Freie Universität Berlin

# Data Science
# Lecture 06

30.05.2014

Dr. Kashif Rasul
kashif @krasul #167

Shoaib Burq
sabman @sabman #167

Welcome to Data Science Lecture 6.

# Last Time

- Bayes' Rule:
  Posterior odds = Prior odds * Likelihood ratio
- LDA: normal and common pop. variance
- QDA: normal and class variance

# Today

- Resampling
- Cross-Validation
- Bootstrap



We've learned about methods for regression and for classification involving predictors and for making predictions from our data. But how do we test these out?

Ideally, we'd like to get a new sample from the population to see how well our prediction did. But we don’t have new data and we can’t use our training data because it will be a little bit optimistic. So today we will talk about Resampling methods and look at Cross-validation and Bootstrap in details.

# Resampling

- Methods involve repeatedly drawing samples from a training set and refitting a model of interest on each sample
- Common methods: cross-validation and bootstrap
- Can be used to evaluate a model's performance: model assessment
- Can be used to select the proper flexibility: model selection



Resampling methods are an indispensable tool in modern statistics.

E.g. in order to estimate the variability of a linear regression fit, we can repeatedly draw different samples from the training data, fit a linear regression to each new sample, and then examine the extent to which the resulting fits differ.

Cross-validation can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance (model assessment), or to select the appropriate level of flexibility (model selection).

The bootstrap is used in several contexts, most commonly to provide a measure of accuracy (standard deviation) of a parameter estimate or of a given statistical learning method.

# Cross-Validation

- Estimate the quantity of the test error by using the training data
- To this by holding out a subset of the training observation from the fitting process
- Validation Set Approach
- LOOCV
- k-Fold CV



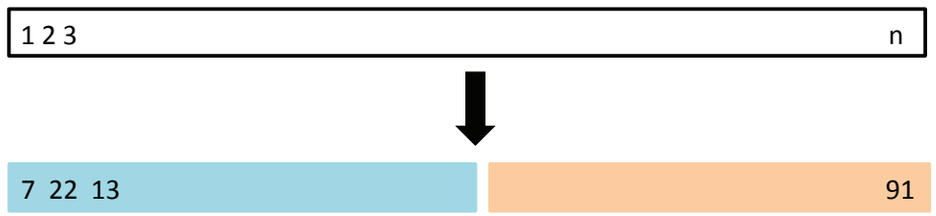
Recall test-error is the average error from using a model to predict the response on a new observation. It is data that was not used to train the model. Training error can be easily calculated by applying the model to the observations used in training. Often these two are quite different. And due to the Bias-Variance tradeoff as we increase the complexity of the model, the test error will start to increase after some point. If we can estimate the test-error we can find this this point.

So in the absence of a very large designated test set that can be used to directly estimate the test error rate, a number of techniques can be used to estimate this quantity directly using the training data.

Cross-Validation is a class of methods that estimate the test error rate by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

We will look at these 3 methods.

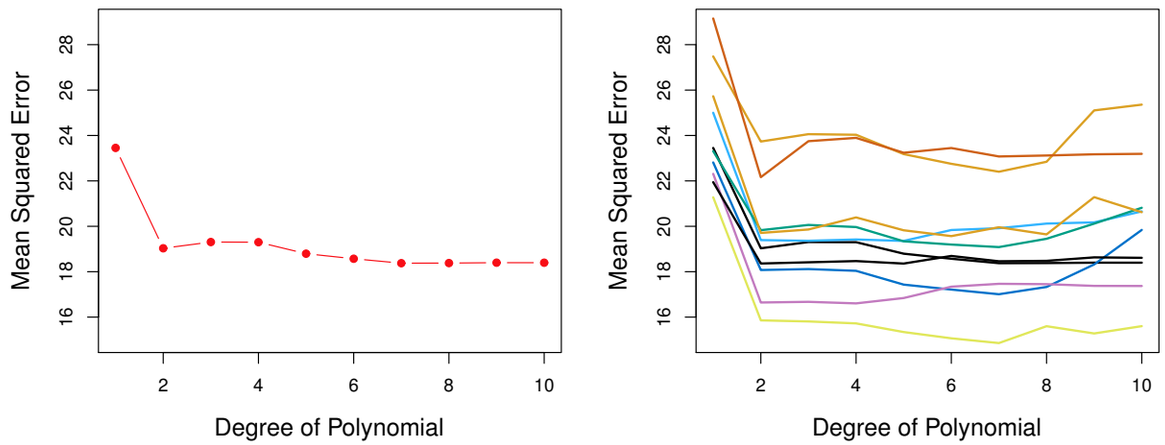
# Validation Set






This is the simplest strategy: It involves randomly dividing the available set of observations into two parts, a training set and a validation set or hold-out set.

