The bootstrap

Introduction to resampling methods
STA 380

Reference: *Introduction to Statistical Learning* Chapter 5.2

Outline

- Uncertainty quantification as a "what if?"
- Sampling distributions
- The bootstrap
- Bootstrapped confidence intervals
- Bootstrapped versus plug-in standard errors
- Bonus topic: the parametric bootstrap

Quantifying uncertainty

From the New England Journal of Medicine in 2006:

> We randomly assigned patients with resectable adenocarcinoma of the stomach, esophagogastric junction, or lower esophagus to either perioperative chemotherapy and surgery (250 patients) or surgery alone (253 patients)….
>
> With a median follow-up of four years, 149 patients in the perioperative-chemotherapy group and 170 in the surgery group had died. As compared with the surgery group, the perioperative-chemotherapy group had a higher likelihood of overall survival (five-year survival rate, 36 percent vs. 23 percent).

Quantifying uncertainty

Conclusion:

- Chemotherapy patients are **13%** more likely to survive past 5 years.

Quantifying uncertainty

Conclusion:

- Chemotherapy patients are **13%** more likely to survive past 5 years.

Not so fast! In statistics, we ask "what if?" a lot:

- What if the randomization of patients just happened, by chance, to assign more of the healthier patients to the chemo group?
- Or what if the physicians running the trial had enrolled a different sample of patients from the same clinical population?

Quantifying uncertainty

Conclusion:

- Chemotherapy patients are **13%** more likely to survive past 5 years.

Always remember two basic facts about samples:

- *All numbers are wrong*: any quantity derived from a sample is just a *guess* of the corresponding population-level quantity.
- *A guess is useless without an error bar*: an estimate of how wrong we expect the guess to be.

Quantifying uncertainty

Conclusion:

- Chemotherapy patients are **13% ± ?** more likely to survive past 5 years, with **??%** confidence.

By "quantifying uncertainty," we mean filling in the blanks.

Quantifying uncertainty

In data science, we equate trustworthiness with *stability*:

- If our data had been different merely due to chance, would our answer have been different, too?
- Or would the answer have been stable, even with different data?

Confidence in your estimates $\iff$ Stability of those estimates under the influence of chance

Quantifying uncertainty

For example:

- If doctors had taken a different sample of 503 cancer patients and gotten a drastically different estimate of the new treatment's effect, then the original estimate isn't very trustworthy.
- If, on the other hand, pretty much any sample of 503 patients would have led to the same estimates, then their answer for *this particular* subset of 503 is probably accurate.

Some notation

Suppose we are trying to estimate some population-level feature of interest, θ. This might be something very complicated!

So we take a sample from the population: $X_1, X_2, \ldots, X_N$. We use the data to form an estimate $\hat{\theta}_N$ of the parameter. Key insight: $\hat{\theta}_N$ is a random variable.

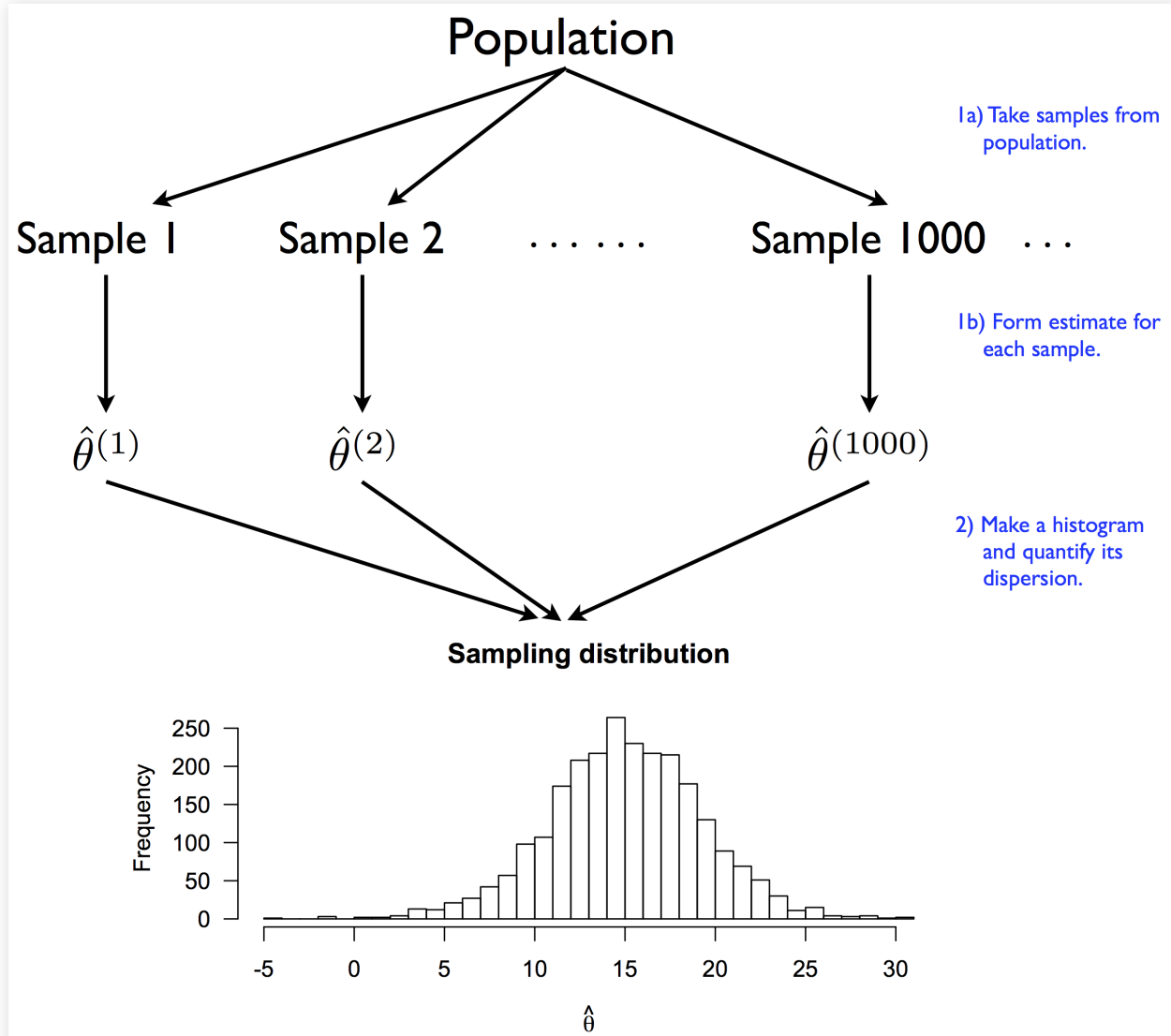
Some notation

Suppose we are trying to estimate some population-level feature of interest, θ. This might be something very complicated!

