The bootstrap

SDS 323
James Scott

Reference: *Data Science: A Gentle Introduction* Chapter 5; *Statistical Learning* Ch 5.2

Outline

- Sampling distributions
- Bias and standard error
- The bootstrap
- Bootstrapped confidence intervals
- Bootstrapped versus plug-in standard errors
- Application: the bootstrap in supervised learning
- Application: bootstrapping for risk estimation

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" or "statistical inference," we mean filling in the blanks.

Quantifying uncertainty

In stats, 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	$\Longleftrightarrow$	Stability of those estimates under the influence of chance

To assess stability, you have to contemplate the possibility of **alternate data universes.**

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.

Let's work through a thought experiment…

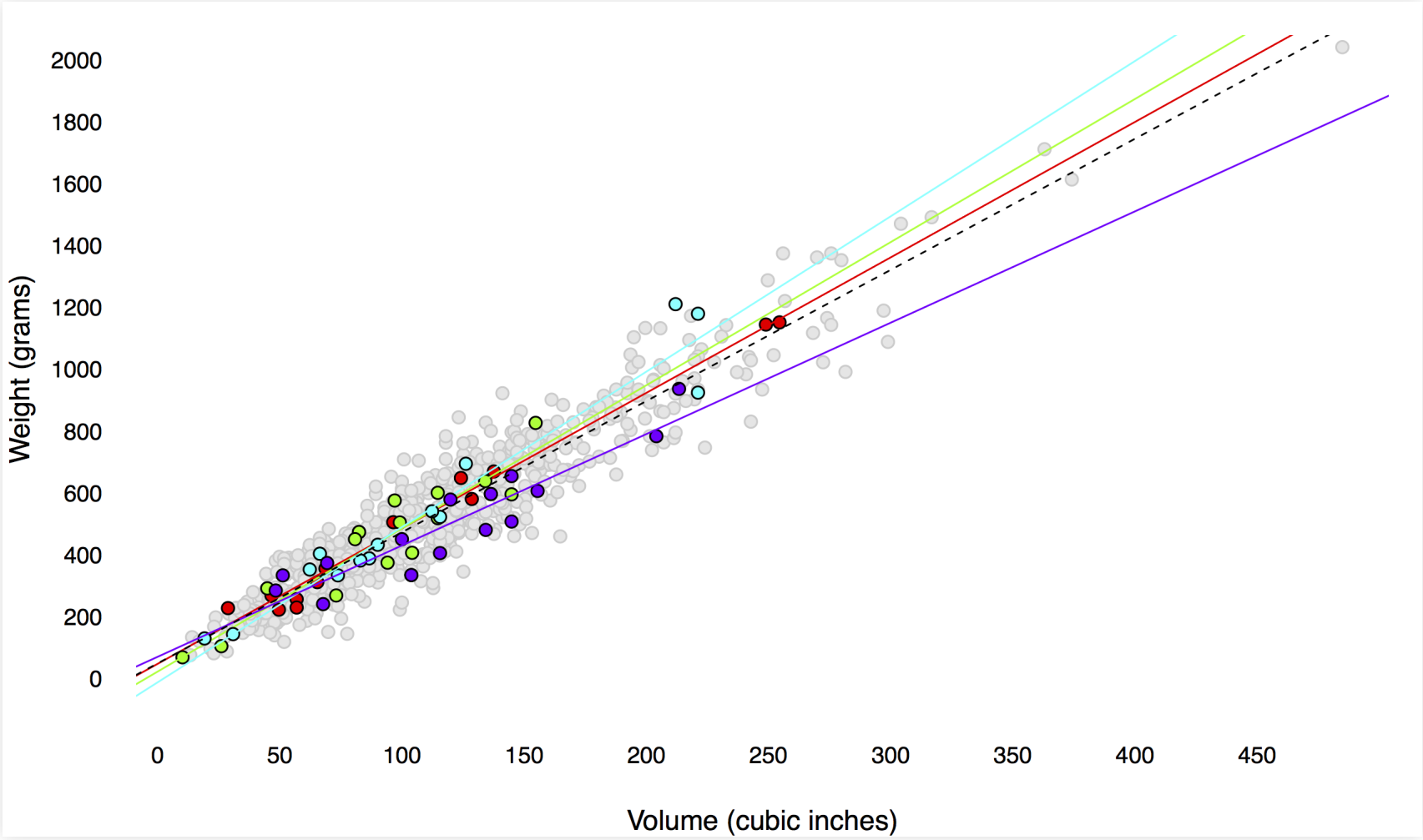
Kolmorogov goes fishing...

Kolmorogov goes fishing...

