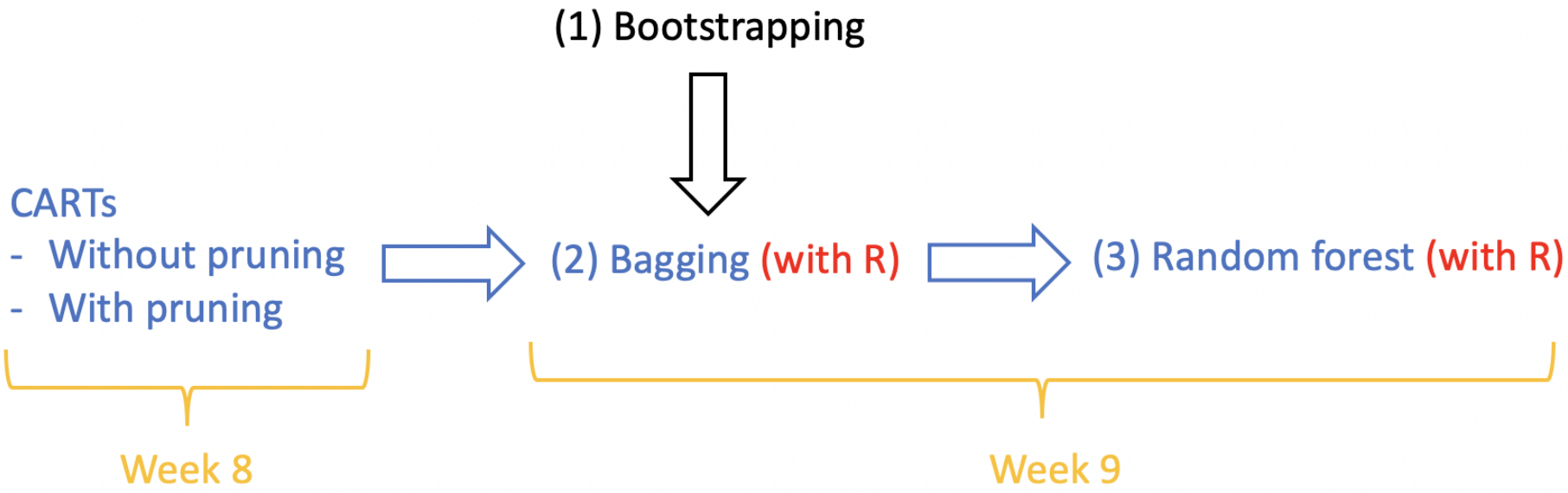
40.016: The Analytics Edge

Week 9 Lecture 1

Bagging and Random Forests

Term 5, 2022

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN


(1) Bootstrapping
CARTs
- Without pruning
- With pruning
(2) Bagging (with R)
(3) Random forest (with R)
Week 8
Week 9

Outline

Outline

Bootstrapping

- The Bootstrap is a resampling method used to quantify the uncertainty associated with a statistical learning method (or a given estimator)
  - Computer technology ("booting")
  - Statistics learning: obtaining "summary statistics" without the aid of additional data
- (The other resampling method introduced in this course is cross-validation)
- The term originates from the expression "to pull oneself up by one's bootstraps"

Example

- We want to invest a given amount 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 α of our money in X, and will invest the remaining $1 - \alpha$ in Y
- Since there is variability associated with the returns on these two assets, we wish to choose the value of α that minimizes the total risk, measured with the variance of our investment, $\mathrm{Var}(\alpha X + (1 - \alpha)Y)$

Example (cont'd)

- The value of α that minimizes the risk is

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

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

- In reality, the values of σ_X^2, σ_Y^2, and σ_{XY} are unknown
- We can calculate estimates ($\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$) using a dataset with observations of X and Y
- And then estimate $\hat{\alpha}$

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

Example (cont'd)

