Statistical Learning: Chapter 5
Resampling methods (Cross-validation and bootstrap)
(Note: prior to these notes, we'll discuss a modification of an earlier train/test experiment from Ch 2)

We discuss 2 resampling methods in this chapter
- cross-validation
- the bootstrap

These methods refit a model of interest to samples formed from the training set, to get additional information about the fitted model.
- Test set prediction error (from CV)
- Standard error of our parameter estimates (from the bootstrap)

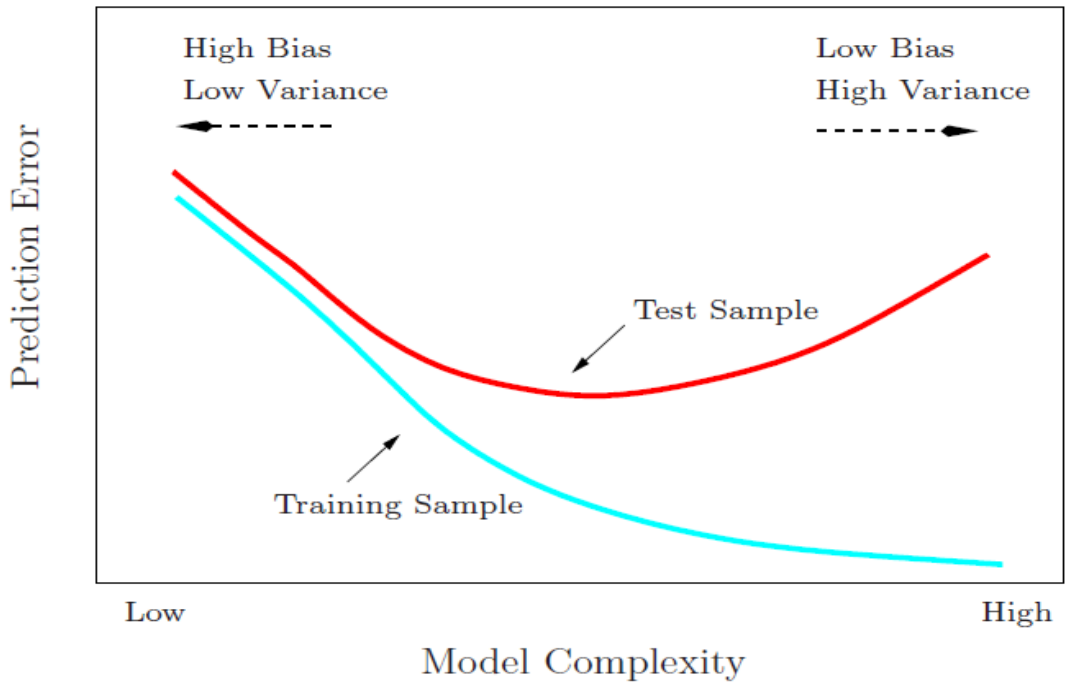
5.1 Cross-validation

Recall the distinction between training and test error:

- Test error is the average (or expected) error that results from using a statistical learning method to predict a response on a new observation, one not used in training the model.

- The average is over all test points and all corresponding responses.

- Training error is the in-sample error, that is the error in predicting a point that was used to train the model.
- We know the training error and test error rates can be quite different, with training error being much smaller in some cases.



Estimating the test error.
- Test error is defined as an average error* over all test X and over all Y you might observe at those X
- The "test error" is a quantity we cannot know exactly.
- So we try to estimate it. In Ch 6 you'll see some adjustments motivated by theory (Cp, AIC, BIC)

* this "average" is with respect to some distribution on X and Y, which is unknown.

- In an application, the best estimate usually comes from having a large test set.
  - In assignment 2, we have this for the HAR data: a set of 20,000 observations not used for training.

(aside: Given that the entire HAR dataset is studying the motion patterns of just 4 people, it is reasonable to question exactly what population of outcomes is repreresented by this particular test set, or the whole sample of data from which it was selected)

- In many real problems we don't have enough data to set aside a large test set.