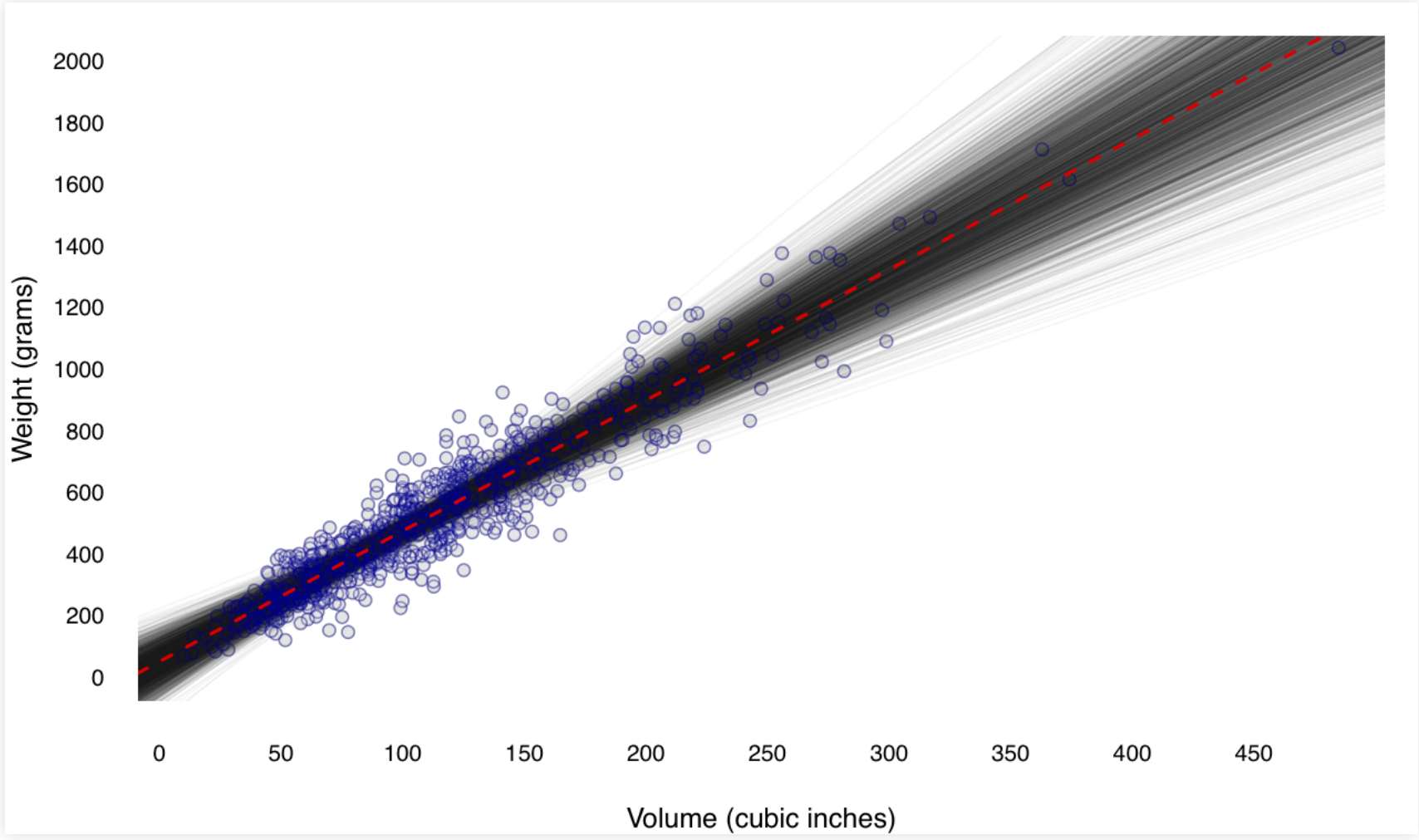
Imagine Andrey Kolmorogov on four-day fishing trip.

- The lake is home to a very large population of fish of varying size and weight.
- On each day, Kolmorogov takes a random sample of size $N = 15$ from this population—that is, he catches (and releases) 15 fish.
- He records the weight and approximate volume of each fish.
- He uses each day's catch to compute a different estimate of the volume–weight relationship for **all** fish in the lake.

Kolmorogov goes fishing...



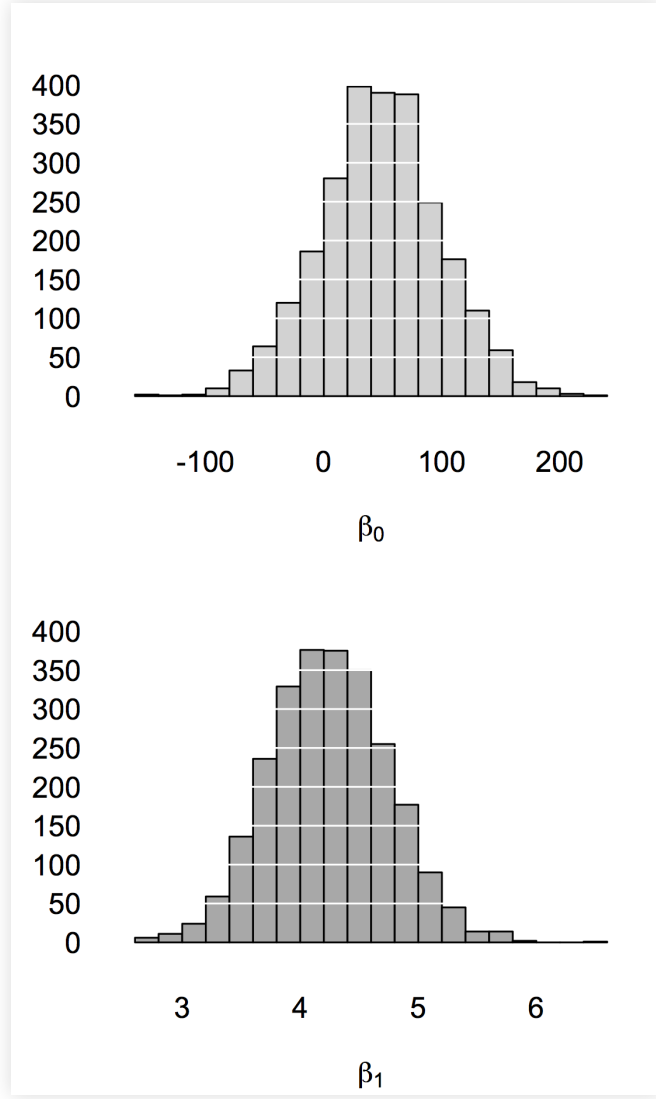
Kolmorogov goes fishing...



Kolmorogov goes fishing...

At right we see the *sampling distribution* for both β_0 and β_1.

- Each is centered on the true population value.
- The spread of each histogram tells us how *variable* our estimates are from one sample to the next.



Some notation

Suppose we are trying to estimate some population-level quantity θ: the *parameter* of interest.