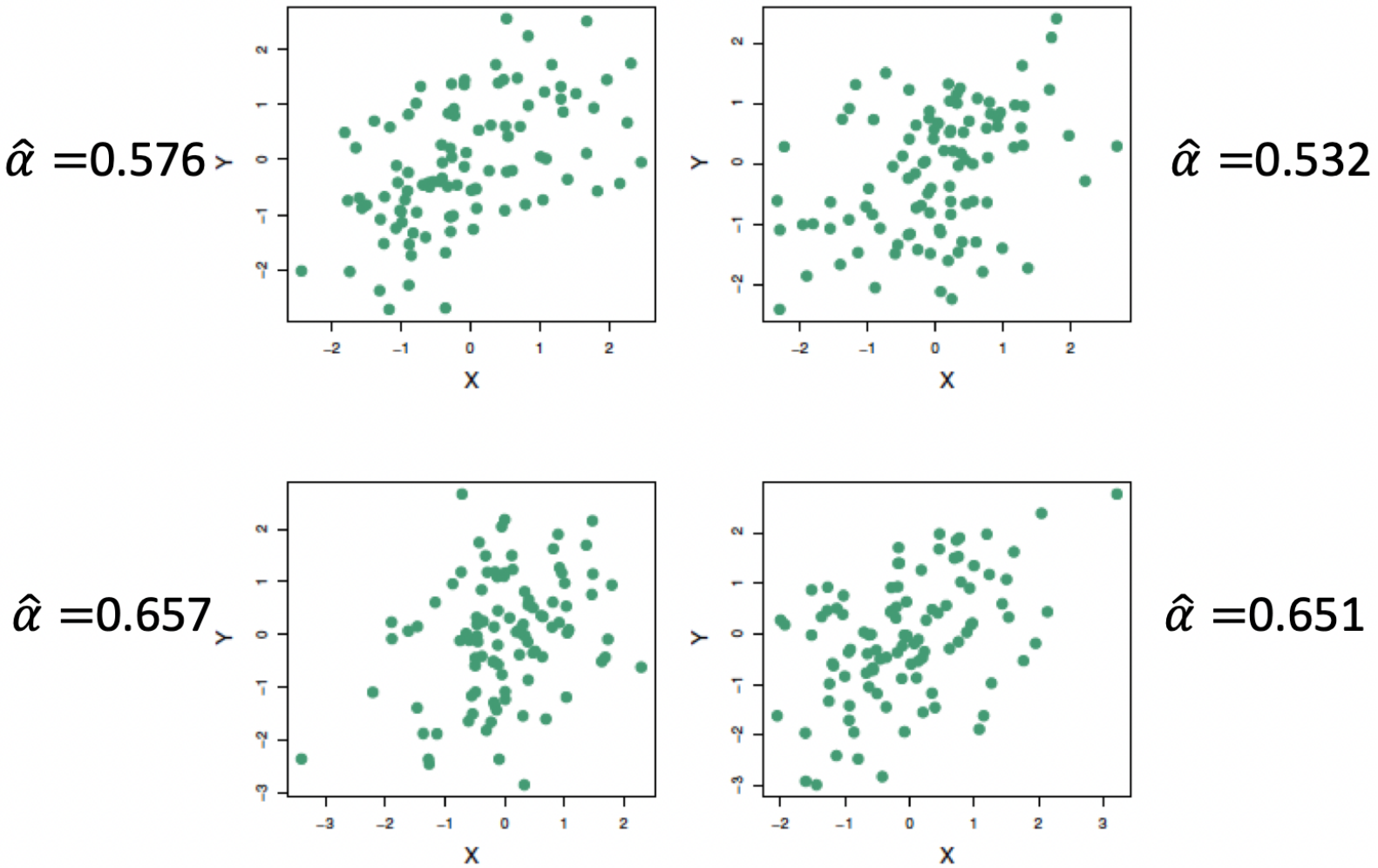


Figure: Each panel displays 100 simulated returns for investments X and Y. (Source: James et al., 2014).

Example (cont'd)

- It is natural to wish to quantify the accuracy of our estimate of α.
- To estimate the standard deviation of $\hat{\alpha}$, we repeated the process 1000 times.
- We thereby obtained 1000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_2, ..., \hat{\alpha}_{1000}$.

