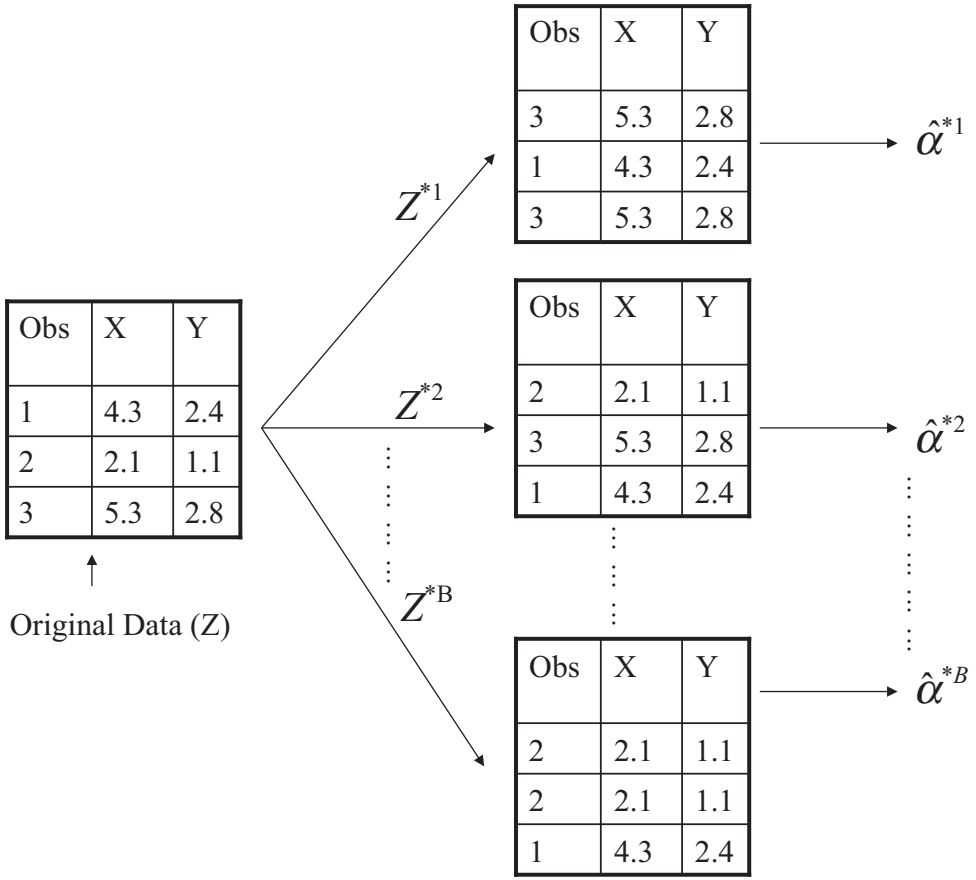
- Standard error -- what is it?
  - A measure of the average difference between an *estimate* of a parameter and the true value of the parameter.
- Standard errors can be computed with simple formulae in some cases, for example for coefficients fit using ordinary least squares (OLS) regression
- However for many other cases there is no simple formula
  - When the assumptions required to apply the formulae don't apply, even if you're doing OLS
    - For example if your errors are strongly correlated
  - When you're using a different algorithm to fit your model, for example linear discriminant analysis.

Computing the standard error numerically

- If you can't directly apply a formula to compute the standard error, you might be able to use your computer to construct the distribution
- You could draw new *sets* of samples from your population repeatedly and recompute the parameter of interest
  - For example draw a set of N = 100 observations and repeat B = 1,000 times
  - The average parameter estimate across the B draws will be the true parameter, and you can measure the standard error from the distribution of parameter estimates
- But! We can't usually repeatedly sample from the true population
  - You can't go out and re-survey a new set of 100 randomly chosen 1,000 times (or else you'll never graduate!)

Enter the bootstrap

- First key idea: create a new sample from your original sample by sampling *with replacement*.
- Repeat this B times
- Record the parameters you care about each time
- The average parameter estimate will equal the true parameter
- Second key idea: the standard error of the parameter estimate *will* be roughly the true standard error.



- Say it again: the standard error of the bootstrapped parameter estimates is a decent approximation of true standard error.