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

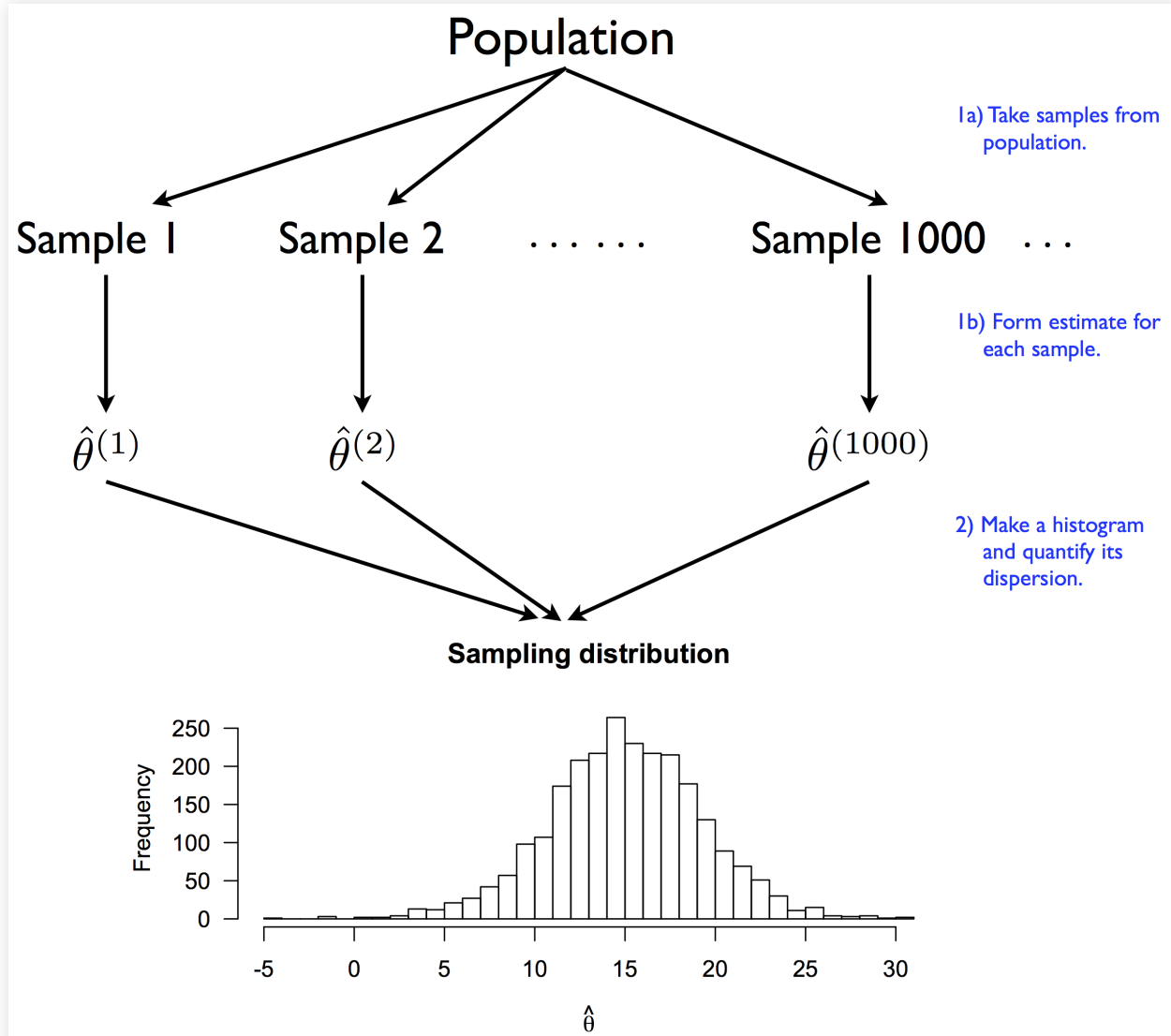
Suppose we are trying to estimate some population-level quantity θ: the *parameter* of interest.

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.

Some notation



Key definitions

Estimator: any method for estimating the value of a parameter (e.g. sample mean, sample proportion, slope of OLS line, etc).

Sampling distribution: the probability distribution of an estimator $\hat{\theta}_N$ under repeated samples of size N.

Bias: Let $\bar{\theta}_N = E(\hat{\theta}_N)$ be the mean of the sampling distribution. The bias of $\hat{\theta}_N$ is $(\bar{\theta}_N - \theta)$: the difference between the average answer and the truth.

Unbiased estimator: $(\bar{\theta}_N - \theta) = 0$.

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?"

Standard error

If an estimator is unbiased, then $\bar{\theta}_N = \theta$, so

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

"If I were to take repeated samples from the population and use this estimator for every sample, how big of an error do I make, on average?"

An analogy

THE WALL STREET JOURNAL.

HOMES

Farmhouse Fever Sweeps City Homes

Forget minimalism. Inspired by Chip and Joanna Gaines' 'Fixer Upper,' the urban farmhouse aesthetic is now in vogue; homes in such rustic styles can push up list prices as much as 30%

An analogy

This is why doctors and lawyers are buying “farmhouses.”

An analogy

Chip and Joanna lifestyle item #1: the farmhouse sink

The farmhouse idyll...

And the fine print

FEATURES

- Acid and alkali resistant glazed surfaces
- Suitable for waste disposal units or basket strainer waste
- Standard 3 1/2" diameter US drain opening (centered)
- Sink measures 30" x 18"
- Due to ±2% dimensions, it is recommended that the cabinet maker wait until your sink is delivered to make the cabinet because no template is available
- Includes “Shaws” blue badge
- Weight 155 lbs.
- 33" minimum cabinet size
- Hygienic due to antibacterial properties

Bowl options	Single bowl
Hole Configuration	1
Mounting Location	Deck
Installation Type	Apron front
Warranty	1-year limited warranty, 10-year limited warranty on fading/staining

Manufacturing tolerances

- On average across many weeks of manufacturing, the fancy sink has width equal to 30".
- But individual sinks vary from the average by about 0.5", due to manufacturing variability.
- So I expect that my specific sink will be somewhere in the vicinity of 30" $\pm$ 0.5".

Don't make any lifestyle choices that require greater precision!

Standard errors

- On average across many samples, my estimator $\hat{\theta}_N$ is equal 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 $\theta_N \pm \mathrm{se}(\theta_N)$.