Example (cont'd)

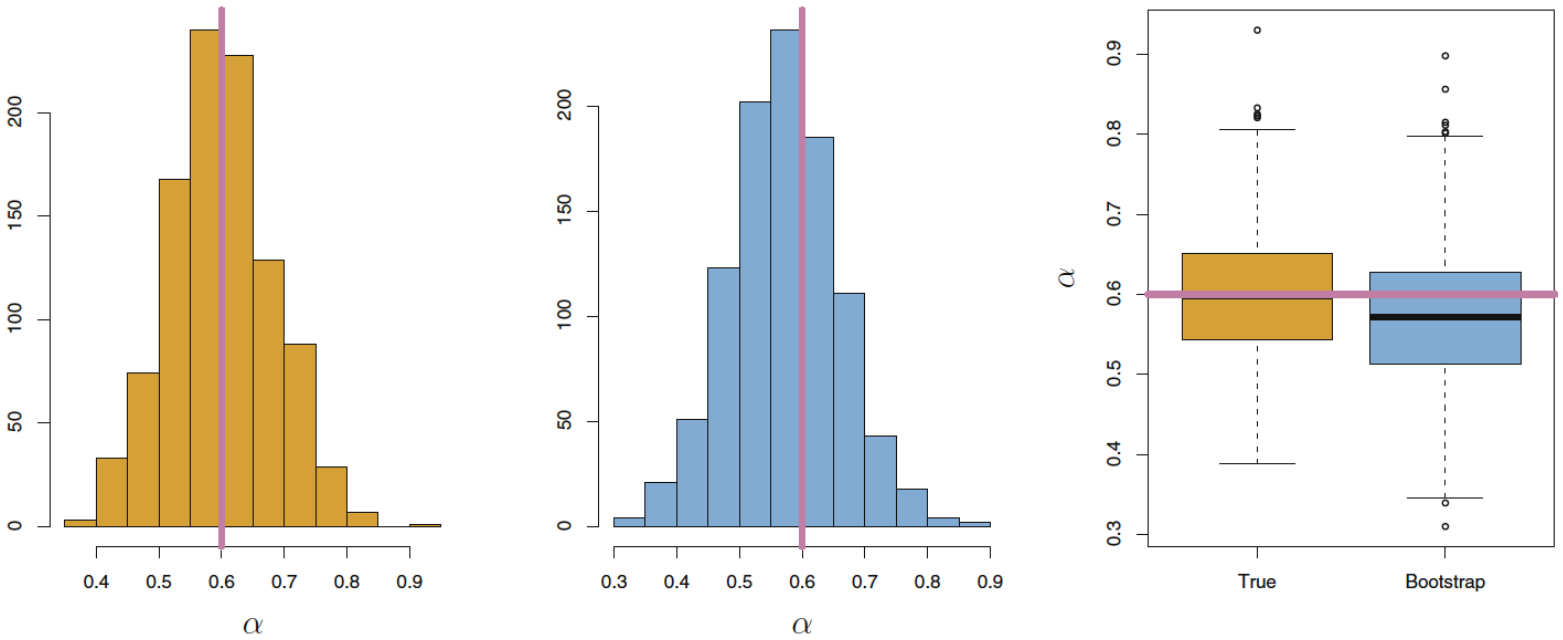


Figure: Left: A histogram of the estimates of α obtained by generating 1000 simulated data sets from the true population. Center: A histogram of the estimates of α obtained from 1000 bootstrap samples from a single data set. Right: The estimates of α displayed in the left and center panels are shown as boxplots. (Source: James et al., 2014).

Example (cont'd)

- For these simulations the parameters were set to $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$, and $\sigma_{XY} = 0.50$, so we know that the true value of α is 0.60
- The mean over all 1000 estimates for α is

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

which is close to 0.60. The standard deviation of the estimates is:

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

- This gives us a very good idea of the accuracy of $\hat{\alpha}$: $\mathrm{SE}(\hat{\alpha}) \approx 0.083$. So roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

Back to the real world ...