The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The resulting validation set error rate is measured via the MSE and provides an estimate of the test error rate.






Recall how for the Auto dataset we used the p-value associated with a cubic term and higher-order polynomial terms in a linear regression to answer the question of weather a higher order polynomial fit might provide better results. Well we can do the same via Validation Sets.

We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations. The validation set error rates that result from fitting various regression models on the training sample and evaluating their performance on the validation sample, using MSE as a measure of validation set error, are shown in the left-hand. We see the quadratic model provides the lowest MSE, with the cubic being a bit higher.

If we repeat the process of randomly splitting the sample set into two parts, we will get a somewhat different estimate for the test MSE. This is shown in the right panel.

All ten curves indicate that the model with a quadratic term has a dramatically smaller validation set MSE than the model with only a linear term. Furthermore, all ten curves indicate that there is not much benefit in including cubic or higher-order polynomial terms in the model. But it is worth noting that each of the ten curves results in a different test MSE estimate for each of the ten regression models considered.

# Drawbacks

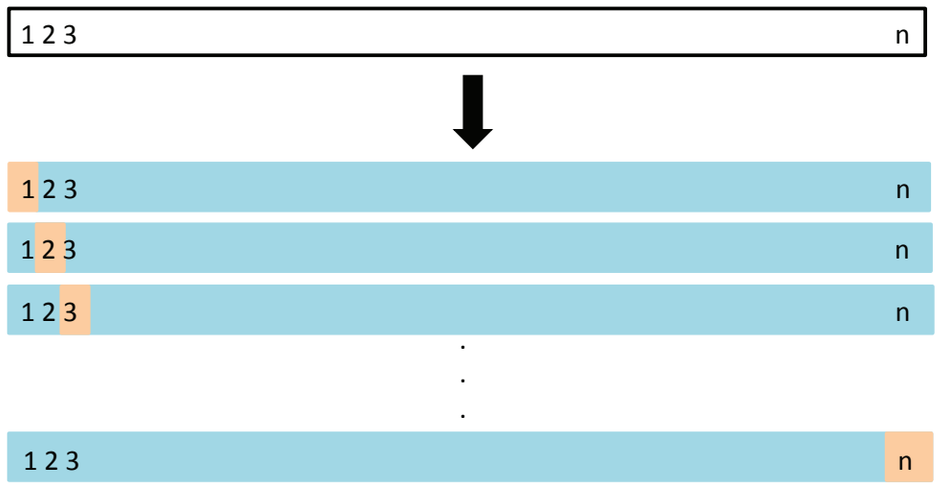
- The validation estimate of the test error rate can be highly variable: splitting into 2 parts
- Only a subset of the observations are used to fit the model: error rate may tend to be over estimated



The validation set approach is conceptually simple and is easy to implement. But it has two potential drawbacks. Firstly, validation estimate of the test error rate can be highly variable. Secondly, since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set. This is because the more data one has the lower the error and here we are throwing half of the data away.

So we need to address these problems via the methods of cross-validation.

# LOOCV






Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach. Here a single observation is used for the validation set and the model is fit on the n-1 observations and a prediction is made e.g. $MSE_1 = (y_1 - \hat{y}_1)^2$. We repeat this for the next observation and end up with n $MSE_i$. The LOOCV estimate for the test error MSE is the average of these n test error estimates and is denoted by $CV_{(n)}$.

This has far less bias since we are fitting with n-1 observations, compared to n/2. So this approach tends not to overestimate the test error rate as much as the validation set approach.

Secondly, performing this multiple times will yield the same result, since there is no randomness in the training/validation set split.