This section discusses several alternatives.

Possibility 1: Divide data into training and validation sets

FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

Note: In Fig 5.1 and others in this chapter, the observations are listed from left to right. This is the opposite of the usual convention that the rows of a matrix correspond to observations, but it make the figures easier to understand.

Train on "training set" , predict on the "validation set".

Prediction errors (e.g., MSE in regression, misclass in classification) are an estimate of the test error.

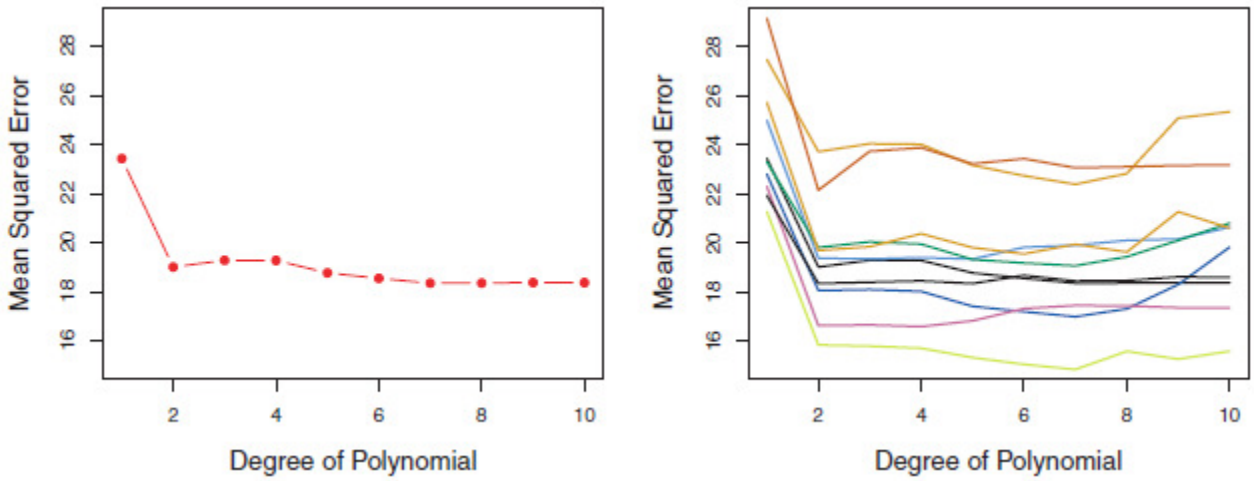
Footnote: When I first read this, my reaction was "Huh?" (i.e., confusion). Isn't what they are now calling the "validation set" just what we've been calling a test set previously?

- I think they are very similar. Careful inspection of Ch 5. reveals that "test set" is only used in this chapter to refer to "test set error", the theoretical quantity arising from an infinite set of data. Any time the book talks about dividing the available data into parts, the part we don't use for training is called the "validation" set.

Example: Predicting "mpg" using polynomial functions of "horsepower" in the Auto dataset.

Left: validation error for one split data into train and validate

Right: 10 different validation error curves for 10 different train/validate splits



Drawbacks of the validation set approach:
- Validation error can be quite a variable estimate of test error, depending on the (random) train/validate split.
- In validation approach, only a subset (e.g. half) of the data are used to train the model.
- That is, the model won't be as well-estimated as if we had used all the data.
- So the model is likely to predict worse on the validation set than if we had more data.
- That is, test error is likely ***overestimated*** by a validation set approach.