- The example above cannot be used, because we cannot generate new samples from the original population
- Bootstrapping mimics this process: we obtain distinct datasets by repeatedly sampling observations from the original data set with replacement
- Each of these bootstrap datasets is created by sampling with replacement, and is the same size as our original dataset

Example (with three observations)

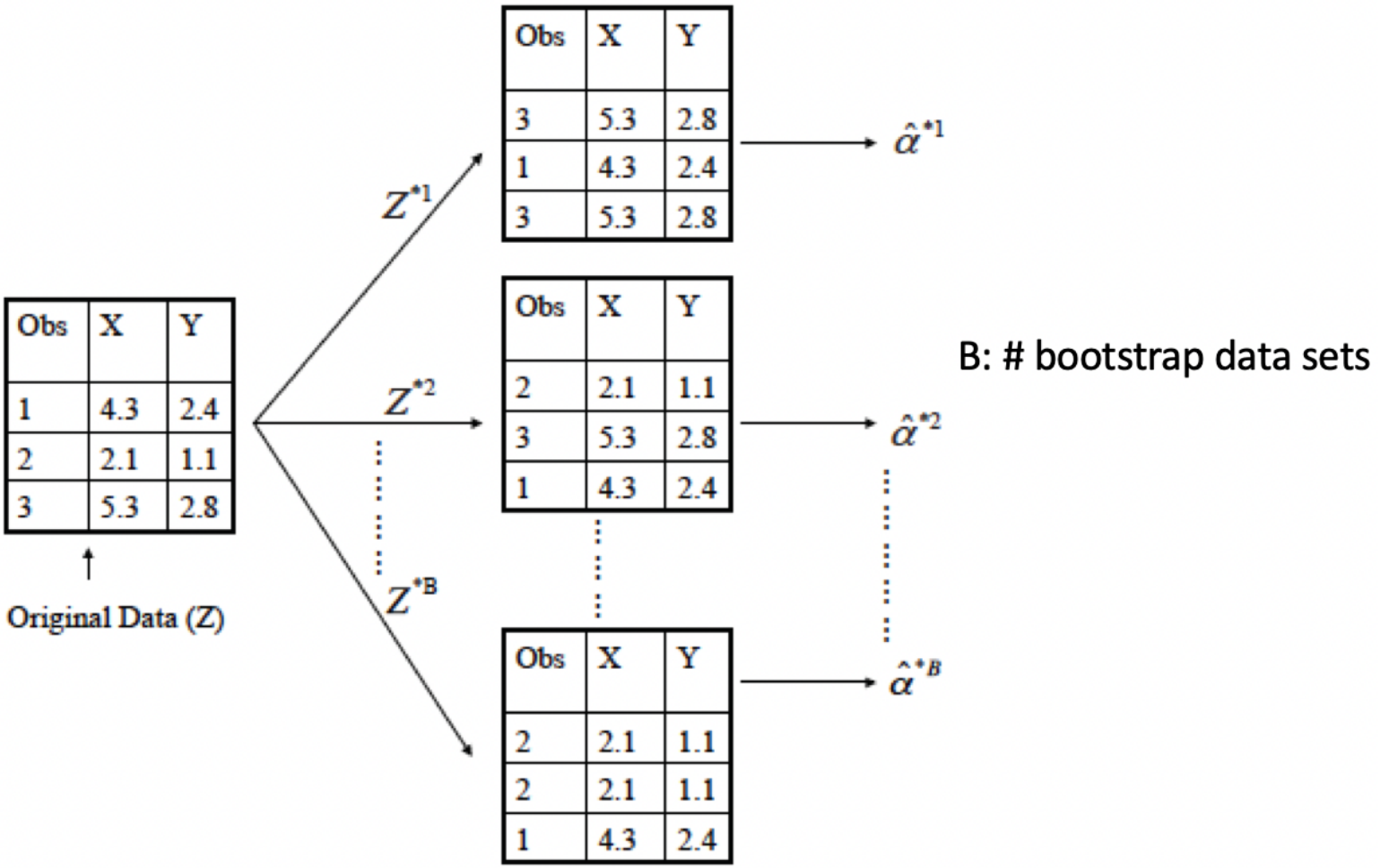


Figure: A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. (Source: James et al., 2014).