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

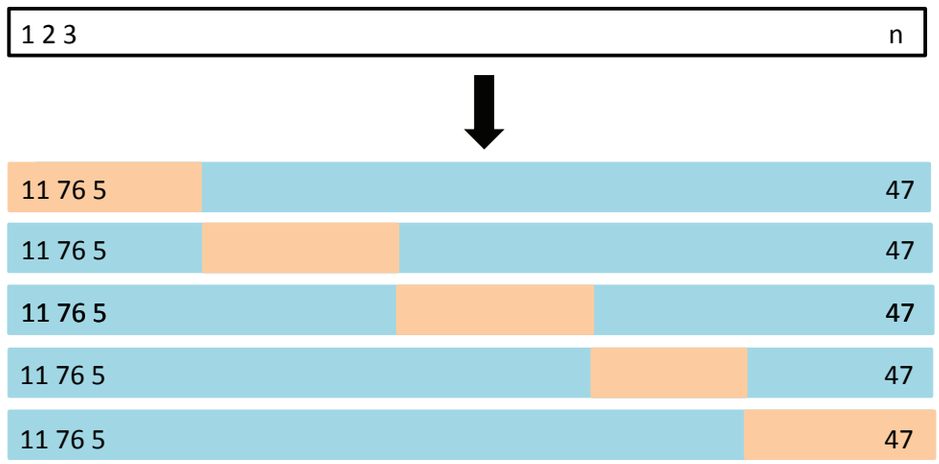
$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{j=1}^{n}(x_j - \bar{x})^2}$$



LOOCV has the potential to be expensive to implement, since the model has to be fit n times. This can be very time consuming if n is large, and if each individual model is slow to fit. With least squares linear or polynomial regression, an amazing shortcut makes the cost of LOOCV the same as that of a single model fit! The following formula holds, where $h_i$ is the leverage. The leverage lies between 1/n and 1 and reflects the the amount that an observation influences its own fit.

LOOCV is a very general method, and can be used with any kind of predictive modelling. For example we could use it with logistic regression or linear discriminant analysis, or any of the methods discussed later. This formula does not hold in general and in that case we have to re-fit the model n times.

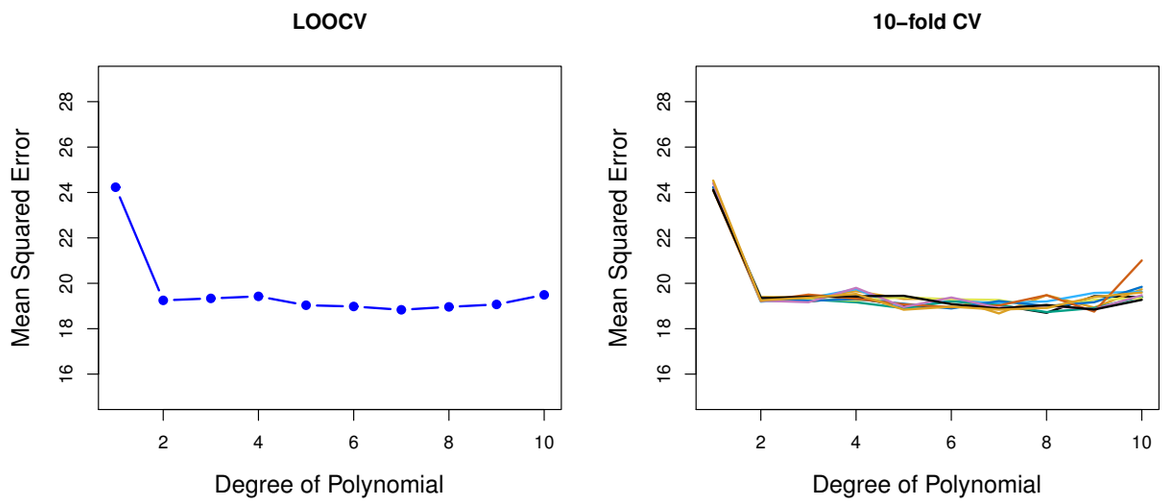
# k-fold CV






This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k − 1 folds. The mean squared error, $MSE_1$, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error and the k-fold CV is the mean of these values.