Possibility 2: K-fold cross-validation
(why does K get used for everything? This has nothing to do with the other K's)

Strategy (with 5-fold CV for illustration):

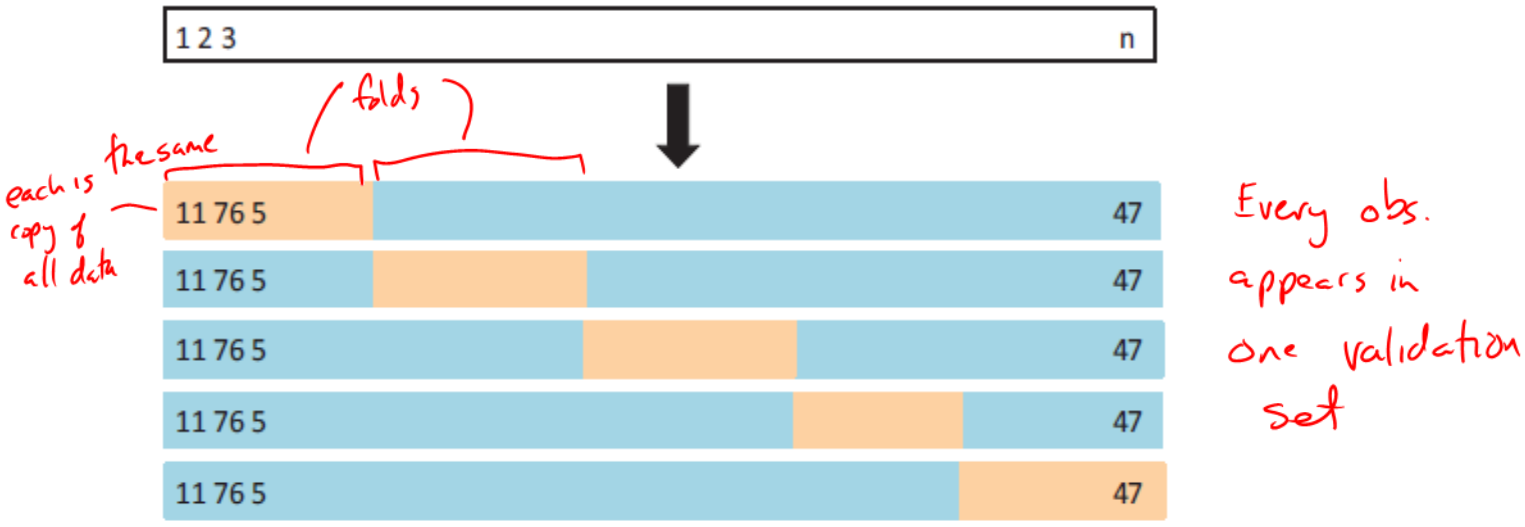


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

K-fold CV is a widely used method to estimate test error.

As the graphic above shows, we split the data into K parts, train on K-1 of them, and predict for the one "held out" part. This process is repeated K times, each time predicting a different one of the K groups or "folds".

Error estimate is an average of K error measures on each validation-set.

This (mostly) solves the problem of training our model with only half of the full dataset (seen above in validation set approach). Now we're using the data fraction (K-1)/K = 1-1/K to train (80% in 5-fold).

It comes with computational cost.
- We're fitting K (e.g. 5) models instead of 1.

And there still could be some bias in our test set estimate from slightly-smaller training sets.

Details of calculating error (for MSE)
- K folds, with the j-th fold having observations "fold j"
- fold j gives mean squared error

$$MSE_j = \frac{1}{n_j} \sum_{i \in \text{fold } j} (y_i - \hat{y}_i^{(j)})^2$$

n_j: $|\text{fold } j|$ = # obs. in fold j

$\hat{y}_i^{(j)}$: prediction for obs. i from model trained without fold j

(overall)

$$MSE = \sum_{j=1}^{K} \frac{n_j}{n} MSE_j = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i^{(j(i))})^2$$

← aggregating over each obs. individually

$$= \sum_{j=1}^{K} \frac{n/K}{n} MSE_j$$

if all folds of equal size, i.e. $\frac{n}{K} = n_j$

$$= \sum_{j=1}^{K} \frac{MSE_j}{K}$$

averaging K MSE_j's

↑ average of the MSE's for each validation set

Aside: viewed this way, as an average of K "samples" $MSE_1, \ldots, MSE_K$, we might also consider standard errors.