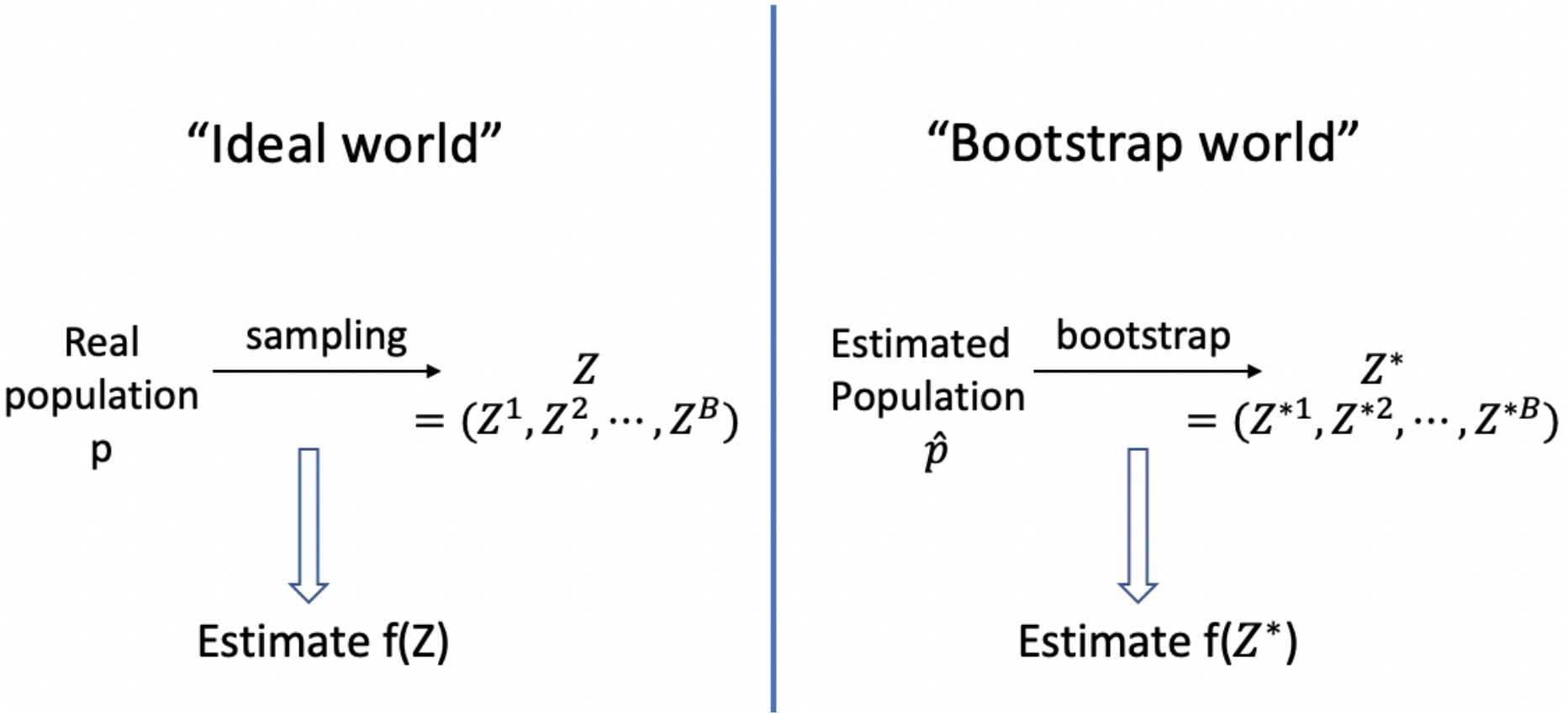
A few more points

- Denoting the first bootstrap data set by Z^{*1}, we use Z^{*1} to produce a new bootstrap estimate for α which we call $\hat{\alpha}^{*1}$
- The procedure is repeated B times (for example, 100 or 1000), in order to produce B different bootstrap datasets (Z^{*1}, Z^{*2}, ..., Z^{*B}), and B corresponding estimates $\hat{\alpha}^{*1}$, $\hat{\alpha}^{*2}$, ..., $\hat{\alpha}^{*B}$
- We estimate the standard error of these bootstrap estimates with the formula