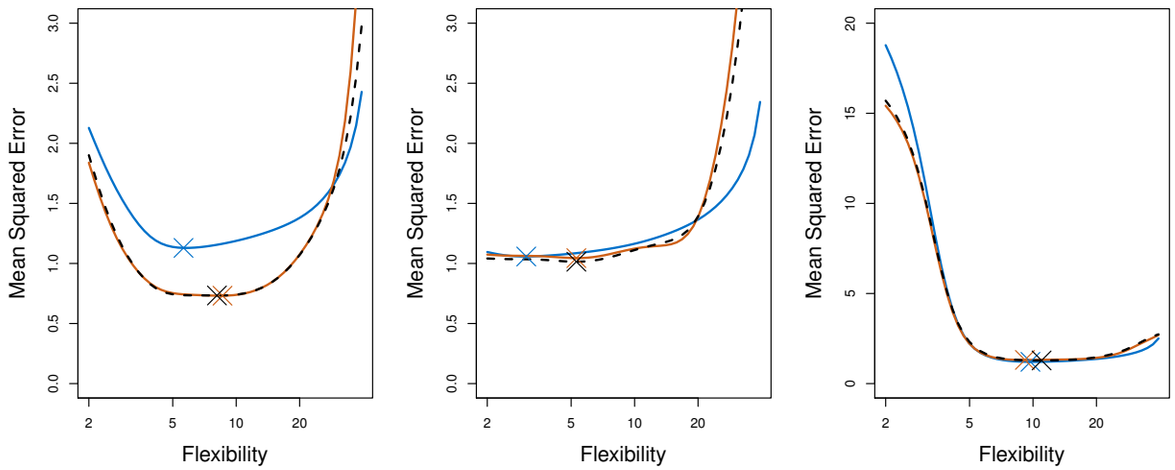
LOOCV is a special case of k-fold CV and the obvious advantage here is computation. E.g. 10-fold CV involves fitting the model 10 times which might be more feasible.






On the right we do a 10-fold CV on the Auto data done nine times. As we can see from the figure, there is some variability in the CV estimates as a result of the variability in how the observations are divided into ten folds. But this variability is typically much lower than the variability in the test error estimates that results from the validation set approach.

When we examine real data, we do not know the true test MSE, and so it is difficult to determine the accuracy of the cross-validation estimate.






However, if we examine simulated data, then we can compute the true test MSE, and can thereby evaluate the accuracy of our cross-validation results. Here we we plot the cross-validation estimates and true test error rates that result from applying smoothing splines to the simulated data sets.

The true test MSE is displayed in blue, the black dashed and orange lines are the LOOCV and 10-fold CV estimates. In all three plots, the two cross-validation estimates are very similar.

In the centre panel the two sets are similar at lower flexibility, while the CV overestimates the test set MSE for higher degree of flexibility. In the left panel, the CV curve have the correct general shape, but underestimate the true test MSE.

So an actual estimate of the test MSE might be of interest when we need to know how well a given statistical learning method can be expected to perform on independent data. At other times we are interested in the location of the minimum point of the test MSE curve for model selection. And despite the fact that the CV curves underestimate the true MSE error, it does come close to finding the correct level of flexibility.

# Bias-Variance for CV

- LOOCV is n-fold CV: low bias & high variance
- Validation set: 2-fold CV: high bias & low variance
- k=10 or 5-fold: sweet spot for bias-variance trade-off



LOOCV will give approximately unbiased estimates of the test error, since each training set contains n − 1 observations, which is almost as many as the number of observations in the full data set. Validation set only contains half the data so that will have a high bias. And performing k-fold CV for, say, k = 5 or k = 10 will lead to an intermediate level of bias, since each training set contains (k − 1)n/k observations, which are fewer than for LOOCV but a lot more than Validation set.

LOOCV has higher variance, since when we do this we are effectively averaging the outputs of n fitted models, each of which is trained on an almost identical set of data. So these outputs are highly positively correlated with each other.

On the other hand k-fold CV with k < n, we are averaging the outputs of k fitted models that are somewhat less correlated with each other, since the overlap between the training sets in each model is smaller.

Since the means of many highly correlated quantities has higher variance we have that LOOCV has high variance and validation set low variance.

Typically k=5 or k=10 hit the sweet spot for CV.

# CV on Classification

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



In the classification setting, CV works just as described earlier, except that instead of the MSE we use the Err which is the number of misclassified observations and $n_i$ is the number of observations in the ith fold. If $n_i = n/k$ then we get the 1/k. So one can fit various logistic regression models to the data and compute the k-fold cross-validation.

