ADEC 7430:
Big Data Econometrics

Resampling Methods in Machine Learning

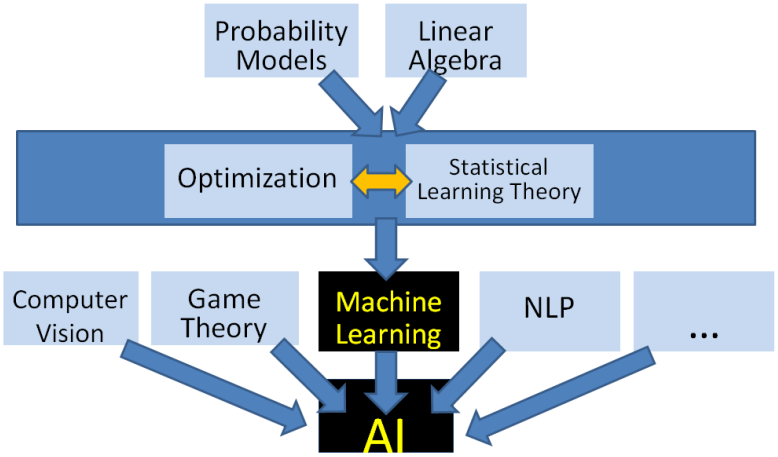
Dr. Nathan Bastian

Woods College of Advancing Studies

Boston College

Assignment

- **Reading:** Ch. 5
- **Study:** Lecture Slides, Lecture Videos
- **Activity:** Quiz 3, R Lab 3, Discussion #3, Individual Project #1



Lesson Goals

- Distinguish the basic concepts of resampling methods for machine learning model selection and assessment.
- Describe the similarities and differences between the validation set approach, leave-one-out cross-validation, and *K*-fold cross-validation.
- Explain the basic theoretical concepts of the bootstrap method for assessing statistical accuracy.
- Recognize that the performance of machine learning models depends on their prediction capability on independent test data.

Resampling Methods

- Involves repeatedly drawing samples from a **training set** and refitting a model of interest on each sample in order to obtain more information about the fitted model.

- Example: We can estimate the variability of a linear regression fit by repeatedly drawing different samples from the training data, fitting a OLS regression to each new sample, and then examining the extent to which the resulting fits differ.

- *Model Assessment:* having chosen a final model, estimating its prediction error on new data.
- *Model Selection:* estimating the performance of different models in order to choose the best one.

Resampling Methods (cont.)

- *Cross-Validation*
  - Used to estimate **test set prediction error** rates associated with a given machine learning method to evaluate its performance, or to select the appropriate level of model flexibility.

- *Bootstrap*
  - Used most commonly to provide a **measure of accuracy** of a parameter estimate or of a given machine learning method.

Model Assessment

- The *generalization* performance of a machine learning method relates to its prediction capability on independent test sets.
- Assessment of this performance is extremely important in practice, since it guides the choice of the machine learning method or model.
- Further, this gives us a measure of the quality of the ultimately chosen model.

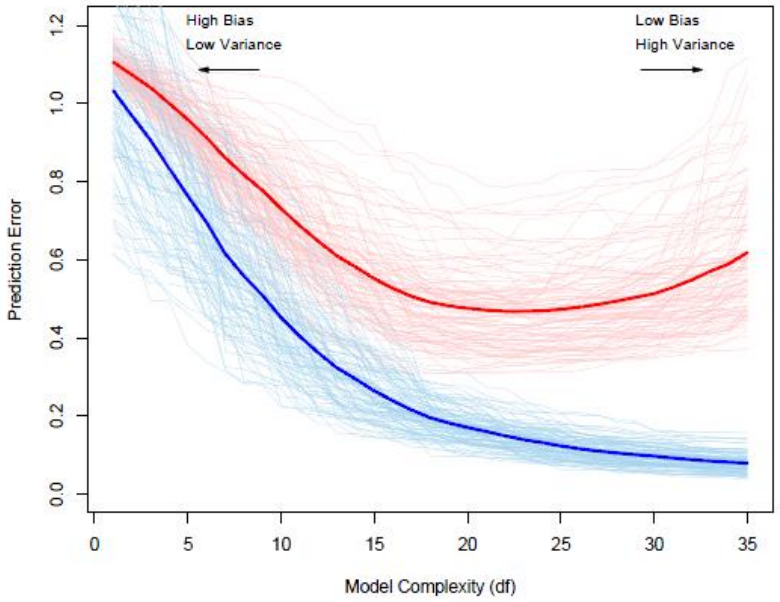
Model Assessment (cont.)

- *Test Error*
  - The average error that results from using a machine learning method to predict the response on a new observation.
  - The prediction error over an independent test sample.

- *Training Error*
  - The average loss over the training sample: $\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$

- **Note:** The training error rate can dramatically *underestimate* the test error rate

Model Assessment (cont.)



- As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures.
- Hence, there is a decrease in bias but an increase in variance.
- However, training error is not a good estimate of the test error.

- Training error consistently decreases with model complexity.
- A model with zero training error is overfit to the training data and will typically generalize poorly.

Model Assessment (cont.)

- If we are in a data-rich situation, the best approach for both *model selection* and *model assessment* is to randomly divide the dataset into **three parts**: training set, validation set, and test set.

- The *training set* is used to fit the models. The *validation set* is used to estimate prediction error for model selection. The *test set* is used for assessment of the prediction error of the final chosen model.

- A typical split might by 50% for training, and 25% each for validation and testing.

Model Assessment (cont.)

- **Best solution:** use a large designated test set, which is often not available. For the methods presented here, there is insufficient data to split it into three parts.

- There are some methods to make mathematical adjustments to the training error rate in order to estimate the test error rate:
  - Cp statistic, AIC, BIC (we will discuss these next week)