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

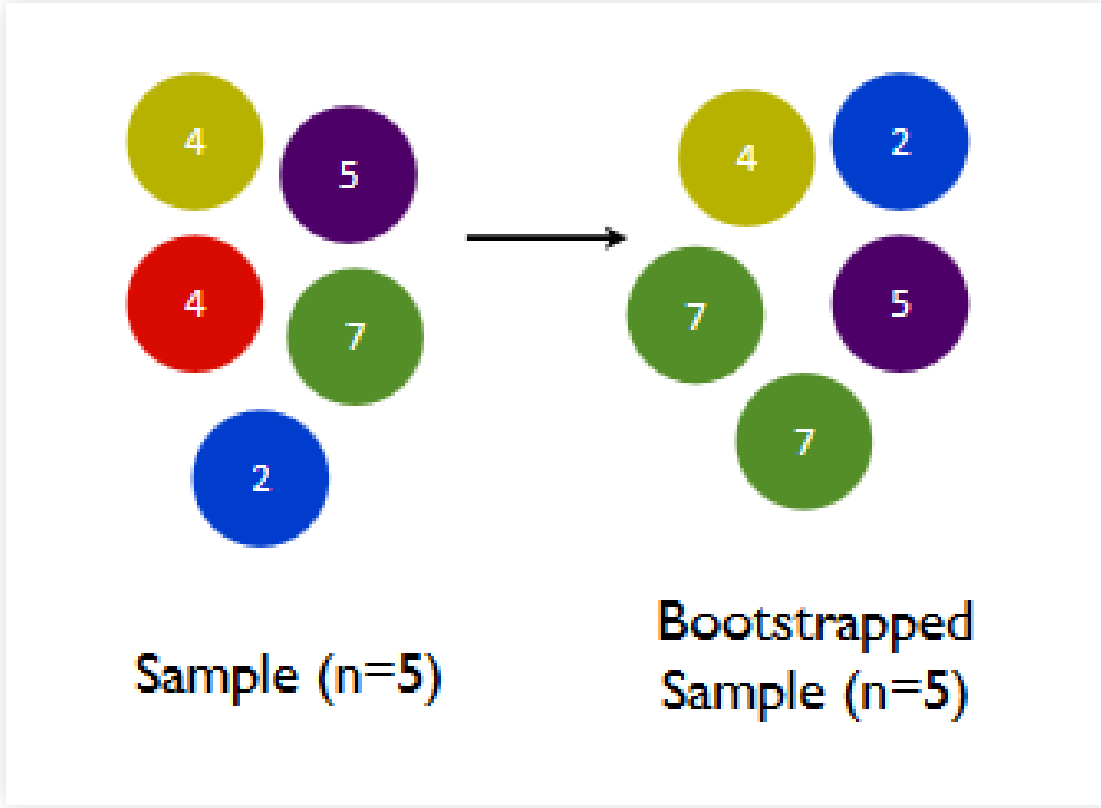
Problem: we can't take repeated samples of size N from the population, to see how our estimate changes across samples.

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*.

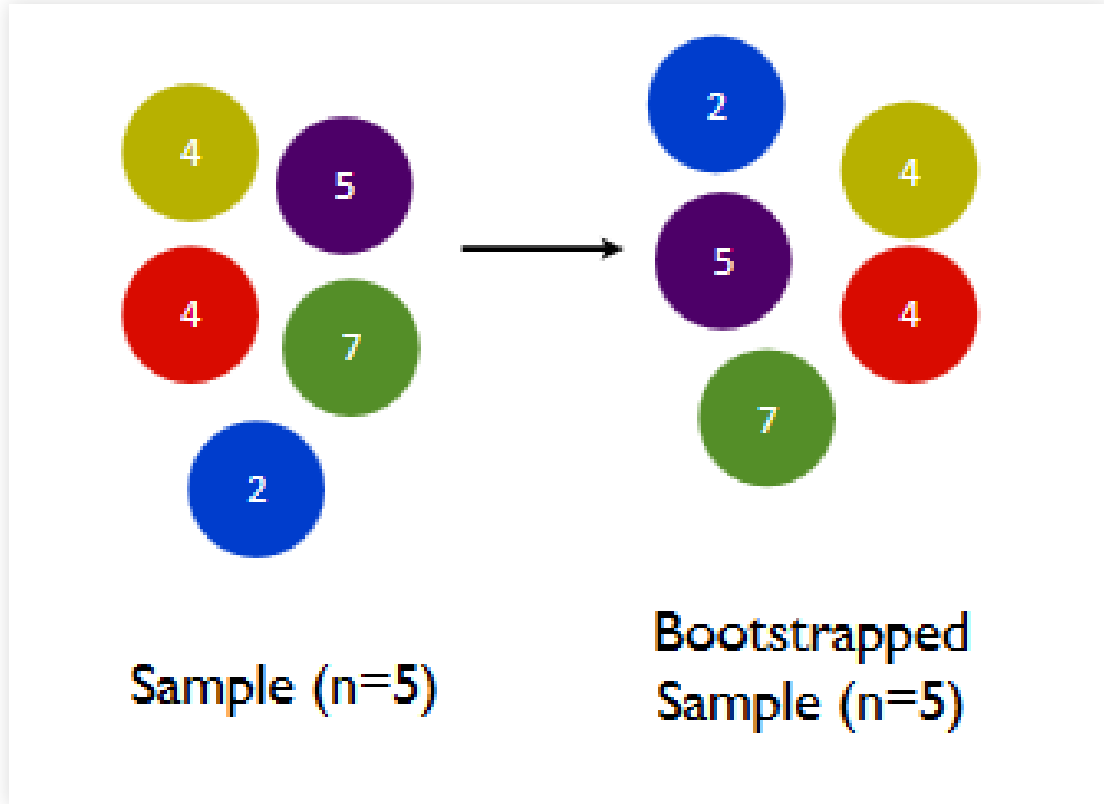
Sampling with replacement is key!

Bootstrapped sample 1



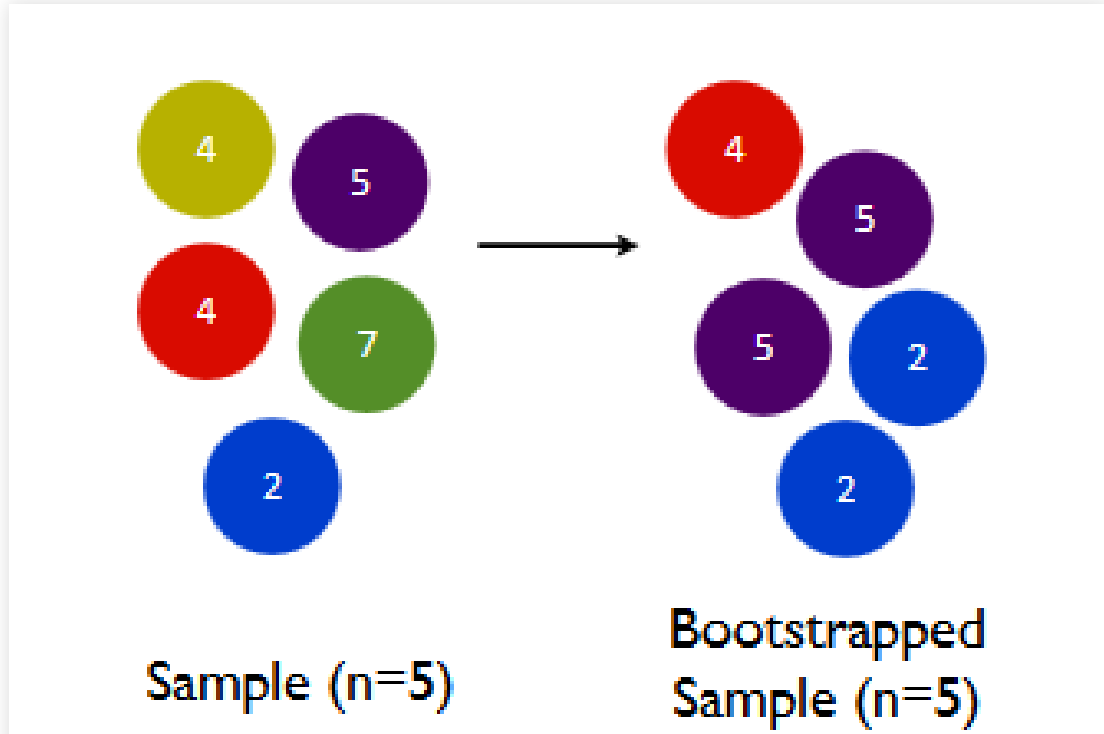
Sampling with replacement is key!