So we take a sample from the population: $X_1, X_2, \ldots, X_N$. We use the data to form an estimate $\hat{\theta}_N$ of the parameter. Key insight: $\hat{\theta}_N$ is a random variable.

Now imagine repeating this process thousands of times! Since $\hat{\theta}_N$ is a random variable, it has a probability distribution: the *sampling distribution*.

Some notation



Standard error

Standard error: the standard deviation of an estimator's sampling distribution:

$$\begin{aligned} \text{se}(\hat{\theta}_N) &= \sqrt{\text{var}(\hat{\theta}_N)} \\ &= \sqrt{E[(\hat{\theta}_N - \bar{\theta}_N)^2]} \\ &= \text{Typical deviation of } \hat{\theta}_N \text{ from its average} \end{aligned}$$

"If I were to take repeated samples from the population and use this estimator for every sample, how much does the answer vary, on average?"

An analogy: manufacturing tolerance

Think about ordering a ceramic bowl off Etsy, made by an artist who uses one of those cool pottery wheels:

- Across many weeks of manufacturing, the artist's bowls have an average diameter of 8".
- But individual bowls vary from the average by about 0.25", since they're made by hand.
- So you should expect that your specific bowl will be somewhere in the vicinity of 8" $\pm$ 0.25".

Don't count on using the bowl for anything that requires greater precision!

Standard errors

Now think about forming an estimate of θ from a noisy sample:

- On average across many samples, my estimator $\hat{\theta}_N$ is close to the right answer (θ).
- But individual estimates vary from the average by about $\mathrm{se}(\hat{\theta}_N)$, due to sampling variability.
- So I expect that the right answer is somewhere in the vicinity of $\hat{\theta}_N \pm \mathrm{se}(\hat{\theta}_N)$.

Don't reach any scientific conclusions that require greater precision!

Standard errors

But there's a problem here…

- Knowing the standard error requires knowing what happens across many separate samples.
- But we've only got our one sample!
- So how can we ever calculate the standard error?

Standard errors

> Two roads diverged in a yellow wood
> And sorry I could not travel both
> And be one traveler, long I stood
> And looked down one as far as I could
> To where it bent in the undergrowth…
>
> –Robert Frost, *The Road Not Taken*, 1916

Quantifying our uncertainty would seem to require knowing all the roads not taken—an impossible task.

The bootstrap

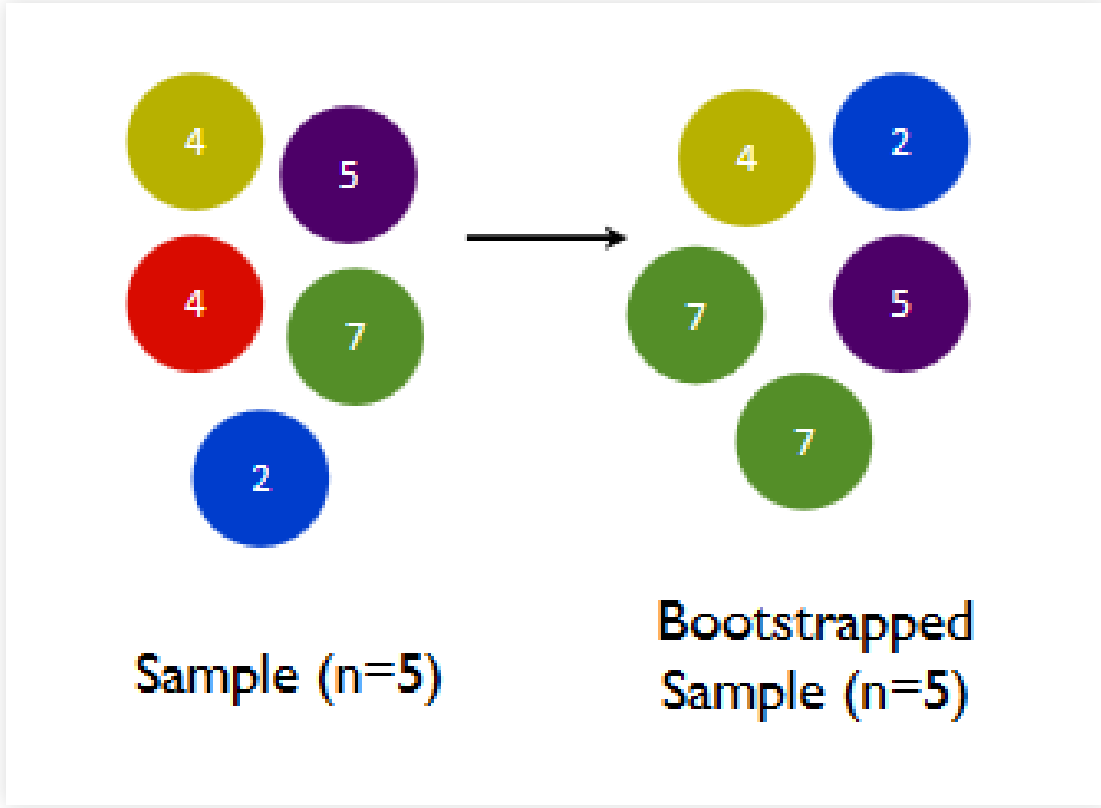
Problem: we can't take repeated samples of size N from the population, to see how our estimate changes across samples.