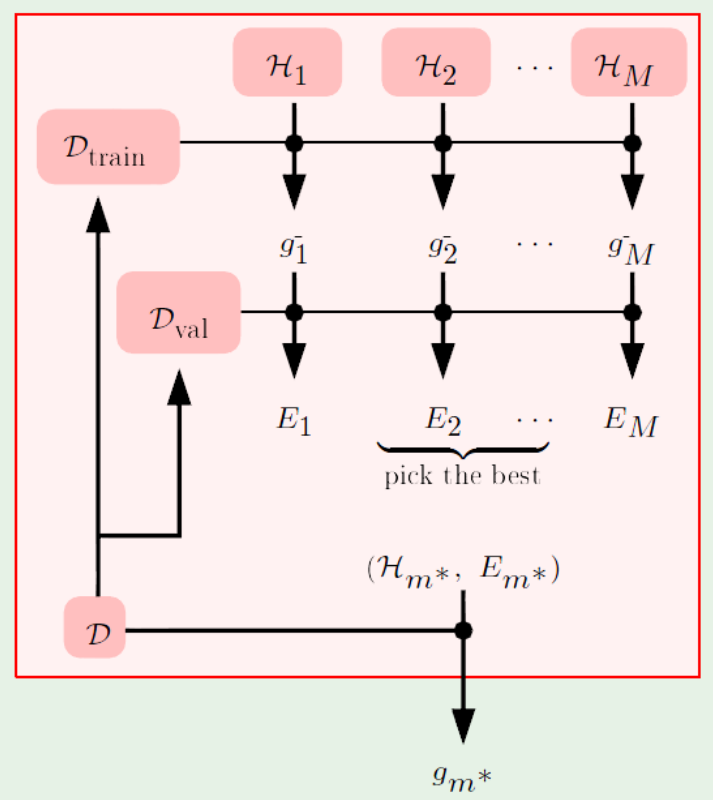
- Here, we consider **cross-validation (CV) methods** that estimate the test error by *holding out* a subset of the training observations from the fitting process, and then applying the machine learning method to those held out observations.

Overview: Model Selection

- By far, the most important use of validation is for *model selection*, which we will discuss in greater detail next week.
- This could be the choice between a linear model and a nonlinear model, the choice of the order of polynomial in a model, the choice of a regularization parameter, or any other choice that affects the learning process.
- In almost every learning situation, there are some choices to be made and we need a principled way of making these choices.
- The leap is to realize that *validation* can be used to estimate the out-of-sample error for more than one model.

Overview: Model Selection (cont.)

- Suppose we have *M* models; validation can be used to select one of these models.
- We use the training data to fit the model, and we evaluate each model on the validation set to obtain the validation errors.
- It is now a simple matter to select the model with the **lowest validation error**.

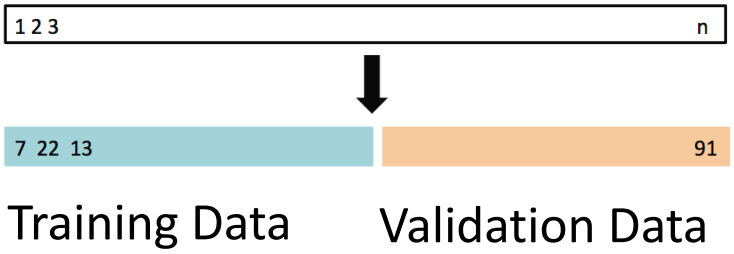


Validation Set Approach

- Suppose that we would like to find a set of variables that give the lowest *validation error rate* (an estimate of the *test error rate*).

- If we have a large data set, we can achieve this goal by randomly splitting the data into separate training and validation data sets.

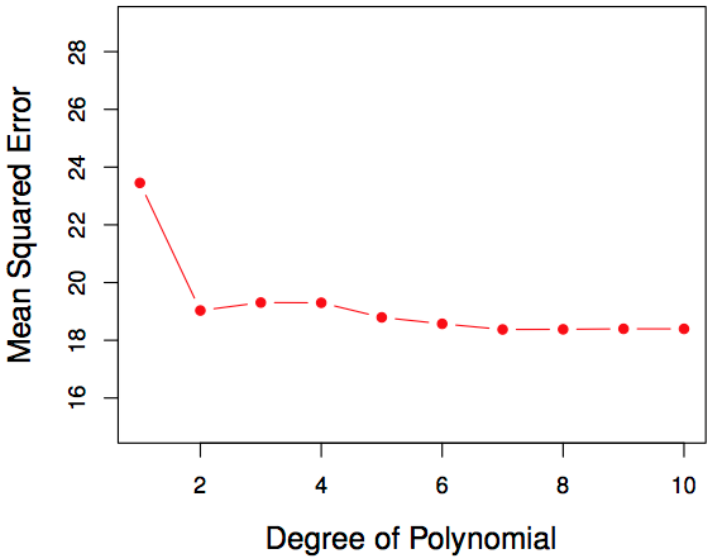
- Then, we use the training data set to build each possible model and select the model that gives the lowest error rate when applied to the validation data set.

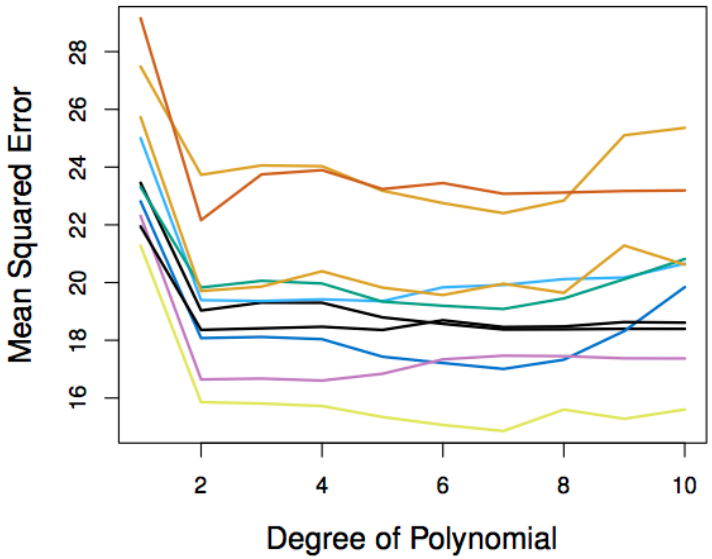


Validation Set Approach: Example

- **Example:** we want to predict *mpg* from *horsepower*
- Two models:
  - mpg ~ horsepower
  - mpg ~ horsepower + horspower^2
- Which model gives a better fit?
  - We randomly split (50/50) *392 observations* into training and validation data sets, and we fit both models using the training data.
  - Next, we evaluate both models using the validation data set.
  - **Winner** = model with the *lowest* validation (testing) MSE

Validation Set Approach: Example Results





- **Left Panel:** Validation error estimates for a single split into training and validation data sets.
- **Right Panel:** Validation error estimates for multiple splits; shows the test error rate is highly variable.

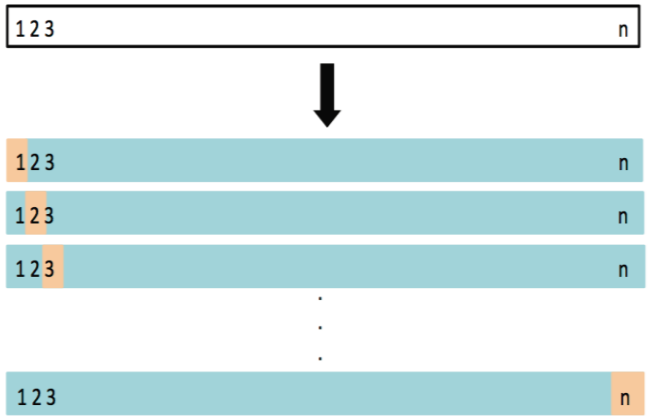
Validation Set Approach: Review

- **Advantages:**
  - Conceptually simple and easy implementation.