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

where $\bar{\hat{\alpha}}^{*} = \frac{1}{B}\sum_{r=1}^{B}\hat{\alpha}^{*r}$

A general picture for the bootstrap

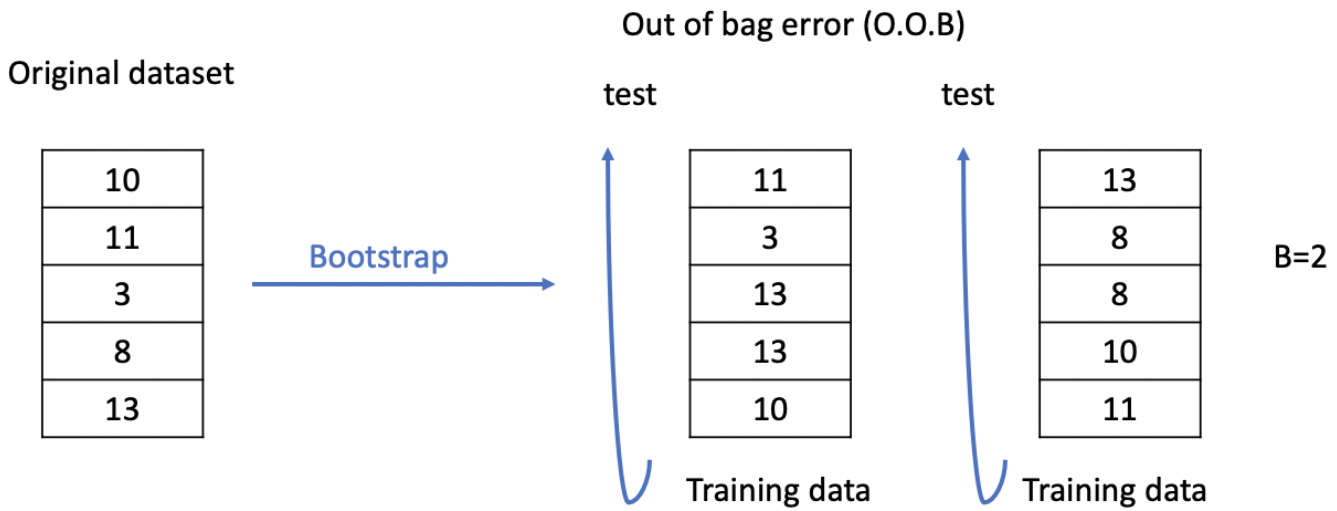


What is the difference between cross-validation and Bootstrapping?

- Bootstrap resamples with replacement, while cross-validation resamples without replacement
- The main goal of cross-validation is to measure, or generalize, the performance of a model
- Bootstrapping is used to establish empirical distribution functions for a widespread range of statistics

Can we use Bootstrapping to estimate the prediction error?

- We could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample
- But each bootstrap sample has significant overlap with the original data $\rightarrow$ the bootstrap will underestimate the true prediction error
- We can fix the problem by using the out-of-bootstrap estimate...



So, where can we use Bootstrapping (in CARTs)?

It can be used to tackle the bias-variance trade-off of some statistical learning methods, such as ... Decision Trees.

Decrease variance, increase bias:

1. Pruning
2. Bootstrapping $\rightarrow$ Bagging $\rightarrow$ Random forests

Outline

Bagging

- **Intuition:** Decision Trees suffer from high variance $\rightarrow$ if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different.
- **Bagging** (or **Bootstrap aggregation**) can be used to reduce the variance of a statistical learning method.

How does it work?