Leave-one-out CV (LOOCV)

OK, so if we're worried that each training set is still slightly smaller than the "full" data available, what do we do?

- Increase K. We're training on 1-1/K observations in each case.
- The logical limit? K = n, so we train on n-1 observations and validate on the other one.
- This is called leave one out CV (LOOCV).
- Computationally expensive.

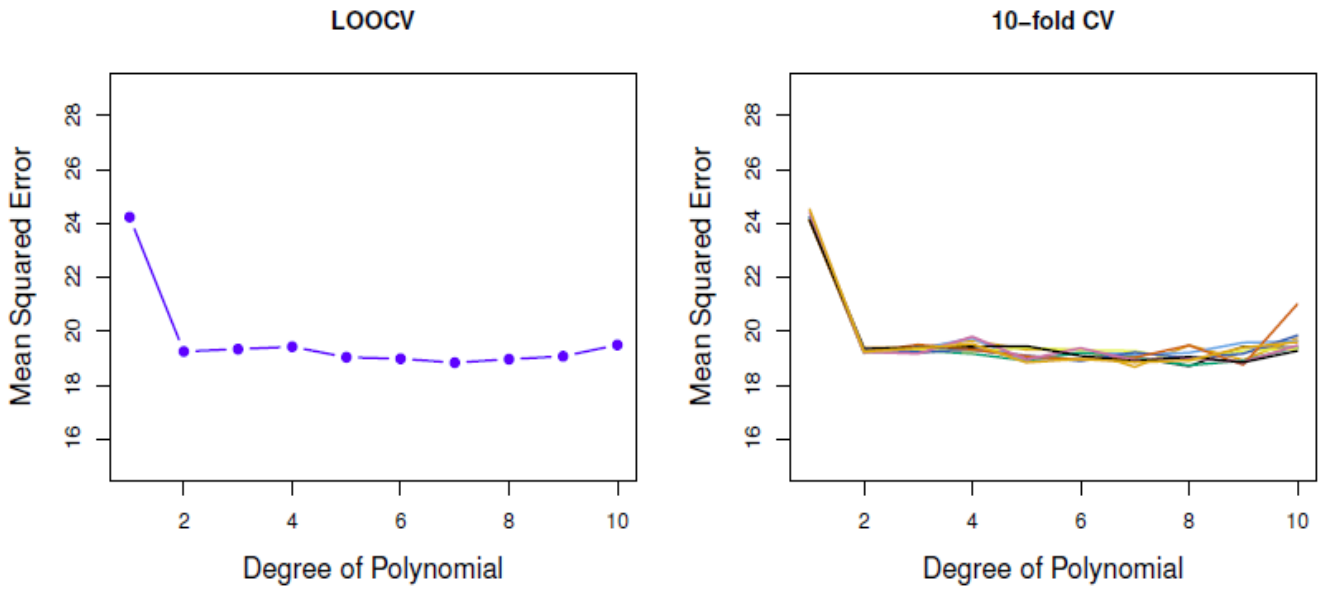
In the case of least squares linear regression, there is a computational shortcut that allows calculation of the LOOCV error with a ***single estimated model using the full dataset***.

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

h_i = leverage, see book (does not depend on y!)

Without the h_i, it would be the training set MSE. So the formula is reweighting the training set MSE!

Recommendation:
In most cases, LOOCV is not worth it
- computationally expensive (except in linear regression)
- training on n-1 observations produces very similar models for each "fold"
- 5-fold or 10-fold CV "shakes up" the data more while training on nearly as many observations as the full dataset.



In Auto example (predict MPG = poly(horesepower)) , both LOOCV and 10-fold CV (do similar.
Note that 10-fold CV has much less variation across different random splits (9 shown on right for illustration) than did the validation set earlier.