- **Drawbacks:**
  - The validation set error rate (MSE) can be highly variable.
  - Only a subset of the observations (those in the training set) are used to fit the model.
  - Machine learning methods tend to perform worse when trained on fewer observations.
  - Thus, the validation set error rate may tend to *overestimate* the test error rate for the model fit on the entire data set.

Leave-One-Out Cross-Validation

- Instead of creating two subsets of comparable size, a single observation is used for the validation set and the remaining observations ($n - 1$) make up the training set.



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

- **LOOCV Algorithm:**
  - Split the entire data set of size *n* into:
    - Blue = training data set
    - Beige = validation data set
  - Fit the model using the training data set
  - Evaluate the model using validation set and compute the corresponding MSE.
  - Repeat this process *n* times, producing *n* squared errors. The average of these *n* squared errors estimates the test MSE.

Validation Set Approach vs. LOOCV

- LOOCV has far less bias and, therefore, tends not to overestimate the test error rate.

- Performing LOOCV multiple times always yields the same results because there is no randomness in the training/validation set splits.

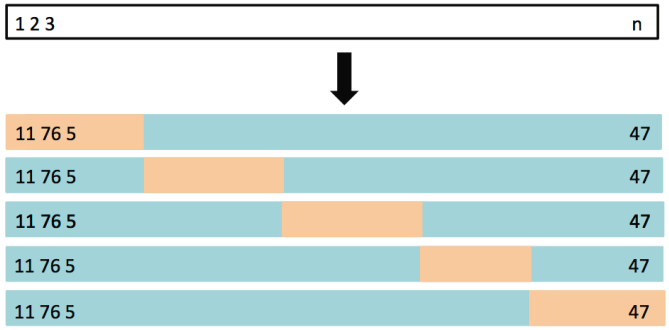
- LOOCV is computationally intensive because the model has to be fit *n* times. However, there is a shortcut with OLS linear or polynomial regression (where h_i is the leverage):

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

K-Fold Cross-Validation

- Probably the simplest and most widely used method for estimating prediction error.
- This method directly estimates the average prediction error when the machine learning method is applied to an independent test sample.
- Ideally, if we had enough data, we would set aside a validation set (as previously described) and use it to assess the performance of our prediction model.
- To finesse the problem, *K*-fold cross-validation uses part of the available data to fit the model, and a different part to test it.

K-Fold Cross-Validation (cont.)



- We use this method because LOOCV is computationally intensive.
- We randomly divide the data set of into *K* folds (typically *K* = 5 or 10).

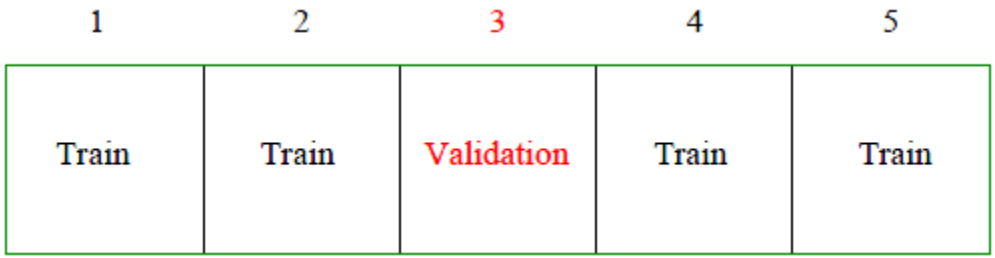
- The first fold is treated as a validation set, and the method is fit on the remaining *K* – 1 folds. The MSE is computed on the observations in the *held-out* fold. The process is repeated *K* times, taking out a different part each time.
- By averaging the *K* estimates of the test error, we get an estimated validation (test) error rate for new observations.

K-Fold Cross-Validation (cont.)

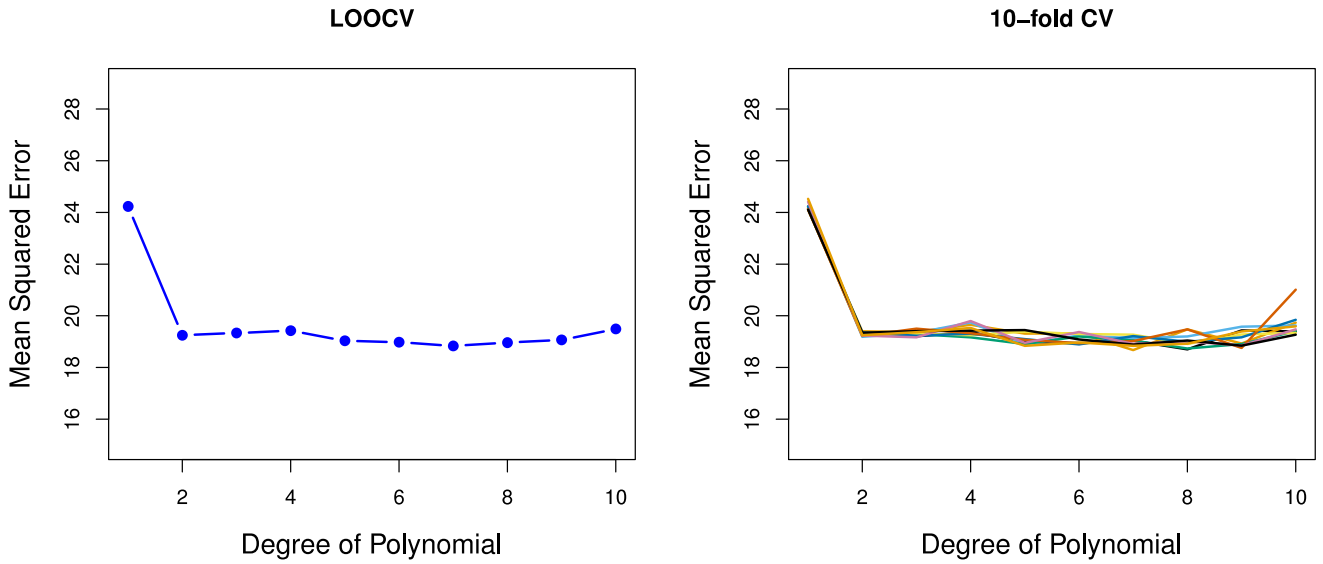
- Let the *K* folds be $C_1, \ldots, C_K$, where C_k denotes the indices of the observations in fold *k*. There are n_k observations in fold *k*: if *N* is a multiple of *K*, then $n_k = n / K$.