- Given a set of n independent observations $Z^1, Z^2, \ldots, Z^n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of observations is $\sigma^2/n \rightarrow$ averaging a set of observations reduces variance
- When learning a statistical model, we can therefore:
  - (Bootstrapping) Take many training sets from the population
  - (Ensemble of models) Build a separate model using each training dataset
  - Average the resulting predictions

Let's write this (slightly) more formally (for regression):

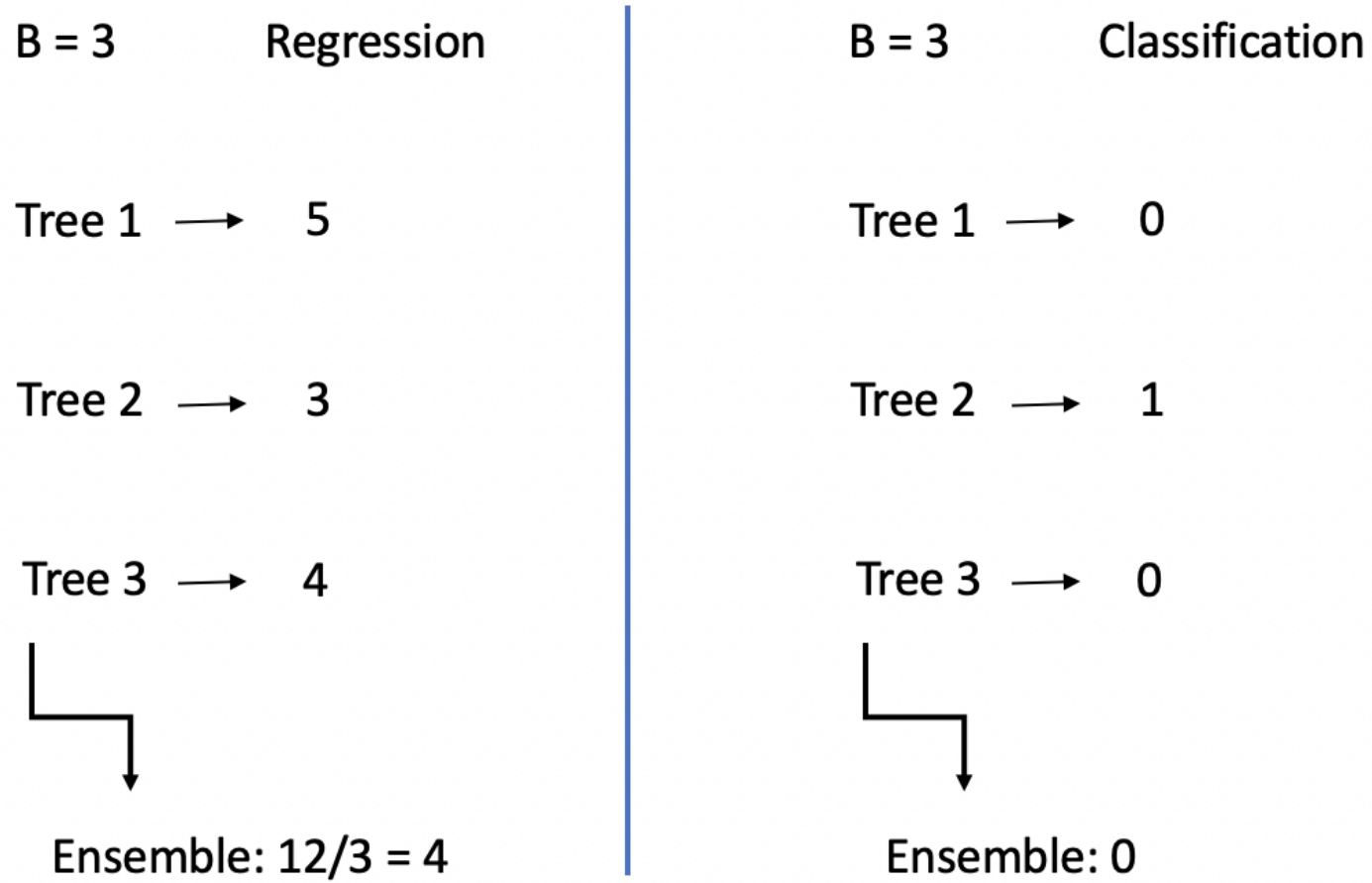
- Take repeated samples from the training dataset (if that's everything we have) $\rightarrow$ we generate B different bootstrapped training datasets
- We train model (e.g. CARTs without pruning) $f^{*b}(x)$ on the b-th bootstrapped training dataset (repeat for all B datasets) $\rightarrow$ B models in the ensemble
- We average all the predictions as follows

$$f_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} f^{*b}(x).$$

This entire process is called Bagging.