Bootstrapped sample 2

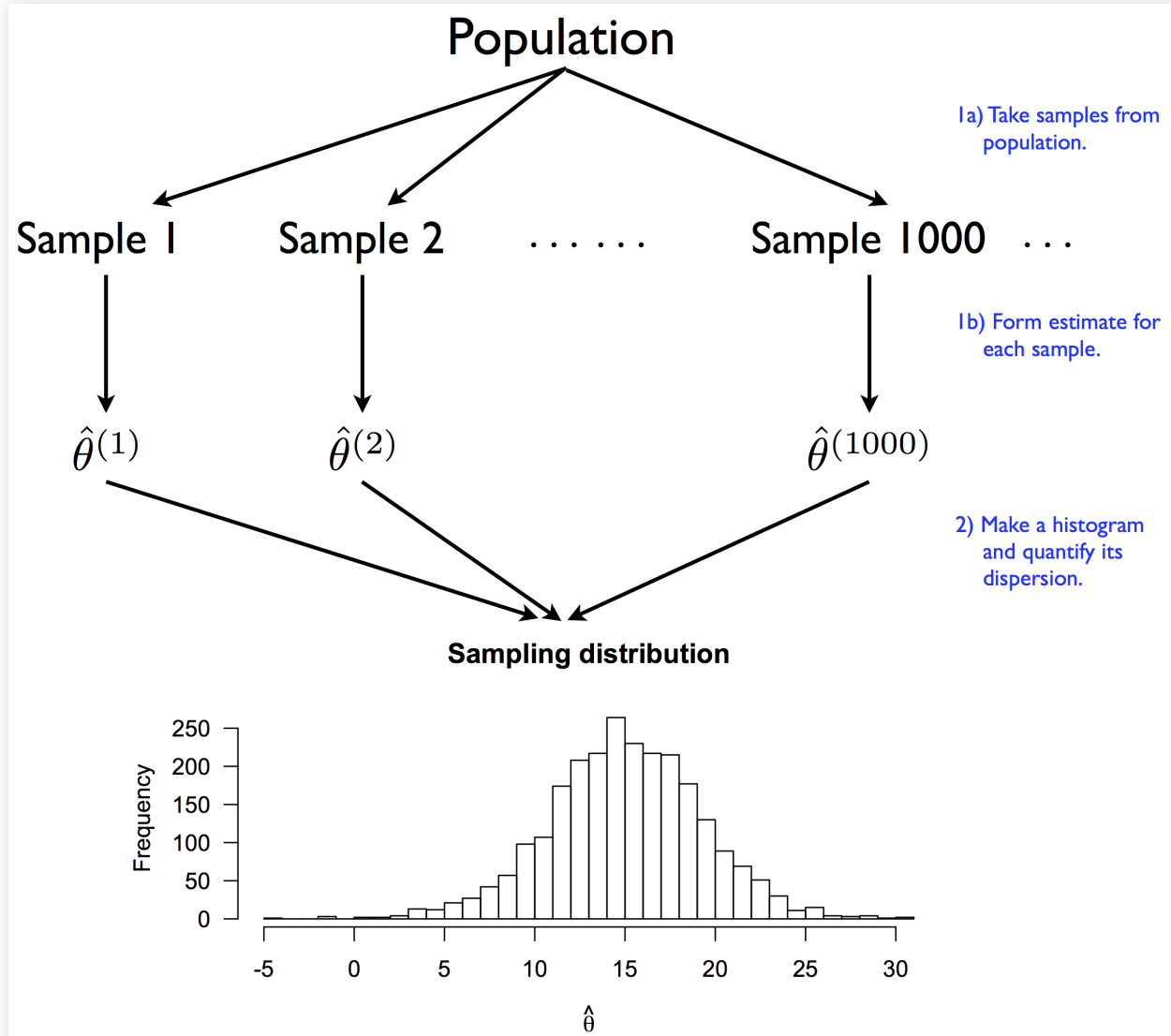


Sampling with replacement is key!