Seemingly hacky solution: take repeated samples of size N, with replacement, *from the sample itself*, and see how our estimate changes across samples. This is something we can easily simulate on a computer.

Basically, we pretend that our sample is the whole population and we charge ahead! This is called *bootstrap resampling*, or just *bootstrapping*.

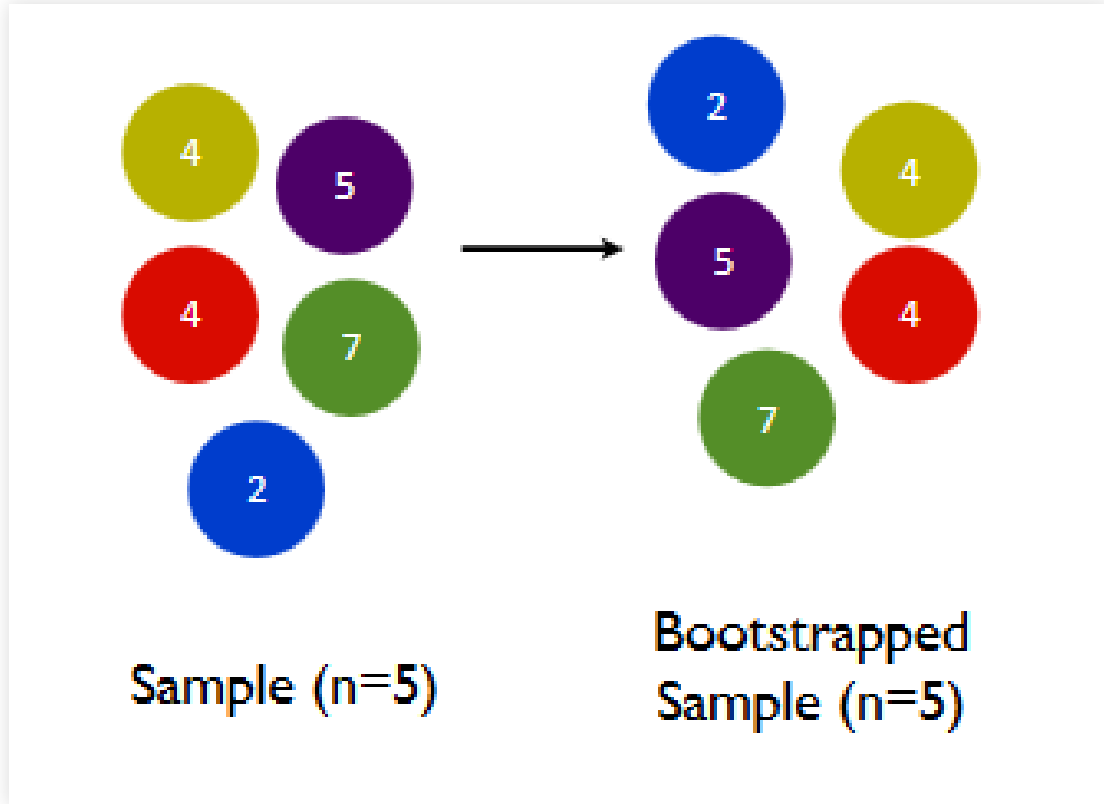
Sampling with replacement is key!

Bootstrapped sample 1



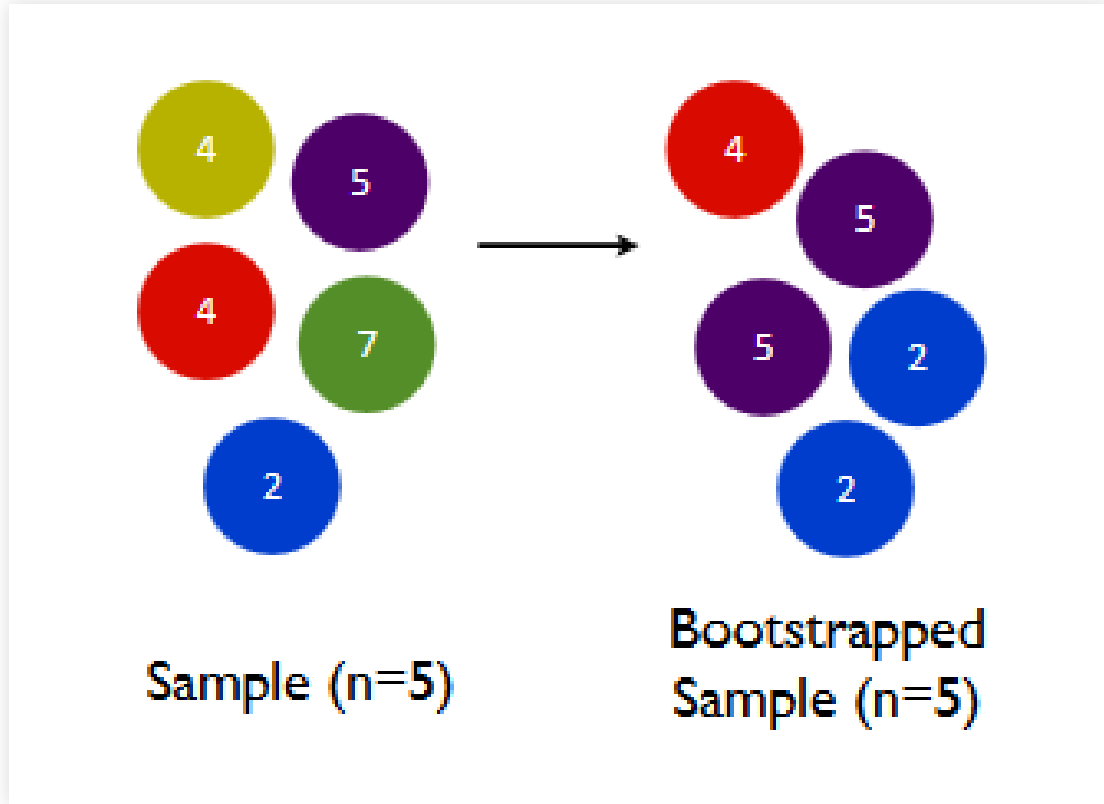
Sampling with replacement is key!