- Compute: $\mathrm{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \mathrm{MSE}_k$

where $\mathrm{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (Y_i - \hat{Y}_i)^2$ and $\hat{Y}_i$ is the fitted value for observation *i*, obtained from the data with fold *k* removed.



K-Fold Cross-Validation vs. LOOCV



- **Left Panel:** LOOCV Error Curve
- **Right Panel:** *10*-fold CV run nine separate times, each with a different random split of the data into ten parts.
- **Note:** LOOCV is a special case of *K*-fold, where $K = n$

Bias-Variance Trade-off for *K*-Fold Cross-Validation

- Which is better, LOOCV or *K*-fold CV?
  - LOOCV is more computationally intensive than *K*-fold CV
  - From the perspective of bias reduction, LOOCV is preferred to *K*-fold CV (when $K < n$)
  - However, LOOCV has higher variance than *K*-fold CV (when $K < n$)
  - Thus, we see the bias-variance trade-off between the two resampling methods

- We tend to use *K*-fold CV with $\boldsymbol{K = 5}$ **or** $\boldsymbol{K = 10}$, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance

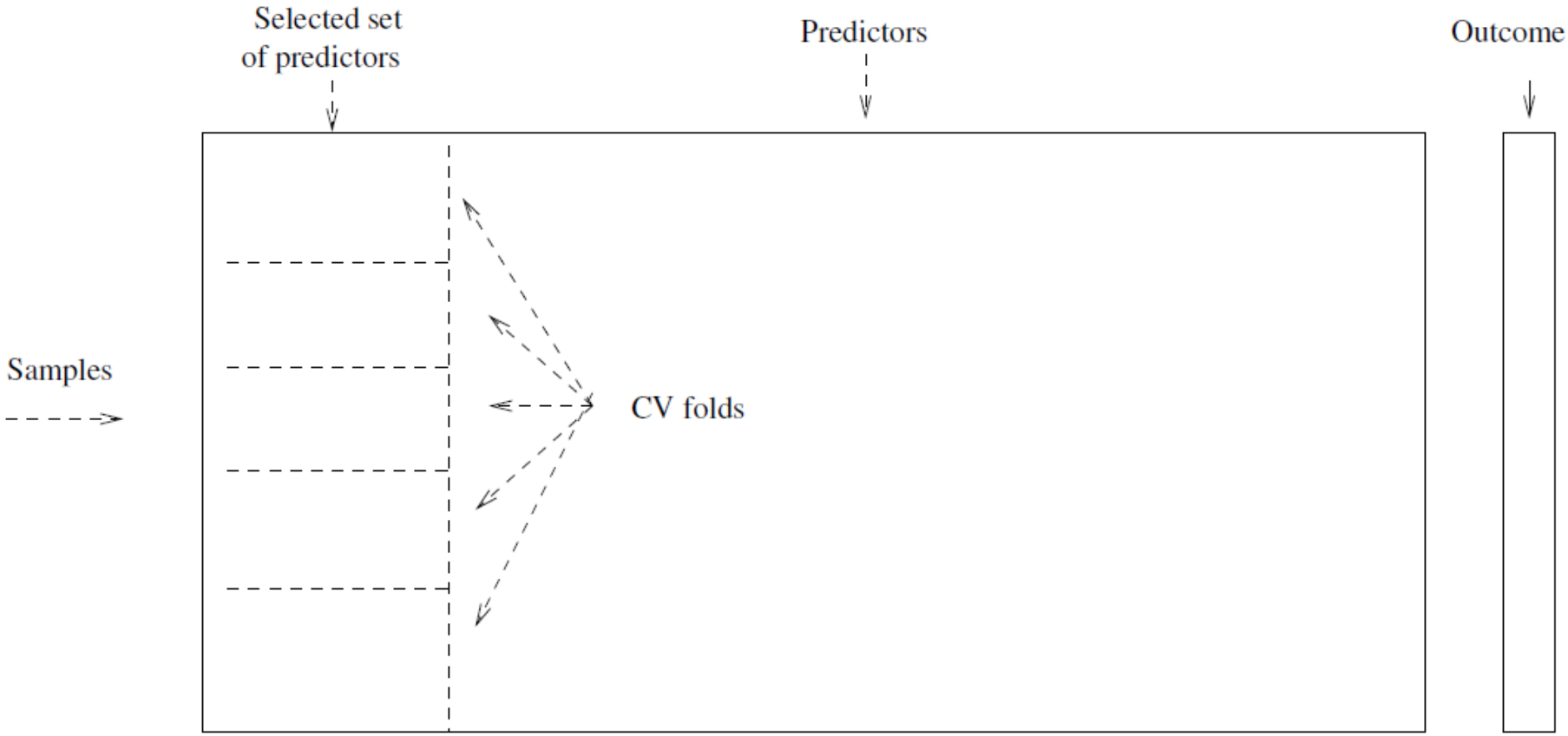
Cross-Validation on Classification Problems

- We will cover classification problems in more detail later in the course, but we briefly show how CV can be used when *Y* is qualitative (categorical) as opposed to quantitative. Here, rather than use MSE to quantify test error, we instead use the number of misclassified observations.