In other words…

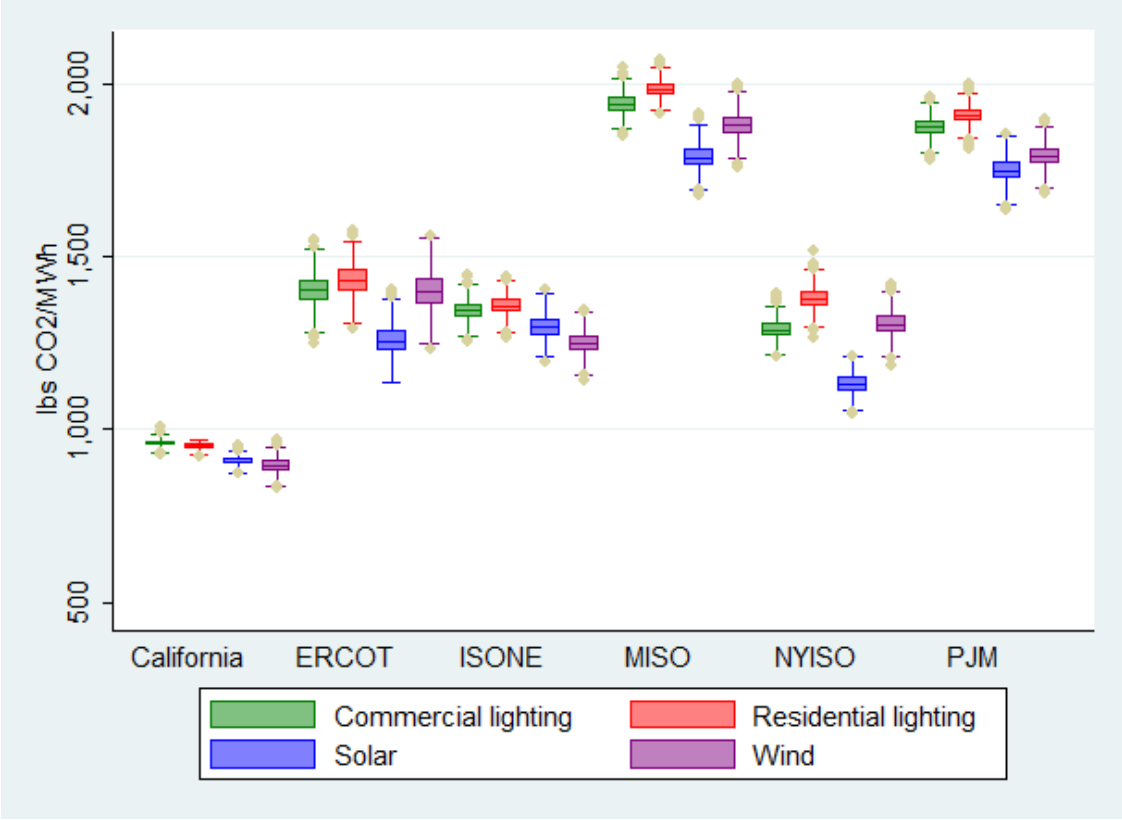
- “The population is to the sample as the sample is to the bootstrapped sample”

How many in the sample? How many samples?

- N = (Rule of thumb) Make your samples equal in size to the original sample
- B = (Good practice) keep generating new samples until your estimates (parameter average and standard error) converge

What can I bootstrap?

- This is one of the advantages of bootstrapping: anything you might want to calculate from your data, you can bootstrap.
- You don't need to limit yourself to regression coefficients
- Example: bootstrapped estimates of CO2 avoided by implementing various technologies in different locations (Callaway, Fowlie and McCormick, JAERE 2017)



Summary

- Cross validation is a neat way to create a test error without ignoring parts of your data
  - But note, by using the cross validation error to choose a hyperparameter (like polynomial order, or k in KNN) the final model does “see” all the data, though indirectly
- So to compare different kinds of models:
  - We will withhold a “test” data set and use this at the very end
  - There is no getting around the problem that we can’t use all data to train the model
- Bootstrapping is a method to construct standard errors for parameters
  - Less relevant for machine learning, but a close sibling to cross-validation
  - Key difference: sample *with* replacement to simulate a data collection process in which samples are randomly collected