Bootstrapped sample 2

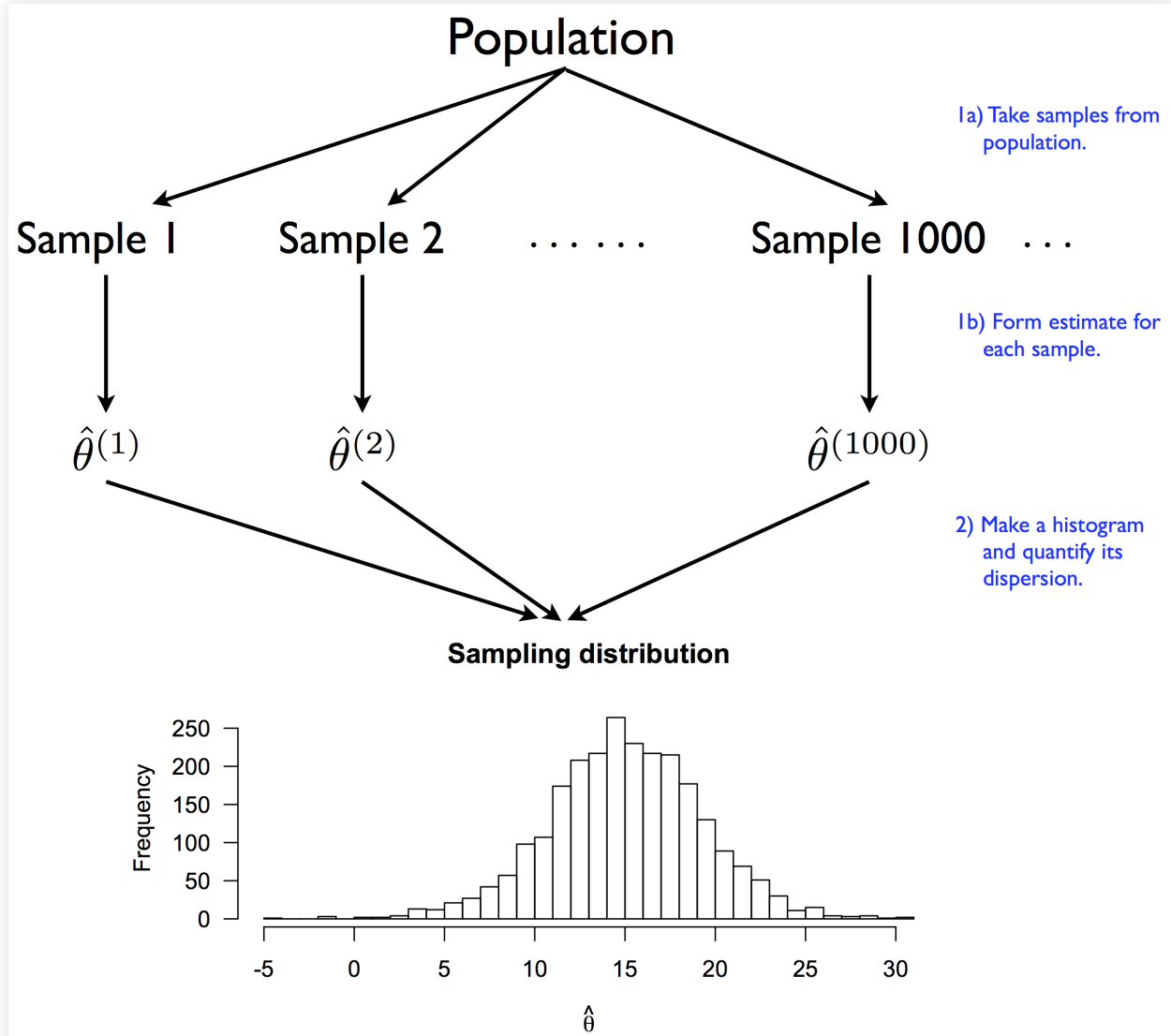


Sampling with replacement is key!