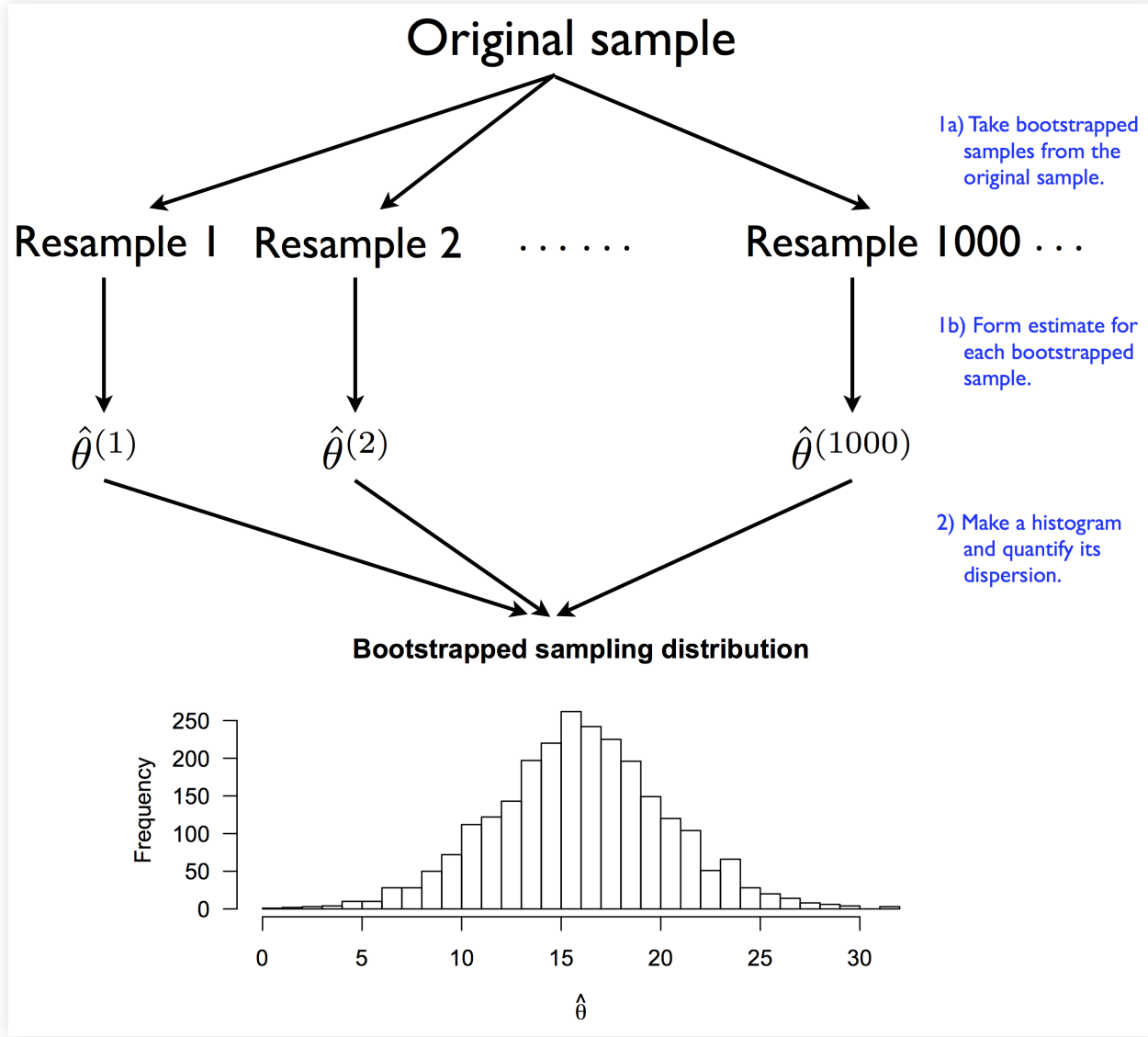
Bootstrapped sample 3



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!

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

Confidence intervals

Informally, an interval estimate is a range of plausible values for the parameter of interest. For example:

- Go out some multiple k of the bootstrapped standard error from your estimate:

$$\theta \in \hat{\theta}_N \pm k \cdot \hat{se}(\hat{\theta}_N)$$

- Use the quantiles (e.g. the 2.5 and 97.5 percentiles) of the bootstrapped sampling distribution, to cover a large fraction (e.g. 95%) of the bootstrapped estimates:

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

Confidence intervals

We'd like to be able to associate a *confidence level* with an interval estimate like this. How?

If an interval estimate satisfies the *frequentist coverage principle*, we call it a confidence interval:

> Frequentist coverage principle: If you were to analyze one data set after another for the rest of your life, and you were to quote X% confidence intervals for every estimate you made, those intervals should cover their corresponding true values at least X% of the time. Here X can be any number between 0 and 100.

Confidence intervals

An interval estimate takes the form $\hat{I}_N = [\hat{L}_N, \hat{U}_N]$. Just like a point estimate $\hat{\theta}_N$, the interval estimate is a random variable, because its endpoints are functions of a random sample.

We say that $[\hat{L}_N, \hat{U}_N]$ is a confidence interval at **coverage level** $1 - \alpha$ if, for every θ,

$$P_\theta \left(\theta \in [\hat{L}_N, \hat{U}_N] \right) \geq 1 - \alpha \,,$$

where P_θ is the probability distribution of the data, assuming that the true parameter is equal to θ.

Confidence intervals

The key statement here can be one of the most confusing in all of statistics:

$$P_\theta \left(\theta \in [\hat{L}_N, \hat{U}_N] \right) \geq 1 - \alpha \,,$$

Three questions to ask yourself:

- What is fixed?
- What is random?
- What is the source of this randomness?

Bootstrapped confidence intervals

So recall our two methods of generating an interval estimate using the bootstrap:

- The standard error method: $\theta \in \hat{\theta}_N \pm z^\star \cdot \hat{se}(\hat{\theta}_N)$, where $z^\star$ is a pre-specified quantile of the normal distribution.
- The quantile method (e.g. the 2.5 and 97.5 percentiles of the bootstrapped sampling distribution)

The obvious question is: do these interval estimates satisfy the frequentist coverage principle?

Bootstrapped confidence intervals

The answer is: not always, but often!

In lots of common situations, both forms of bootstrapped interval estimate *approximately* satisfy the coverage requirement:

$$P_\theta\left(\theta \in [\hat{L}_N, \hat{U}_N]\right) \approx 1 - \alpha\,,$$

And the approximation gets better with larger sample sizes. That is, as N gets large,