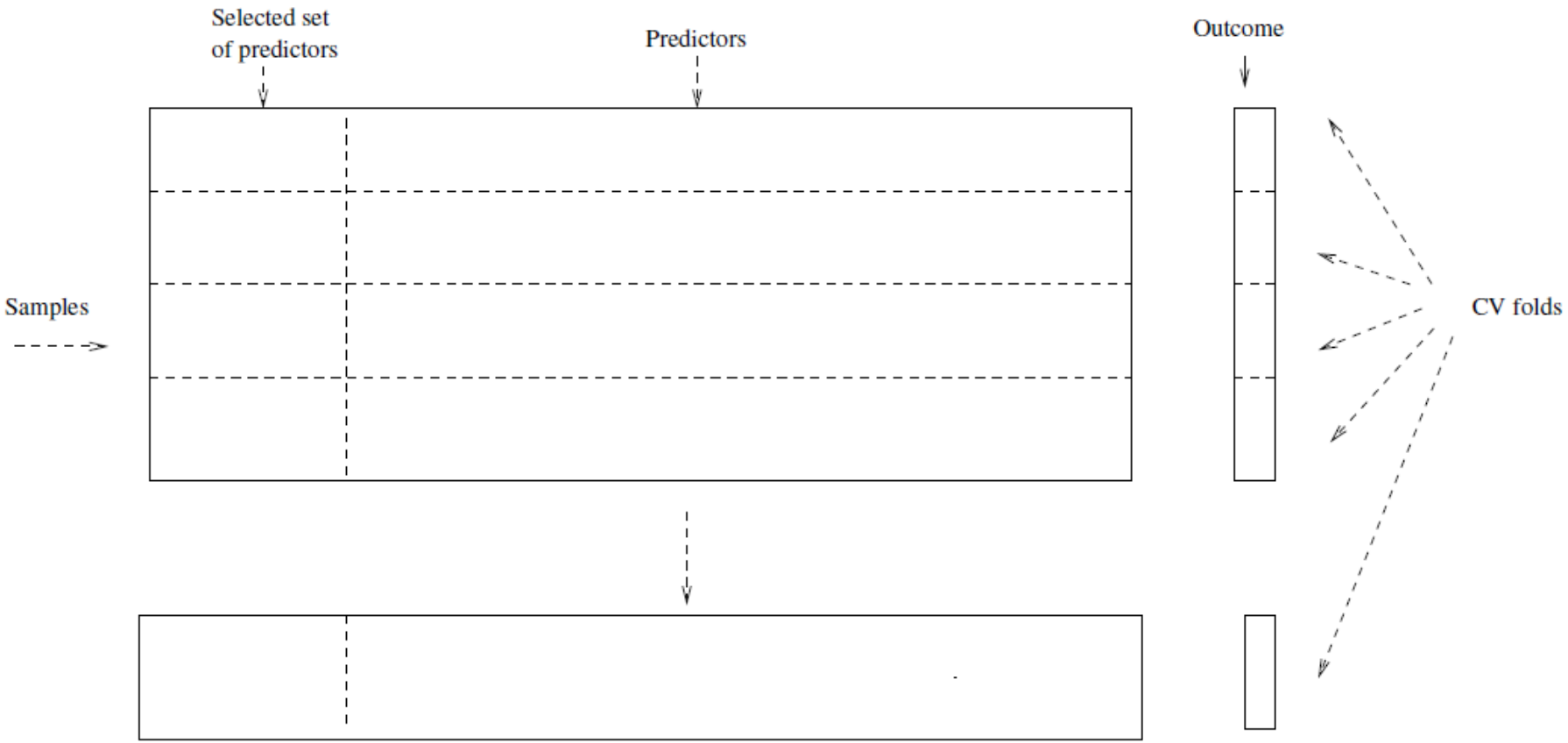
- LOOCV Error Rate: $\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i$, where $\mathrm{Err}_i = I(Y_i \neq \hat{Y}_i)$

- We use CV as follows:
  - Divide data into *K* folds; hold-out one part and fit using the remaining data (compute error rate on hold-out data); repeat *K* times.
  - CV Error Rate: average over the *K* errors we have computed

Cross-Validation: Wrong Way

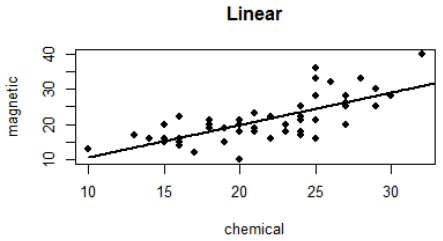


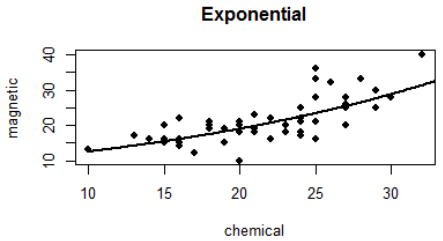
Cross-Validation: Right Way

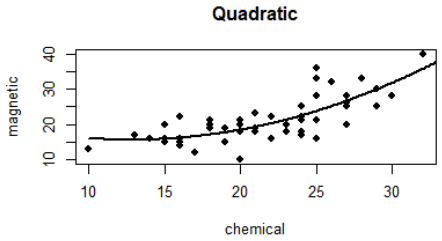


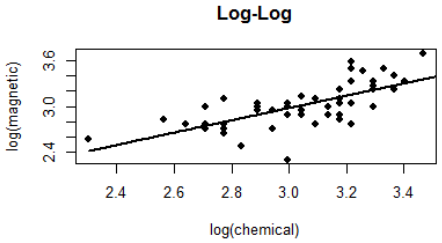
Cross-Validation Example in R

- We are looking at 53 measurements of iron content by two methods, chemical and magnetic.
- The scatterplots suggest that the variables are positively correlated, but the relation may not be linear.
- A quadratic polynomial (or possibly exponential or logarithmic model) might fit the data better than a line.
- Let's use LOOCV to estimate the prediction error, and then perform model selection.









Cross-Validation Example in R (cont.)

LOOCV Procedure:

- For $k = 1,\ldots,n$, let observation (x_k, y_k) be the test point and use the remaining obs. to fit the model.
  - Fit the model(s) using only the n-1 observations in the training set, (x_i, y_i) $i \neq k$.
  - Compute the predicted response for the test point.
  - Compute the prediction error.

- Estimate the mean of the squared prediction errors.

```
### Model selection: LOOCV
n <- length(magnetic)   #in DAAG ironslag
e1 <- e2 <- e3 <- e4 <- numeric(n)

# for n-fold cross validation
# fit models on leave-one-out samples
for (k in 1:n) {
    y <- magnetic[-k]
    x <- chemical[-k]

    J1 <- lm(y ~ x)
    yhat1 <- J1$coef[1] + J1$coef[2] * chemical[k]
    e1[k] <- magnetic[k] - yhat1

    J2 <- lm(y ~ x + I(x^2))
    yhat2 <- J2$coef[1] + J2$coef[2] * chemical[k] +
          J2$coef[3] * chemical[k]^2
    e2[k] <- magnetic[k] - yhat2

    J3 <- lm(log(y) ~ x)
    logyhat3 <- J3$coef[1] + J3$coef[2] * chemical[k]
    yhat3 <- exp(logyhat3)
    e3[k] <- magnetic[k] - yhat3

    J4 <- lm(log(y) ~ log(x))
    logyhat4 <- J4$coef[1] + J4$coef[2] * log(chemical[k])
    yhat4 <- exp(logyhat4)
    e4[k] <- magnetic[k] - yhat4
}

c(mean(e1^2), mean(e2^2), mean(e3^2), mean(e4^2))

#selected model
L2

par(mfrow = c(2, 2))     #layout for graphs
plot(L2$fit, L2$res)     #residuals vs fitted values
abline(0, 0)             #reference line
qqnorm(L2$res)           #normal probability plot
qqline(L2$res)           #reference line
par(mfrow = c(1, 1))     #restore display
```