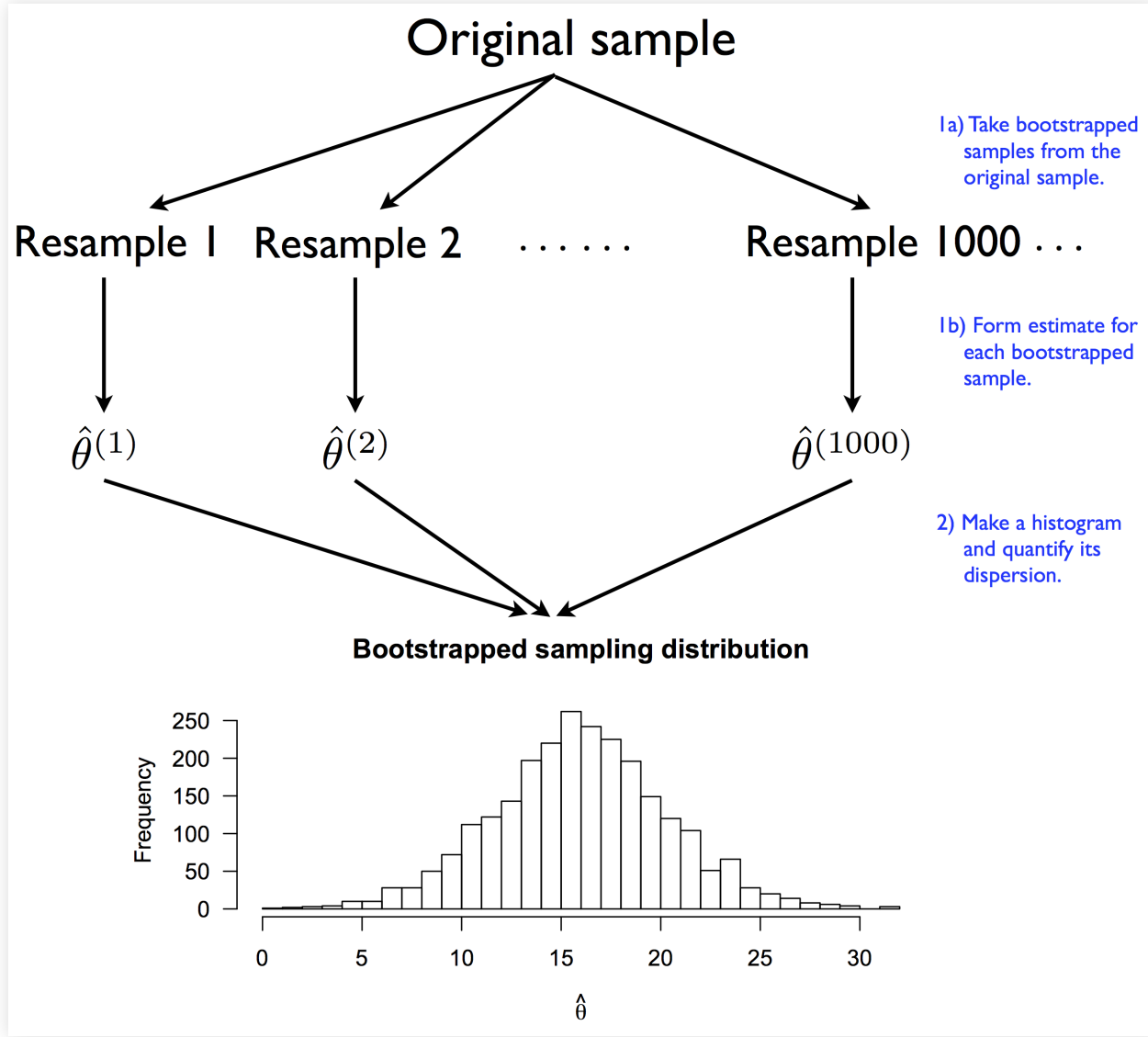
Bootstrapped sample 3



The true sampling distribution



The bootstrapped sampling distribution



The bootstrapped sampling distribution

- Each bootstrapped sample has its own pattern of duplicates and omissions from the original sample.
- These duplicates and omissions create variability in $\hat{\theta}$ from one bootstrapped sample to the next.
- This variability mimics the *true* sampling variability you'd expect to see across real repeated samples from the population.

Bootstrapping: pseudo-code

- Start with your original sample $S = \{X_1, \ldots, X_N\}$ and original estimate $\hat{\theta}_N$.

- For $b = 1, \ldots, B$:

  1. Take a bootstrapped sample $S^{(b)} = \{X_1^{(b)}, \ldots, X_N^{(b)}\}$
  2. Use $S^{(b)}$ to re-form the estimate $\hat{\theta}_N^{(b)}$.

- Result: a set of B different estimates $\hat{\theta}_N^{(1)}, \hat{\theta}_N^{(b)}, \ldots, \hat{\theta}_N^{(B)}$ that approximate the sampling distribution of $\hat{\theta}_N$.

Then what?

Calculate the *bootstrapped standard error* as the standard deviation of the bootstrapped estimates:

$$\hat{se}(\hat{\theta}_N) = \text{std dev}\left(\hat{\theta}_N^{(1)}, \hat{\theta}_N^{(b)}, \ldots, \hat{\theta}_N^{(B)}\right)$$

This isn't the true standard error, but it's often a good approximation!

Then what?

Or form an interval estimate: a range of plausible values for the parameter of interest.

The simplest way is to use the quantiles (e.g. the 2.5 and 97.5 percentiles):

$$\theta \in (q_{2.5}, q_{97.5})$$

There's some very hairy mathematics showing that these intervals will contain the true value approximately 95% of the time (or whatever your coverage level is).

Example

Let's dig in to some R code and data: `creatinine_bootstrap.R` and `creatinine.csv` (both on class website).

We'll bootstrap two estimators:

- the sample mean
- the OLS estimate of a slope

Summary

- The *standard error* is the standard deviation of the sampling distribution.
- Roughly speaking, it answers the question: how far off do I expect my estimate to be from the truth?
- A practical way of estimating the standard error is by *bootstrapping*: repeatedly re-sampling with replacement from the original sample, and re-calculating the estimate each time.

Example 1: nonparametric regression

Let's see this example in nonparametric regression, where

$$y_i = f(x_i) + e_i$$

Suppose we use a nonparametric method to form an estimate $\hat{f}(x)$ (e.g. using K-nearest neighbors), and we want to quantify our uncertainty about how well we've estimated the true f.

Example 1: nonparametric regression

- Question: "how might my estimate $\hat{f}(x)$ have been different if I'd seen a different sample of (x_i, y_i) pairs from the same population?"
- Assumption: each (x_i, y_i) is a random sample from a joint distribution $P(x, y)$ describing the population from which your sample was drawn.
- Problem: We don't know $P(x, y)$.
- Solution: Approximate $P(x, y)$ by $\hat{P}(x, y)$, the empirical joint distribution of the data in your sample.
- Key fact for implementation: sampling from $\hat{P}(x, y)$ is equivalent to sampling with replacement from the original sample.

Example 1: nonparametric regression

This leads to the following algorithm. For b = 1 to B:

- Construct a bootstrap sample from $\hat{P}(x, y)$ by sampling N pairs (x_i, y_i) *with replacement* from the original sample.
- Refit the model to each bootstrapped sample, giving you $\hat{f}^{(b)}$.

This gives us B draws from the bootstrapped sampling distribution of $\hat{f}(x)$.