By doing say a 10-fold CV for real data where the Bayes decision boundary is un-known and the the test error rates are unknown, we still get the characteristic U-shaped curve of the estimated CV error rate which will lie typically below the actual test error and above the training error. And we can use this to select the optimal flexibility of the logistic model.

We can also compute the approximate standard deviation of the $CV_{(k)}$ and thus can even plot error bands on our curve which is useful sometimes too.

# CV done wrong ⚠️

1. Start with 5,000 predictors and 50 samples: find the 100 predictors with largest correlation to labels

2. Apply a classifier e.g. logistic regression to the data with these 100 predictors

3. Apply CV in step 2?



Due to it's popularity and importance CV is used a lot but there is a scenario where CV can be very wrong. Consider the case (which is not that un-common these days e.g. genomic data) for a binary classifier where we do 1 and 2.

How do we estimate the test set performance of this classifier? Can we apply CV in step 2? Turns out the answer is NO! Many published studies make this mistake so it is worth pointing out this error.

Why is this the case? Well by doing step 1 we have already seen all the training data. And so we cannot start CV at step 2 and ignore the fact that we have seen all the data in step 1. You can try it yourself by generating data with the test error of say 50%. The you find the top predictors with the largest correlation and by applying CV to this you end up with a test error of 0! CV is saying your classifier is perfect where in fact it is the same as flipping a coin. Think about why this happens: well suppose instead of 5000 predictors we we had 5 million predictors and from that we find the best 100. Well we will find some really good predictors and they are going to look very good to CV. So we are fooling CV by cherry-picking a good set of predictors and going over the whole data.

# CV done right 😸

- Apply CV to both steps:
  - Define the folds say 5 or 10 and remove one fold
  - Now we do step 1 on the remaining folds
  - Then step 2



The key point being, though, that we form the folds before we filter or fit to the data.

So that we're applying cross-validation to the entire process, not just the second step. So that's the right way to do it. So in each of the 4/5ths folds, we might screen off a different set of predictors each time.

# 🥾 Bootstrap

- Flexible and power technique used to quantify uncertainty associated with a given model
- Can give estimate of the standard errors of the coefficients from a linear regression fit
- Easily applied to wide range of learning methods



Now lets talk about a related idea called Bootstrap for predicting uncertainty in estimates. And it is particularly good for getting standard errors of an estimate, and getting confidence limits.

In linear regression this might not be particularly useful, since scikit-learn already does that, but it it can be easily applied to a wide range of statistical learning methods, including some for which a measure of variability is otherwise difficult to obtain and is not automatically output by statistical software.

The name comes from the idea of pulling yourself up by your bootstraps, which is from a fable by Rudolph Erich Raspe: “The Surprising Adventures of Baron Munchausen” where the Munchausen got out of the lake by pulling himself by his bootstraps. The idea is similar here: we will use the data itself to get more information about our estimator.