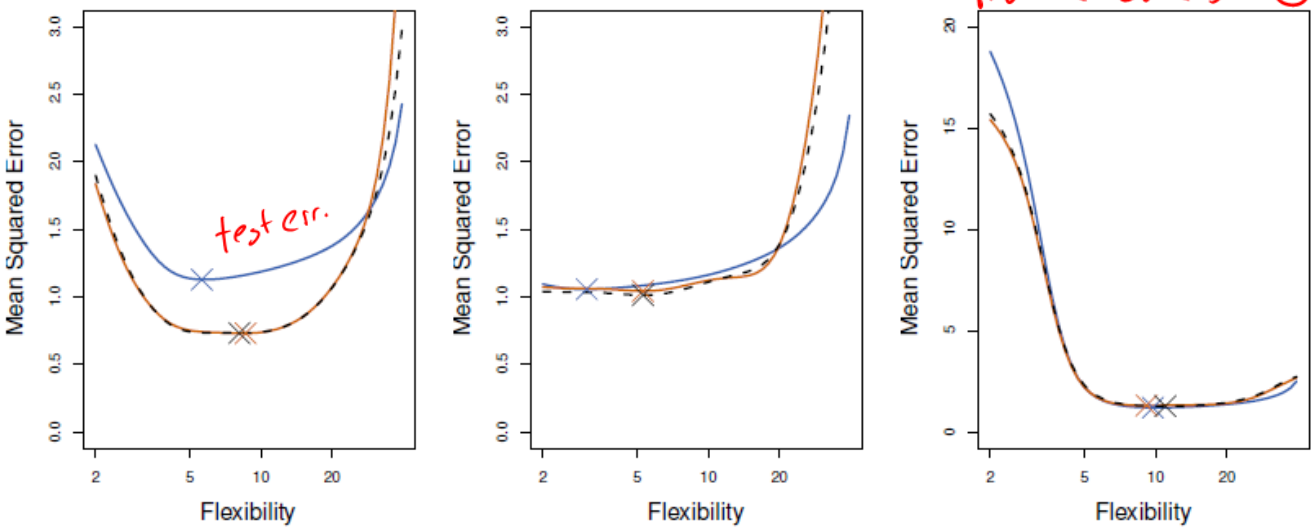
Revisiting the 3 one-dimensional examples from Ch 2
(blue = true MSE, black dash = LOOCV MSE, orange = 10-fold CV)

Both methods can get the error estimate wrong.

Note however, the "Flexibility" parameter minimizing the LOOCV or 10-fold CV error is pretty close to the one minimizing the "true" MSE.

So cross-validation methods provide estimates of the test error. These estimates have bias and variability, but they are invaluable to select a suitable level of model complexity.

↘ the give a good estimate of the bottom of the true (blue) "∪"



Last step: Refit model with "optimal" flexibility using 100% of data.

This lets us make future predictions with greater accuracy.

getting standard errors

The Bootstrap (5.2)

We can think of the bootstrap as a general-purpose tool for quantifying the uncertainty in a statistical model.

We've already seen that in the case of linear (and logistic) regression, theory gives us standard errors for estimates and predictions.

But for complex estimators, it can be hard to get these standard errors.

The bootstrap is an easy computational tool to get the standard errors of estimators.

The book starts with an example before explaining how the bootstrap exactly works.

Investment: 2 assets with (random) returns X and Y.