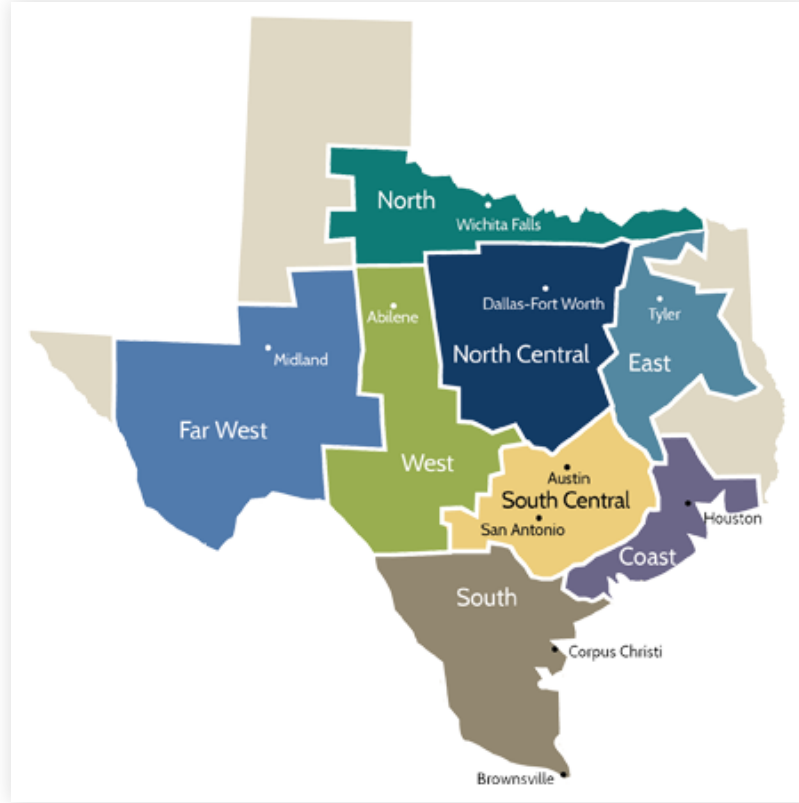
Use these draws to form (approximate) confidence intervals and standard errors for $f(x)$.

Predicting electricity demand

Predicting electricity demand

ERCOT operates the electricity grid for 75% of Texas.

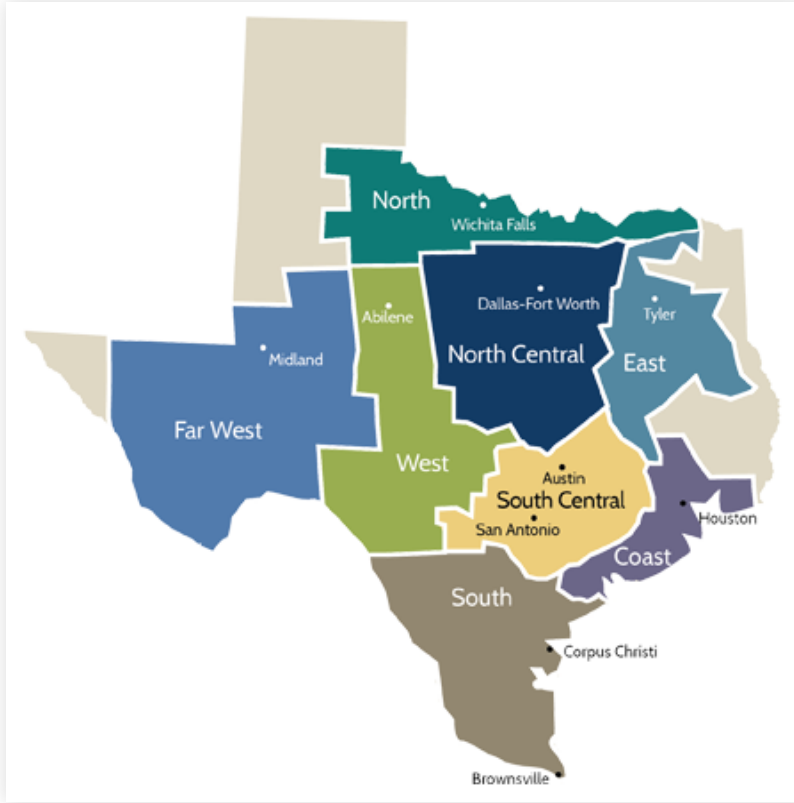
The 8 ERCOT regions are shown at right.



Example: predicting electricity demand

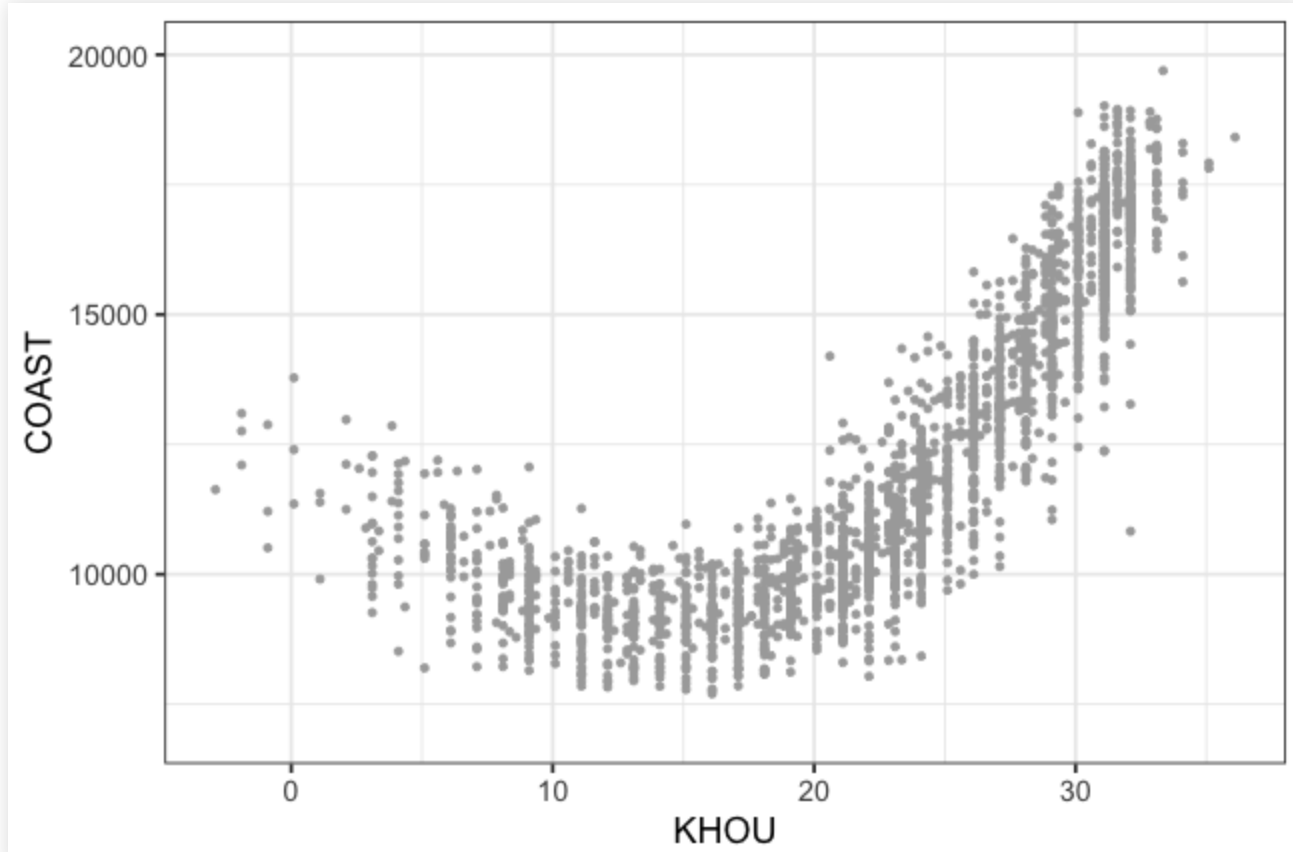
We'll focus on a basic prediction task:

- y = demand (megawatts) in the Coast region at 3 PM, every day from 2010-2016.
- x = average daily tempature measured at Houston's Hobby Airport (degrees Celsius)
- Date sources: scraped from the ERCOT website and the National Weather Service



Example: predicting electricity demand

```
ggplot(data = loadhou) +
  geom_point(mapping = aes(x = KHOU, y = COAST), color='darkgrey') +
  ylim(7000, 20000)
```



Example: predicting electricity demand

Suppose we want to know $f(5)$ and $f(25)$, i.e. the expected values of `COAST` when `KHOU` $=$ `5` and `KHOU` $=$ `25`, respectively. Let's bootstrap a KNN model, with $K = 40$:

```
library(mosaic)
library(FNN)

X_test = data.frame(KHOU=c(5,25))
boot20 = do(500)*{
  loadhou_boot = resample(loadhou)  # construct a boostrap sample
  X_boot = select(loadhou_boot, KHOU)
  y_boot = select(loadhou_boot, COAST)
  knn20_boot = knn.reg(X_boot, X_test, y_boot, k=40)
  knn20_boot$pred
}
head(boot20, 3)  # first column is f(5), second is f(25)
```

```
        V1       V2
1 10708.95 11882.67
2 10802.91 11997.82
3 10513.39 11812.59
```

Example: predicting electricity demand

Now we can calculate standard errors and/or confidence intervals.

- Standard errors: take the standard deviation of each column.

```
se_hat = apply(boot20, 2, sd)
se_hat
```

```
      V1       V2
172.5259 156.6715
```

- Confidence intervals: calculate quantiles for each column

```
apply(boot20, 2, quantile, probs=c(0.025, 0.975))
```

```
             V1       V2
2.5%  10283.88 11532.70
97.5% 10974.92 12125.45
```

Example: predicting electricity demand

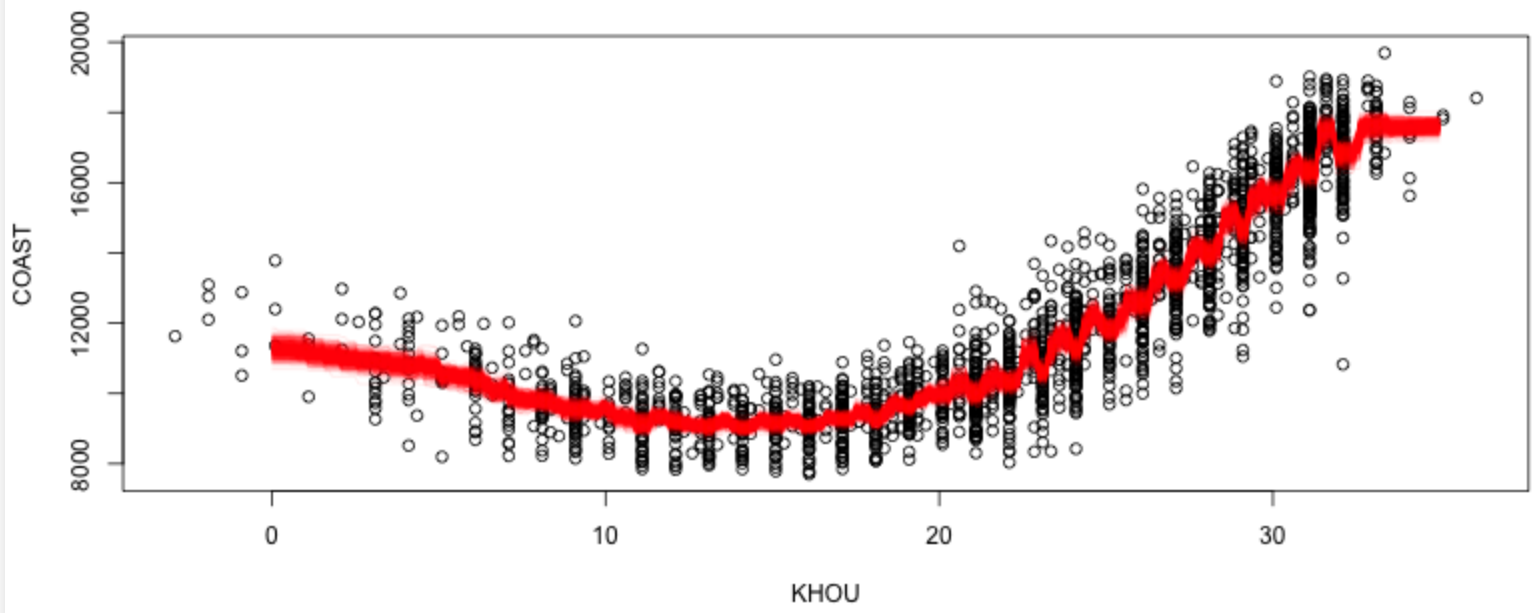
- Shortcut:

```
confint(boot20)
```

```
  name    lower    upper level     method estimate
1   V1 10283.88 10974.92  0.95 percentile 10638.78
2   V2 11532.70 12125.45  0.95 percentile 11906.23
```