How does it work for Classification Trees?

We only change Step 3: we record the class predicted by each of the B trees and take a majority vote.



Out-of-Bag Error Estimation

- It turns out that there is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach.
- One can show that on average, each bagged tree makes use of around two-thirds of the observations.
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the ith observation.

Out-of-Bag Error Estimation (cont'd)

- More specifically:
  - To obtain a single prediction for the i-th observation, we can average these predicted responses (for regression) or can take a majority vote (for classification). This leads to a single OOB prediction for the i-th observation
  - An OOB prediction can be obtained in this way for each of the n observations, from which we compute the overall OOB MSE or classification error
- The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.
- When B is sufficiently large, we have a decent alternative to cross-validation

Back to R!

How can we implement Bagging in R? Options:

- Write our own code
  1. Bootstrap the "train" dataset
  2. `for i = 1:B`

     `CART(i) = rpart(y~., train = Z*i)`
  3. Averaging / Majority vote
- Use the function `bagging` (package `ipred`). But note this only works for Decision Trees.

  `model <- bagging(formula, data, coob=TRUE)` Fit a bagging model

  `print(model)` Short summary of model

  `pred <- predict(model, newdata, type)` Predict via bagging

Outline

Random Forests

Problem 1: CART suffers from high variance, low bias (pruning or bagging)
Problem 2: When building Decision Trees with Bagging, trees tend to be correlated.

Question: How do we "de-correlate" B trees in the ensemble?

Example:

- Suppose the training dataset has a strong predictor and a few moderately strong predictors
- Then, most of the bagged trees will use the strong predictor in the top split(s) $\rightarrow$ trees will be correlated, so we won't reduce variance much

Random Forests

Idea: When building the trees, we consider a random sample of m predictors from the full set of predictors p (at each split):

- For each split, we consider only a subset of predictors
- On average, $(p - m)$ predictors are not considered (in each model), so other predictors will have a chance
- This process decorrelates the trees

Recursive binary splitting with (randomly) $m < p$ predictors

Example: $p = 10$, $n = 10$, $\rightarrow$ 100

$m = 2$, $n = 10$, $\rightarrow$ 20

Algorithm

Main steps:

- Generate B boostrapped training datasets
- For each dataset, train a Decision Tree. At each split, use a subset m of the p available predictors
- Average the predictions from the B trees. (For classification, use majority voting.)

On the value of m

The fundamental difference between Bagging and Random Forests stands in the subset of predictors m:

- If $m = p$, then there is no difference between the two methods
- Recommended values of m:
  - Regression: $m = p/3$
  - Classification: $m = \sqrt{p}$
- Note that these values were found experimentally, so there is no theoretical guarantee they will provide the best performance on all datasets

Hyperparameters tuning

It is common practice to explore the effect of the hyperparameters value on the performance of Random Forests. To recap, we have the following parameters:

- Number of trees, B
- Number of predictors used at each split, $m \leq p$
- (number of points in each terminal leaf)

There are no optimization routines to find their values. We typically use grid search, or similar.

Back to R!

To learn a Random Forest, we will use the function `randomForest`, implemented in the package ... `randomForest`:

`forest <- randomForest(formula, data, ntree, mtry, ...)` Fit

`predictforest <- predict(forest,newdata,type)` Predict

`importance(forest)` or `varImpPlot(forest)` Variable importance

`varUsed(forest, by.tree=FALSE, count=TRUE)` Frequencies of variables

`forest$err.rate[ntree,1]` OOB error rate

Advantages and Disadvantages of Random Forests

Pros:

- Better bias-variance trade-off than CARTs
- Higher accuracy (on the test dataset)

Cons:

- Less interpretable
- Higher computational requirements

References

- James et al. (2014) *An Introduction to Statistical Learning with Applications in R*, Springer, 2014. Chapter 5.2 and 8.2.
  Chapter 5.2: Bootstrapping
  Chapter 8.2: Bagging, Random forests