To minimize risk (= variance of return) of a portfolio of αX and $(1-\alpha)Y$, the best choice of α is

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_X\sigma_Y}$$

$\sigma_{XY} = \mathrm{Cov}(X,Y)$

$\sigma_X^2 = \mathrm{Var}(X)$, $\sigma_Y^2 = \mathrm{Var}(Y)$

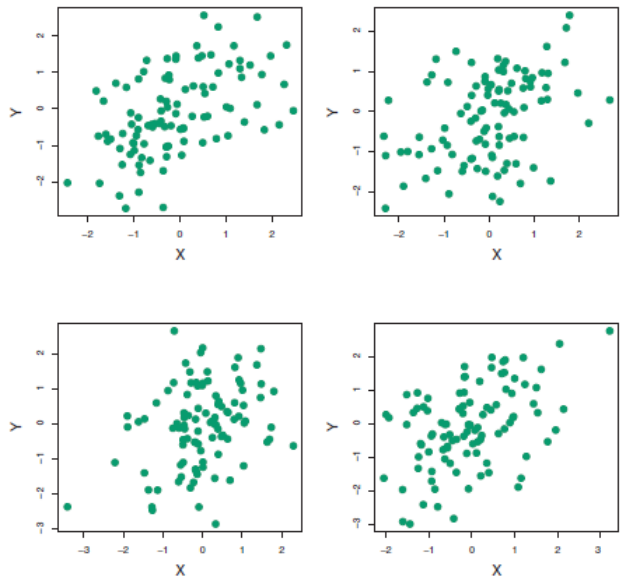
$0 \le \alpha \le 1$

α = % of $ invested in "X"

But we don't know $\sigma_X^2, \sigma_Y^2, \sigma_{XY}$. So we plug in estimates $\hat{\sigma}_X^2, \hat{\sigma}_Y^2, \hat{\sigma}_{XY}$ to get $\hat{\alpha}$.

So what's the sampling distribution (or even just the standard error) of alpha-hat?

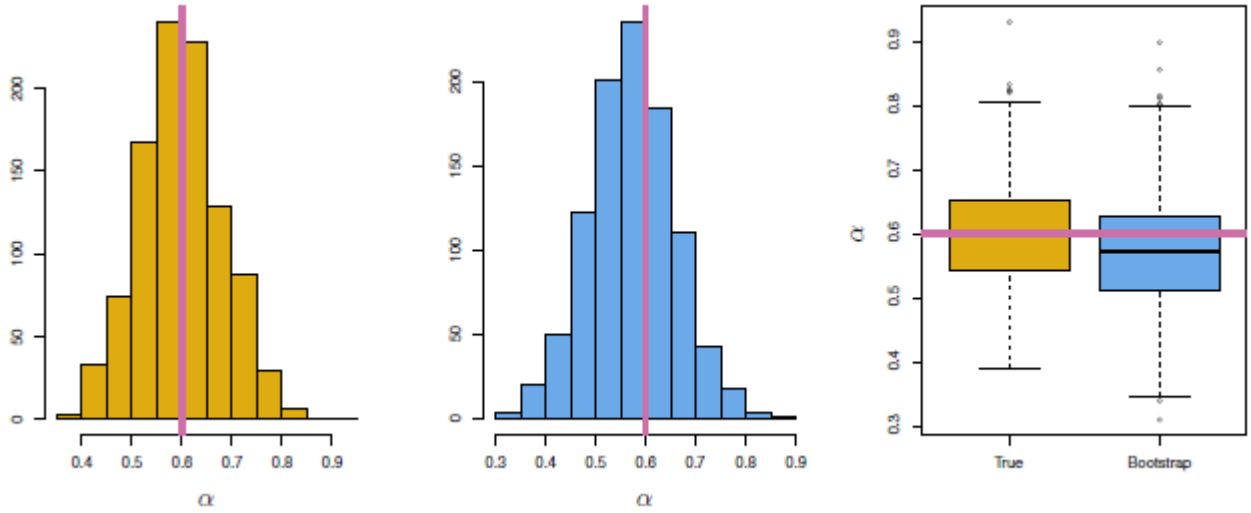
We'll "play god" for a minute, and simulate 1000 different datasets of 100 (x,y) return pairs per dataset.
- Each dataset gives estimates of variances and covariances, and thus an estimate of alpha
- We can examine the distribution of these 1000 different estimates, and figure out the standard error of our estimator.