# Toy example

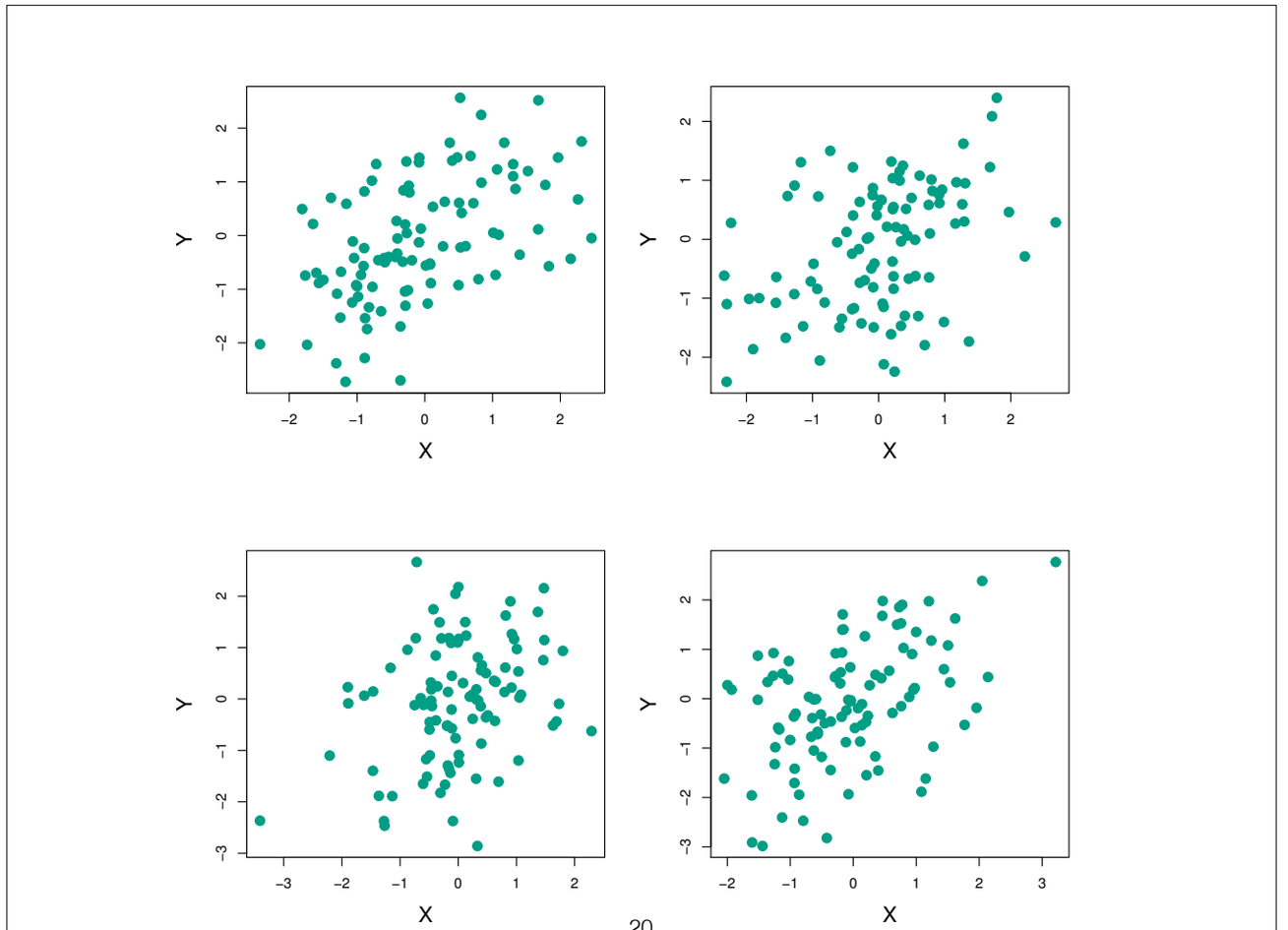
- X and Y two random financial assets
- Invest a fraction α into X and 1-α into Y
- Choose α to minimise total risk i.e. Var(αX+(1-α)Y)

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



So to start suppose we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities. We will invest a fraction $\alpha$ of our money in X, and will invest the remaining $1 - \alpha$ in Y.

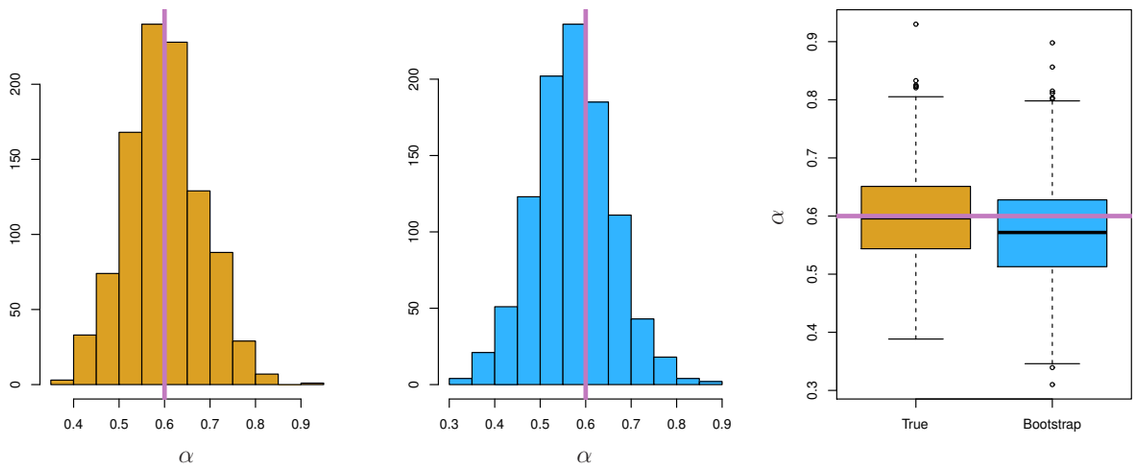
One can show that the minimum $\alpha$ is given by the above formula where $\sigma^2_X$ = Var(X), $\sigma^2_Y$ = Var(Y) and $\sigma_{XY}$=Cov(X,Y). In reality these quantities are unknown.