A spaghetti plot

```
X_test = data.frame(KHOU=seq(0, 35, by=0.1))
plot(COAST ~ KHOU, data=loadhou)
for(i in 1:500) {
  loadhou_boot = resample(loadhou)  # construct a boostrap sample
  X_boot = select(loadhou_boot, KHOU)
  y_boot = select(loadhou_boot, COAST)
  knn20_boot = knn.reg(X_boot, X_test, y_boot, k=40)
  knn20_boot$pred
  lines(X_test$KHOU,  knn20_boot$pred, col=rgb(1, 0, 0, 0.1))
}
```



Example 2: portfolio modeling

Suppose that you're trying to construct a portfolio: that is, to decide how to allocate your wealth among D financial assets. Things you want might to track include:

- the expected value of your portfolio at some point in the future (e.g. when you retire).
- the variance of your portfolio's value at some point in the future.
- the probability of losing some specific amount of money (10K, 20% of total value, etc)
- some measure of "tail risk," i.e. what a bad week/month/year might look like.

Key idea: **use the bootstrap to simulate portfolio performance.**

Example 2: portfolio modeling

Notation:

- Let T be our investing horizon (e.g. T = 20 days, T = 40 years, etc), and let t index discrete time steps along the way.

- Let $X_{t,j}$ be the value of asset $j = 1, \ldots, D$ at time period t.

- Let $R_{t,j}$ be the *return* of asset j in period t, so that we have the following recursive update:

$$X_{t,j} = X_{t-1,j} \cdot (1 + R_{t,j})$$

Example 2: portfolio modeling

Notation:

- A portfolio is a set of investment weights over assets: $(w_{t1}, w_{t2}, \ldots, w_{tD})$. Note: these weights might be fixed, or they might change over time.

- The value of your portfolio is the weighted sum of the value of your assets:

$$W_t = \sum_{j=1}^{D} w_{t,j} X_{t,j}$$

Example 2: portfolio modeling

We care about W_T: the random variable describing your terminal wealth after T investment periods.

Problem: this random variable is a super-complicated, nonlinear function of $T \times D$ individual asset returns:

$$W_T = f(R) \quad \text{where} \quad R = \{R_{t,j} : t = 1, \ldots, T; j = 1, \ldots, D\}$$

Example 2: portfolio modeling

If we knew the asset returns, we could evaluate this function recursively, starting with initial wealth W_0 at time $t = 0$ and sweeping through time $t = T$:

Starting with initial wealth $X_{1,j}^{(i)}$ in each asset, we sweep through from $t = 1$ to $t = T$:

$$X_{t,j}^{(f)} = X_{t,j}^{(i)} \cdot (1 + R_{t,j}) \qquad \text{(Update each asset)}$$

$$W_t = \sum_{j=1}^{D} w_{t,j} X_{t,j}^{(f)} \qquad \text{(Sum over assets)}$$

$$X_{t+1,j}^{(i)} = w_{t+1,j} \cdot W_t \qquad \text{(Rebalance)}$$

Example 2: portfolio modeling

But of course, we don't know the asset returns! This suggests that we should use a Monte Carlo simulation, where we repeat the following `for` loop many times.

For $t = 1, \ldots, T$:

1. Simulate $R_t = (R_{t1}, R_{t2}, \ldots, R_{tD})$ from the joint probability distribution of asset returns at time t.
2. Use these returns to update $X_{j,t}$, the value of your holdings in each asset at step t.
3. Rebalance your portfolio to the target allocation.

The precise math of the update and rebalance steps are on the previous slide.

Example 2: portfolio modeling

The difficult step here is (1): simulate a high-dimensional vector of asset returns from its joint probability distribution.

- very complicated correlation structure
- probably not something simple like a Gaussian!

In general, using simple parametric probability models (e.g. multivariate Gaussian) to describe high-dimensional joint distributions is a very dicey proposition.

A simple approach: bootstrap resampling

Suppose we have M past samples of the asset returns, stacked in a matrix:

$$R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1D} \\ R_{21} & R_{22} & \cdots & R_{2D} \\ \vdots & & & \\ R_{M1} & R_{M2} & \cdots & R_{MD} \end{pmatrix}$$

where R_{tj} is the return of asset j in period t.

A simple approach: bootstrap resampling

The key idea of bootstrap resampling is the following:

- We may not be able to describe what the joint distribution $P(R_1, \ldots, R_D)$ is.
- But *we do know that every row of this R matrix is a sample from this joint distribution.*
- So instead of sampling from some theoretical joint distribution, we will sample from the sample—i.e. we will bootstrap the past data.
- Thus every time we need a new draw from the joint distribution $P(R_1, \ldots, R_D)$, we randomly sample (with replacement) a single row of R.

A simple approach: bootstrap resampling

Thus our Monte Carlo simulation looks like the following at each draw.

For $t = 1, \ldots, T$:

1. Simulate $R_t = (R_{t1}, R_{t2}, \ldots, R_{tD})$ by drawing a whole row, with replacement, from our matrix of past returns.
2. Use these returns to update $X_{j,t}$, the value of your holdings in each asset at step t.
3. Rebalance your portfolio to the target allocation.

Example

Let's go to the R code! See `portfolio.R` on the website.

Key discussion question

Why do we draw an entire row of R at a time?