Left plots are 4 of the 1,000 realizations. Each gives an alpha estimate (0.576, 0.532, 0.657 and 0.651 from upper left to lower right).

The left panel below shows the distribution of all 1000 estimated alphas. The sample mean of the 1000 alphas is 0.5996, very close to 0.6 which is the optimal value for the population. The sample standard deviation of the estimates is 0.083.

$\alpha = 0.6$ is obtained using pop values for σ_X, σ_Y σ_{XY}

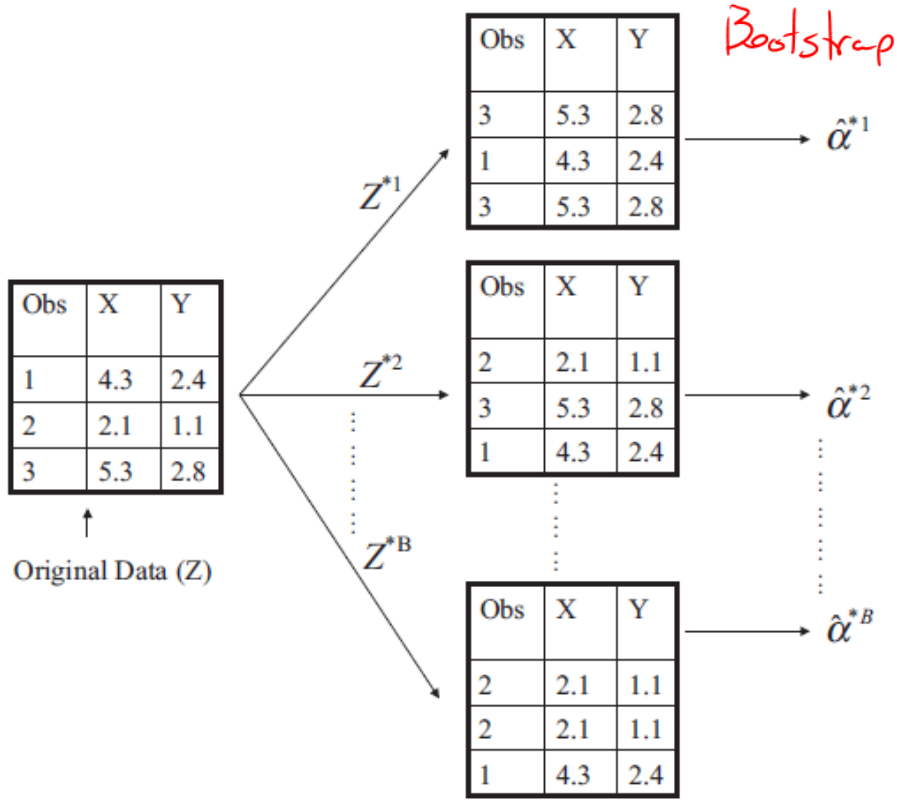


Thus far, in this example, we have not been realistic (we "played god", simulating 1000 different datasets).

Now, we take a single simulated dataset and use the bootstrap in it. The middle plot shows the bootstrap estimate of the distribution of alpha-hat. The boxplots at right compare them as well. They're very similar.

OK so this looks promising. How does the bootstrap work?
- The key idea is to create many new "bootstrap samples".
- Each bootstrap sample a dataset of the same number of observations as the original training data. This is done by ***sampling from the training data with replacement.***
- This mimicks the process by which a sample is generated from a population. So the bootstrap is a surrogate for the random selection of samples from a population.

next page: simple example for a very small dataset (3 observations)



Bootstrap sample #1

Previous page (middle panel blue hist)

B= 1000 bootstrap samples

FIGURE 5.11. *A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α.*

A general remark about bootstrapping and cross-validation:

Ideally, we should enclose inside a bootstrap or a CV loop **all** parts of our analysis that "see" the response Y in a supervised learning problem.