But we can estimate them using a data set that contains say part measurements for X and Y. We show this approach by this diagram where in each panel we simulated 100 pair of returns for the investments X and Y. We use this to estimate $\hat{\alpha}$. The values here range from 0.532 to 0.657.

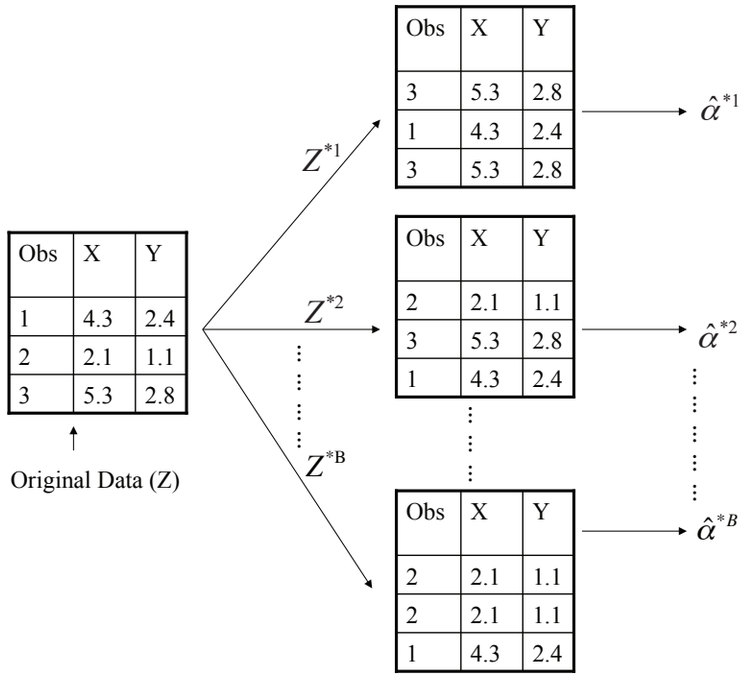
It is natural to wish to quantify the accuracy of our estimate of $\alpha$. To estimate the standard deviation of $\hat{\alpha}$ we repeat this process of simulating 100 paired observations X and Y and estimating $\alpha$ say a 1,000 times. We then obtain 1,000 estimates for $\alpha$ which we call $\hat{\alpha}_1$ , $\hat{\alpha}_2$ ,…,$\hat{\alpha}_{1,000}$ . Then we can use this to estimate the mean and standard deviation of $\alpha$.






Here we see the true distribution of the resulting estimate in orange (we know this since we generated this test data) and in blue we see the histogram of the α obtained from 1,000 bootstrap samples. The right panel shows the box plot and the pink line is the true value of α. And we see that the mean of the simulated α is very close to the real.

So by using a computer simulation we can emulate the process of obtaining new sample sets so that we can estimate the variability of $\hat{\alpha}$ without generating additional samples. 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.