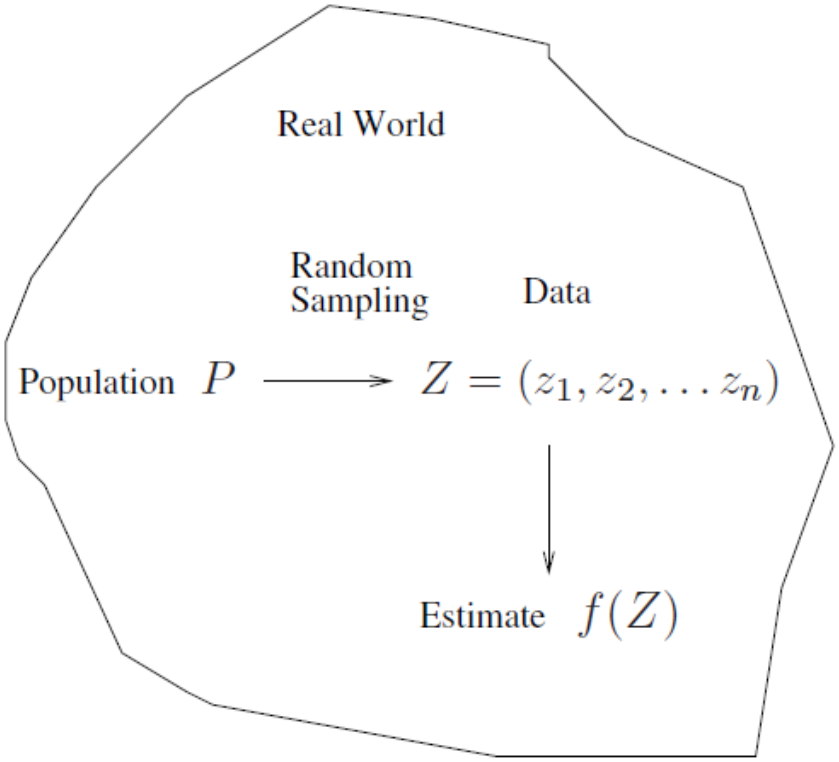
The Bootstrap

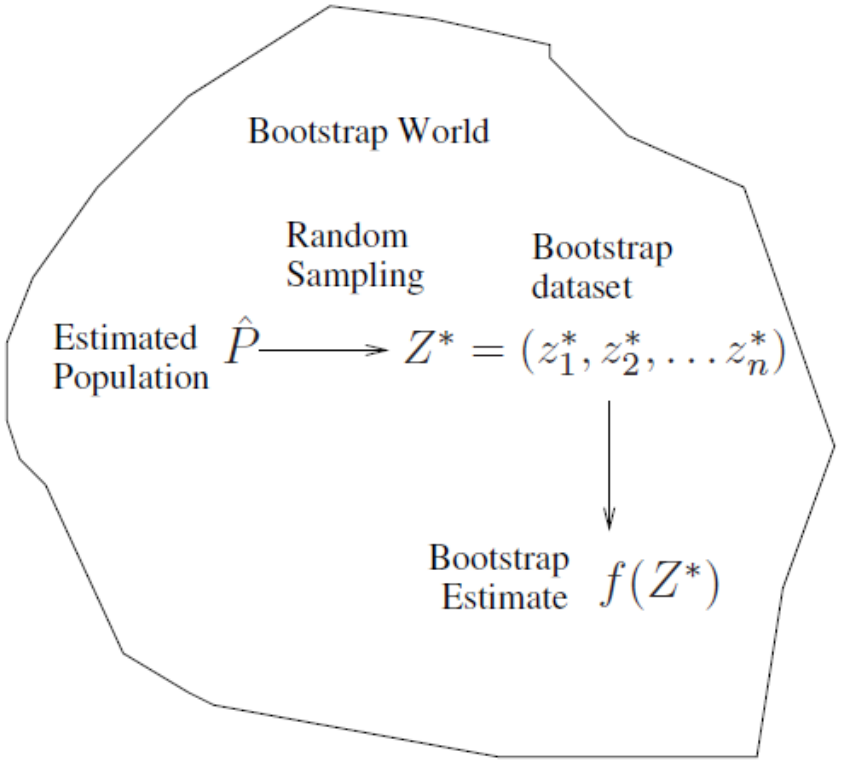
- The *bootstrap* is a flexible and powerful statistical tool that can be used to quantify uncertainty associated with a given estimator or machine learning method; it is a general tool for assessing statistical accuracy.

- As an example, the bootstrap can be used to estimate the standard errors of the coefficients from a linear regression fit, or a confidence interval for that coefficient.

- The use of the term bootstrap derives from the phrase “to pull oneself up by one’s bootstraps.”

The Bootstrap: Overview





The Bootstrap: Overview (cont.)

- Suppose we have a model fit to a set of training data. We denote the training set by $\mathbf{Z} = (z_1, z_2, \ldots, z_N)$ where $z_i = (x_i, y_i)$.

- The basic idea is to randomly draw datasets **with replacement** from the training data, each sample the same size as the original training set.

- This is done *B* times, producing *B* bootstrap datasets. Then we refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the *B* replications.

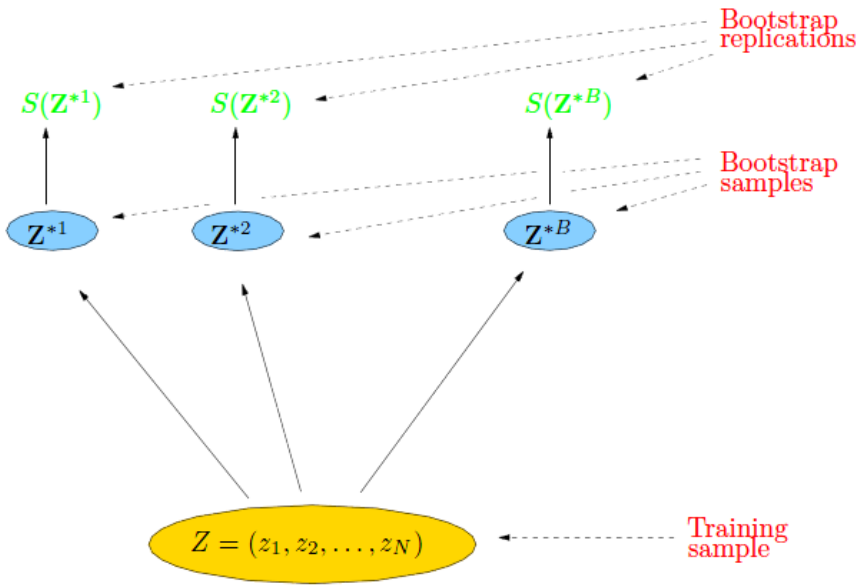
The Bootstrap: Overview (cont.)