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



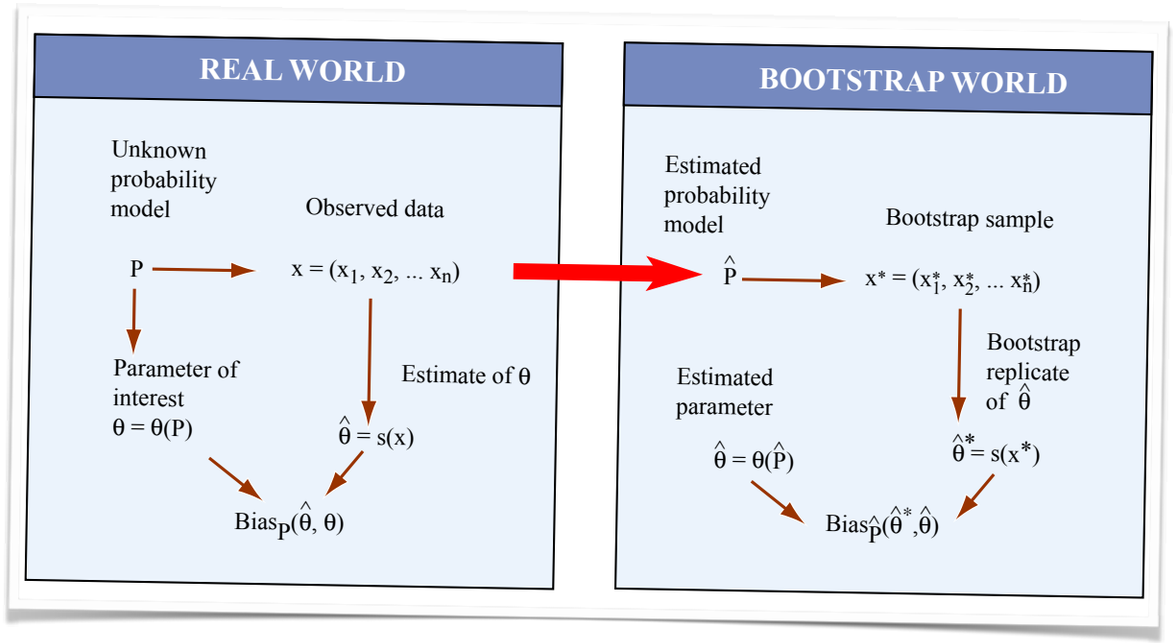
So to summarise, the bootstrap approach is illustrated above on a simple data set, which we call Z. We randomly select n observations from the data set in order to produce a bootstrap data set $Z^{*1}$.

The sampling is performed with replacement, which means that the same observation can occur more than once in the bootstrap data set. In this example, $Z^{*1}$ contains the third observation twice, the first observation once, and no instances of the second observation.

Note that if an observation is contained in $Z^{*1}$, then both its X and Y values are included.

We can use $Z^{*1}$ to produce a new bootstrap estimate for $\alpha$, which we call $\alpha^{*1}$. We repeat this for some large value B to produce B bootstrap estimates for $\alpha$. We can then compute the standard error of these bootstrap estimates via the formula above.

The great thing about the Bootstrap is that it can be applied in almost any situation. No complicated mathematical calculations are required and it gives us the accuracy of any statistic of interest.






Here is another diagram which is useful. In the real world from a population we get data which we can observe. It is like our training data. And from that we derive a
statistic, an estimate like the α^. Now if we had access to the population we could get more data, but in practice that is not possible. So a way to think of the bootstrap is to say let's replace the population by P^ which in the bootstrap case is the the training sample itself. It is a population that puts prob. 1/n on each training point.

So it says our guess for the population is a uniform prob. that it can come from the first observed data. P^ is called the empirical distribution function and we draw from this empirical distribution, i.e. we sample with replacement. And we add a superscript * to denote this.

And from this we derive estimates by repeating this process say hundreds of times. And we can then think about how to carry this out for more general situations.

# Bootstrap in general

- Need to think about how to apply bootstrap sampling to complex data situations
- E.g. for time series data we cannot sample the observations with replacement
- Setup the data: which parts are independent?



We cannot sample with replacement here since the data are not independent. We expect say the stock price for a given day to be correlated with the stock price of the day before. In fact we hope it is correlated! So thats a problem for bootstrap.

So when setting up bootstrap we must figure out which parts of the data are independent. So for example with time series we can block bootstrap: divide the data into blocks and between blocks one assumes things are independent. So we sample with replacement from all the blocks and paste them together into a new time series.

The point is we have to sample things that are uncorrelated or arrange to find parts of the data that are uncorrelated. And we keep the blocks intact and sample them as units.

And the main use of bootstrap is to get the SE and confidence intervals for the parameters $\alpha$ in our example. And we can interpret it to say if we were to repeat the experiment from the population many times it would be the case that the confidence interval we get will contain the true $\alpha$ 90% of the time. And this way of making confidence intervals is called Bootstrap Percentile Interval which is an entire field of statistics all to itself!

# Lab 4

- Look at how to do CV
- Repeat with Auto dataset



Try Lab 4 with Auto data set yourself.

http://bit.ly/data-science-6



Thank you that's it for this lecture. Your feedback is greatly appreciated as it will help us to improve and drive the course in the direction you would like it to go.