- *S*(**Z**) is any quantity computed from the data **Z**, for example, the prediction at some input point.

- From the bootstrap sampling we can estimate any aspect of the distribution of *S*(**Z**), for example, its variance:

$$\widehat{\mathrm{Var}}[S(\mathbf{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\mathbf{Z}^{*b}) - \bar{S}^*)^2$$

$$\bar{S}^* = \sum_b S(\mathbf{Z}^{*b})/B$$



Note that this estimated variance can be thought of as a Monte-Carlo estimate of the variance of *S*(**Z**) under sampling from the empirical distribution function.

The Bootstrap: An Example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns X and Y; note that X and Y are random quantities.

- We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y.

- **Goal:** Since there is variability associated with the returns on these two assets, we wish to choose α to minimize the total risk, or variance, of our investment.

The Bootstrap: An Example (cont.)

- In other words, we want to minimize Var($\alpha X + (1 - \alpha)Y$)
- The value that minimizes the risk, in this example, is: $\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$

where $\sigma_X^2 = \mathrm{Var}(X), \sigma_Y^2 = \mathrm{Var}(Y)$, and $\sigma_{XY} = \mathrm{Cov}(X, Y)$

- In reality, the quantities σ_X^2, σ_Y^2, and σ_{XY} are unknown, so we can compute estimates for these quantities using a data set that contains past measurements for *X* and *Y*. We can then estimate the value of α that minimizes risk using:

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

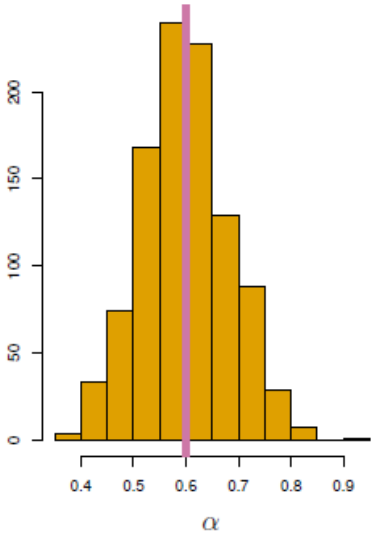
The Bootstrap: An Example (cont.)

- We can use this approach for estimating α on a simulated data set, where 100 pairs of returns for the investments *X* and *Y* are simulated.
- In order to quantify the accuracy of our estimate of α, we also estimate the standard deviation of $\hat{\alpha}$.
- To do so, the process of simulating 100 paired observations of *X* and *Y* is repeated 1,000 times; now we have 1,000 estimates for α.
- The mean over all 1,000 estimates for α →

$$\bar{\alpha} = \frac{1}{1000}\sum_{r=1}^{1000}\hat{\alpha}_r = 0.5996$$

- The standard deviation of the estimates →

$$\sqrt{\frac{1}{1000-1}\sum_{r=1}^{1000}(\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$



The Bootstrap: An Example (cont.)

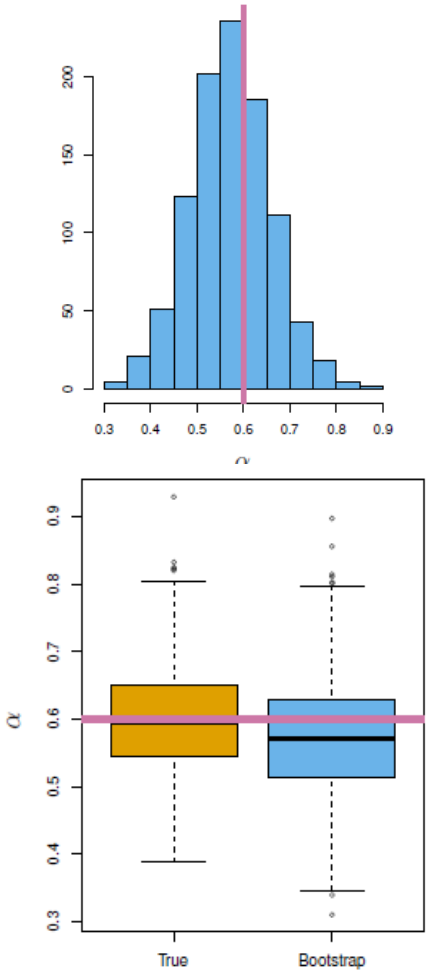
- In rough terms, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

- In the real world, this procedure **cannot be applied** because for real data we cannot generate new samples from the original population.

- The **bootstrap** approach, however, allows us to use a computer to mimic the process of obtaining new data sets; this enables us to estimate the variability of our estimate without generating additional samples.

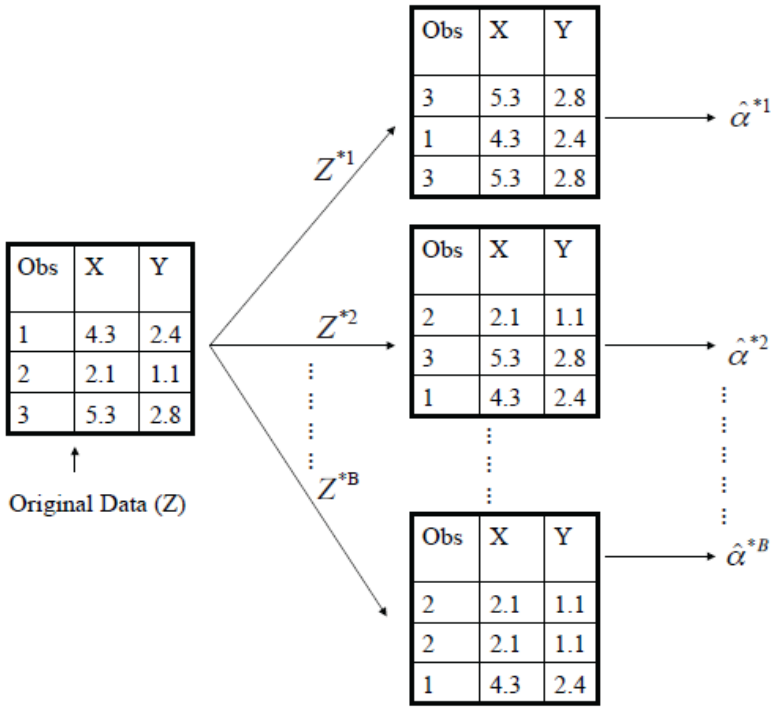
The Bootstrap: An Example (cont.)

- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set **with replacement**.
- Each of these *bootstrap data sets* is created by sampling *with replacement*, and is the *same size* as the original data set.
- As a result, some observations may appear more than once in a given bootstrap data set and some not at all.



The Bootstrap: An Example (cont.)

- 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.



The Bootstrap: An Example (cont.)

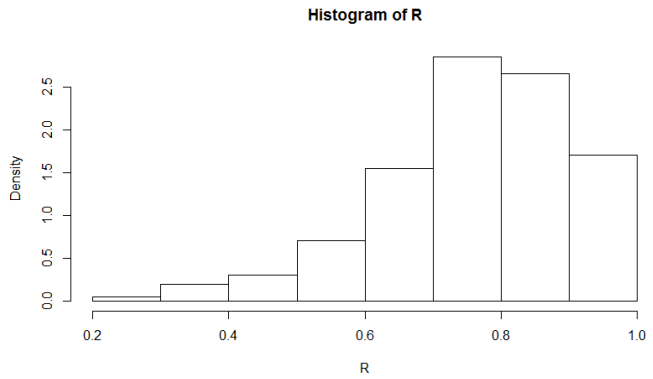
- Denoting the first bootstrap data set by Z^{*1}, we can use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{*1}$.
- This procedure is repeated *B* times for some large value of *B* (say 100 or 1000), in order to produce *B* different bootstrap data sets, Z^{*1}, Z^{*2},…, Z^{*B}, and *B* corresponding α estimates, $\hat{\alpha}^{*1}$, $\hat{\alpha}^{*2}$,…, $\hat{\alpha}^{*B}$.
- We compute the standard error of these bootstrap estimates using:

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^{*}\right)^2}$$

- This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set.

Bootstrap Example in R

```r
### Example: Bootstrap estimate of standard error

library(bootstrap)    # for the law data
law
law82
print(cor(law$LSAT, law$GPA))
print(cor(law82$LSAT, law82$GPA))

#set up the bootstrap
B <- 200            #number of replicates
n <- nrow(law)      #sample size
R <- numeric(B)     #storage for replicates

#bootstrap estimate of standard error of R
for (b in 1:B) {
    #randomly select the indices
    i <- sample(1:n, size = n, replace = TRUE)
    LSAT <- law$LSAT[i]       #i is a vector of indices
    GPA <- law$GPA[i]
    R[b] <- cor(LSAT, GPA)
}
#output
print(se.R <- sd(R))
hist(R, prob = TRUE)


### Example: Bootstrap estimate of standard error: boot function

r <- function(x, i) {
    #want correlation of columns 1 and 2
    cor(x[i,1], x[i,2])
}

library(boot)       #for boot function
obj <- boot(data = law, statistic = r, R = 2000)
obj
y <- obj$t
sd(y)
```



- The law school data set in the bootstrap() package contains LSAT (average score on law school admission test score) and GPA (average undergraduate GPA) for 15 law schools.
- The data set is a random sample from the universe of 82 law schools.
- **Goal:** estimate the correlation between LSAT and GPA scores, and compute the bootstrap estimate of the standard error of the sample correlation.

Bootstrap Example in R (cont.)

- The **bias** of an estimator $\hat{\theta}$ for θ is: $bias(\hat{\theta}) = E[\hat{\theta} - \theta] = E[\hat{\theta}] - \theta$.

- The bootstrap estimation of bias uses the bootstrap replicates of $\hat{\theta}$ to estimate the sampling distribution of $\hat{\theta}$.

- The sample mean of the replicates is unbiased for its expected value, so the bootstrap estimate of bias is: $\widehat{bias}(\hat{\theta}) = \overline{\hat{\theta}^*} - \hat{\theta}$

Mean of the distribution of the bootstrap estimates

Estimate computed from the observed sample

```
### Bootstrap estimate of bias

#sample estimate for n=15
theta.hat <- cor(law$LSAT, law$GPA)

#bootstrap estimate of bias
B <- 2000   #larger for estimating bias
n <- nrow(law)
theta.b <- numeric(B)

for (b in 1:B) {
    i <- sample(1:n, size = n, replace = TRUE)
    LSAT <- law$LSAT[i]
    GPA <- law$GPA[i]
    theta.b[b] <- cor(LSAT, GPA)
}
bias <- mean(theta.b - theta.hat)
bias
```

Positive bias indicates that $\hat{\theta}$ on average tends to overestimate θ.

The Bootstrap: More Details

- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thoughts.

- For example, if the data is a time series, we cannot simply sample the observations with replacement.

- However, we can instead create blocks of consecutive observations, and sample those with replacement. Then, we paste together sampled blocks to obtain a bootstrap dataset.

The Bootstrap: More Details (cont.)

- Although the bootstrap is used primarily to obtain standard errors of an estimate, it can also provide approximate confidence intervals for a population parameters.
  - One approach is known as *Bootstrap Percentile* confidence interval.

- In cross-validation, each of the *K* validation folds is distinct from the other *K* – 1 folds used for training (i.e. there is no overlap).

- To estimate the prediction error using the bootstrap, one approach would be to fit the model in question on a set of bootstrap samples, and then keep track of how well it predicts the original training set.

The Bootstrap: More Details (cont.)

- If $\hat{f}^{*b}(x_i)$ is the predicted value at x_i, from the model fitted to the bth bootstrap dataset, our estimate is:

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i))$$

- This estimate does not provide a good estimate in general because the bootstrap datasets are acting as the training samples, while the original training set is acting as the test sample, and these two samples have observations in common.
  - Each bootstrap sample has significant overlap with the original data.

The Bootstrap: More Details (cont.)

- Note that about 2/3 of the original *N* data points appear in each bootstrap sample:
  - $\Pr\{\text{observation } i \in \text{bootstrap sample } b\} = 1 - (1 - \frac{1}{N})^N \approx 1 - e^{-1} = 0.632$

- This overlap can make overfit predictions like unrealistically good, and is the reason that cross-validation explicitly uses non-overlapping data for the training and test samples.

- In other words, this will cause the bootstrap to seriously *underestimate* the true prediction error.

- By mimicking cross-validation, a better bootstrap estimate can be obtained.

The Bootstrap: More Details (cont.)

- For each observation, we only keep track of prediction from bootstrap samples not containing that observation.
- The leave-one-out bootstrap estimate of prediction error is defined by:

$$\widehat{\mathrm{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i))$$

- Here C^{-i} is the set of indices of the bootstrap samples b that do *not* contain observation I, and $|C^{-i}|$ is the number of such samples.

- Note that the leave-one-out bootstrap solves the problem of overfitting, but has a training-set-size bias.
  - The ".632 estimator" is designed to alleviate this bias.

Summary

- Resampling methods for machine learning model selection and assessment.
- The validation set approach, leave-one-out cross-validation, and *K*-fold cross-validation.
- The bootstrap method for assessing statistical accuracy.