$$P_\theta\left(\theta \in [\hat{L}_N, \hat{U}_N]\right) \to 1 - \alpha$$

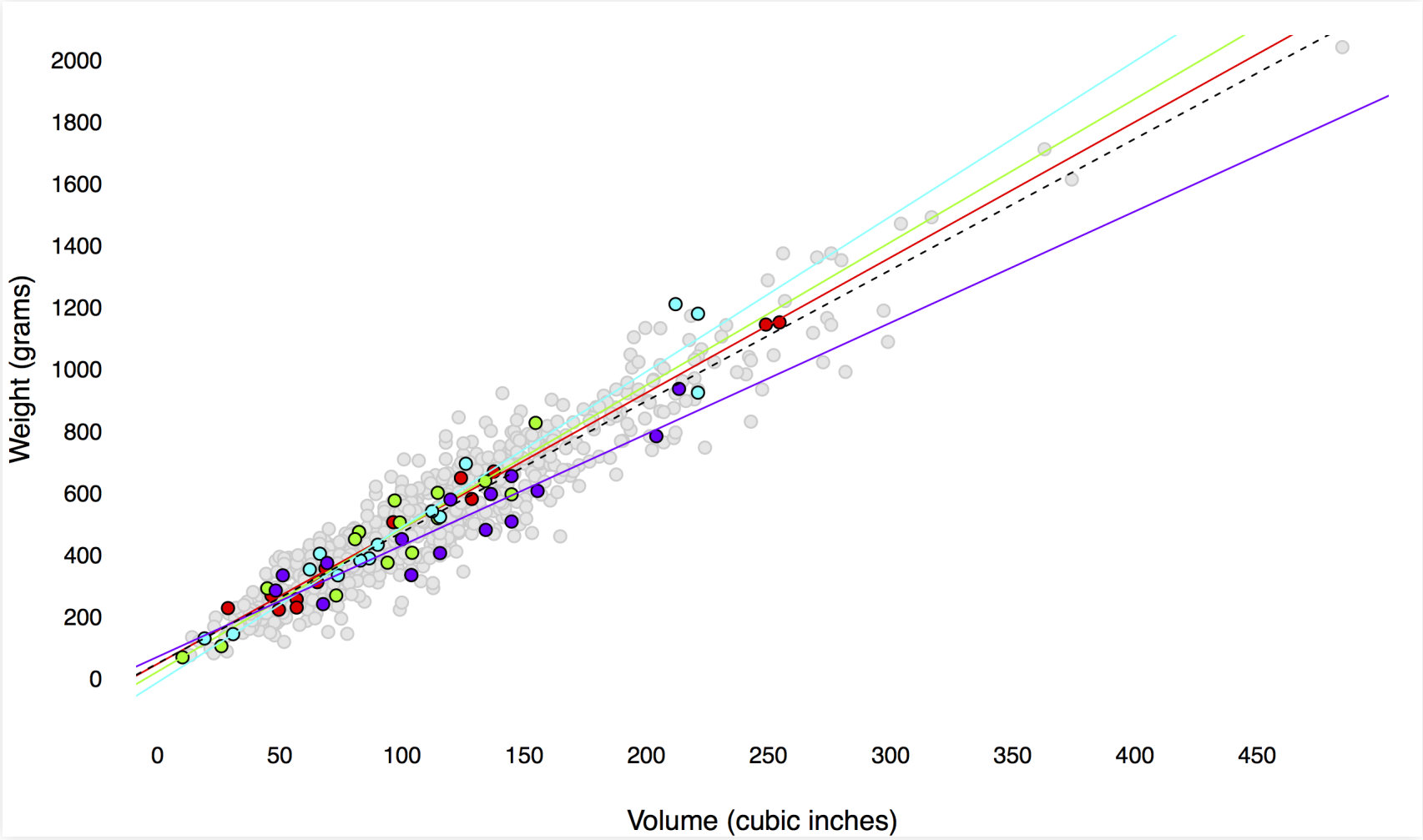
Bootstrapped confidence intervals

The math here is super hairy; we won't go into it. (Google "empirical process theory" if want to learn and you've got a year or two to spare…)

But we can run a sanity check through Monte Carlo simulation!

Let's revisit our thought experiment about fishing…

Bootstrapped confidence intervals

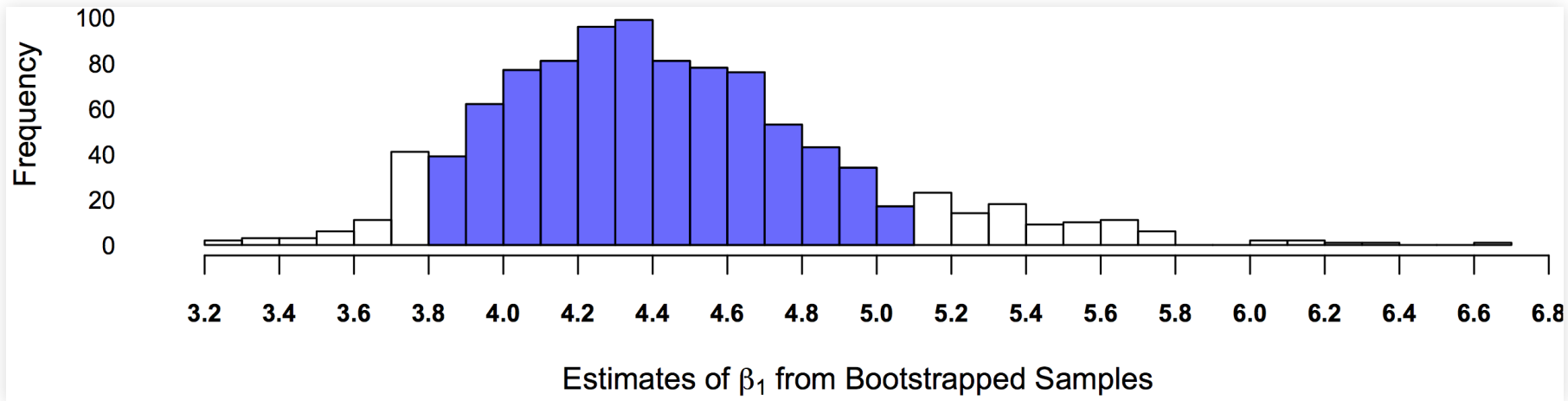


Bootstrapped confidence intervals

Let's go on a 100 fishing trips. On each trip:

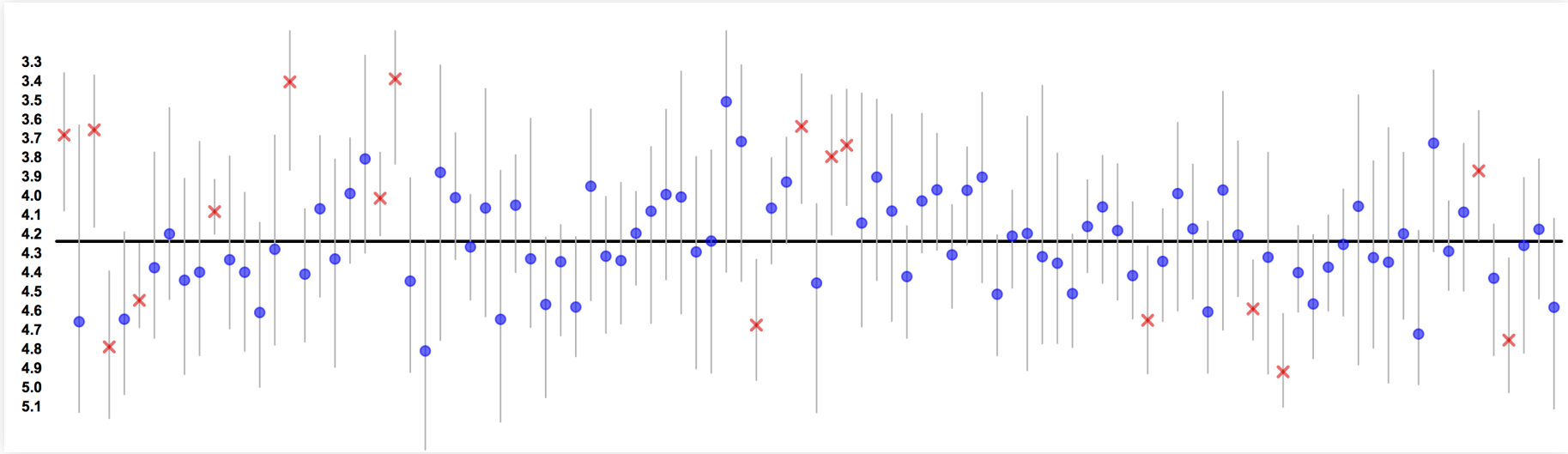
- we catch a sample of $N = 30$ fish
- we run OLS on our sample to estimate β_1: the slope of the weight-vs.-volume line
- we bootstrap our sample to get a 80% confidence interval for the slope.

Bootstrapped confidence intervals



Because we know the slope of the true line ($\beta_1 = 4.25$), we can check whether each bootstrapped confidence interval contains the true value. About 80% of them should!

Bootstrapped confidence intervals



- 100 different samples
- 100 different 80% confidence intervals
- 83 of them cover the truth—pretty good!

Plug-in standard errors

Sometimes we can use probability theory to calculate a “plug-in” estimate of an estimator's standard error. Some simple cases include:

- means and differences of means
- proportions and differences of proportions

Let's see an example and compare the result with a bootstrap estimate of the standard error.

Plug-in standard errors

Suppose that $X_1, X_2, \ldots, X_N$ are a sample of independent, identically distributed (IID) random variables with unknown mean μ and variance σ^2. Let $\bar{X}_N$ be the sample mean:

$$\bar{X}_N = \frac{1}{N} \sum_{i=1}^{N} X_i$$

Clearly $\bar{X}_N$ is a sensible estimate of μ, since it is unbiased: $E(\bar{X}_N) = \mu$ (show this!)

Plug-in standard errors

We can also calculate the theoretical variance of $\bar{X}_N$ as:

$$\begin{aligned}\text{var}(\bar{X}_N) = \text{var}\left(\frac{1}{N}\sum_{i=1}^{N} X_i\right) &= \frac{1}{N^2}\text{var}\left(\sum_{i=1}^{N} X_i\right) \\ &= \frac{1}{N^2} N\sigma^2 \\ &= \frac{\sigma^2}{N}\end{aligned}$$

Plug-in standard errors

This tells us that the *true standard error* of the sample mean is:

$$\text{se}(\bar{X}_N) = \frac{\sigma}{\sqrt{N}}$$

Or in words:

$$\text{Average error of the sample mean} = \frac{\text{Average error of a single measurement}}{\text{Square root of sample size}}$$

This is sometimes called de Moivre's equation, after Abraham de Moivre.

Plug-in standard errors

There's only one problem with de Moivre's equation: we don't know the true σ!

$$\text{se}(\bar{X}_N) = \frac{\sigma}{\sqrt{N}}$$

The obvious solution is to estimate σ from the data. This results in the so-called "plug-in" estimate of the standard error:

$$\hat{\text{se}}(\bar{X}_N) = \frac{\hat{\sigma}}{\sqrt{N}}$$

where $\hat{\sigma}$ is an estimate of the population standard deviation (e.g. the sample standard deviation).

Plug-in standard errors

Suppose we have an estimator $\hat{\theta}_N$ and we want to know its standard error. The general plug-in procedure involves three steps:

1. Use probability theory to derive an expression for the *true standard error* $\mathrm{se}(\hat{\theta}_N)$.
2. Use the data to estimate any unknown population parameters ϕ that appear in the expression for $\mathrm{se}(\hat{\theta}_N)$. (Note: this might even include θ, the parameter of interest itself.)
3. Plug in the estimate $\hat{\phi}$ into this expression to yield the plug-in standard error, $\hat{se}(\hat{\theta}_N)$

Let's see an example in `predimed_plugin.R`.

Summary

- Any estimator $\hat{\theta}_N$ is a random variable.
- Its probability distribution is called the *sampling distribution*.
- The sampling distribution describes the results of a thought experiment: *what if* we took lots and lots of samples, each of size N, and tracked how much our estimate changed?

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.
- In simple cases we can also calculate a *plug-in* estimate of the standard error, using probability theory together with sample estimates of unknown parameters.

Summary

- From the bootstrapped sampling distribution, we can get an interval estimate for the parameter of interest (using the standard-error method or the quantile method).
- Both methods approximately satisfy the frequentist coverage principle: under repeated sampling, they contain the true value roughly the correct percentage of the time.
- I tend to use the quantile method because it's pretty intuitive!

The bootstrap in supervised learning

Let's turn back to supervised learning!

So we believe that $y_i = f(x_i) + \epsilon_i$, we've collected some data, and we have our estimate $\hat{f}(x)$… How can we quantify our uncertainty for this estimate?

Fundamental frequentist thought experiment: "How might $f(x)$ have been different in an alternate data universe?"

The bootstrap in supervised learning

But what does "alternate data universe" actually mean?

- a different sample (of size N) of (x_i, y_i) pairs from the same population?
- a different set of residuals?
- a different realization of the same underlying random process/phenomenon?

The bootstrap

There's a version the bootstrap for all three situations:

- a different sample (of size N) of (x_i, y_i) pairs from the same population? **The nonparametric bootstrap**: pretty much example what we've already seen.
- a different set of residuals? **The residual-resampling bootstrap**
- a different realization of the same underlying random process/phenomenon? **The parametric bootstrap**

Let's see all three versions of the bootstrap one by one.

The nonparametric bootstrap

If you've seen the bootstrap before, it was probably this one!

- 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.

The nonparametric bootstrap

This leads to the following algorithm.

For b = 1 to B:

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

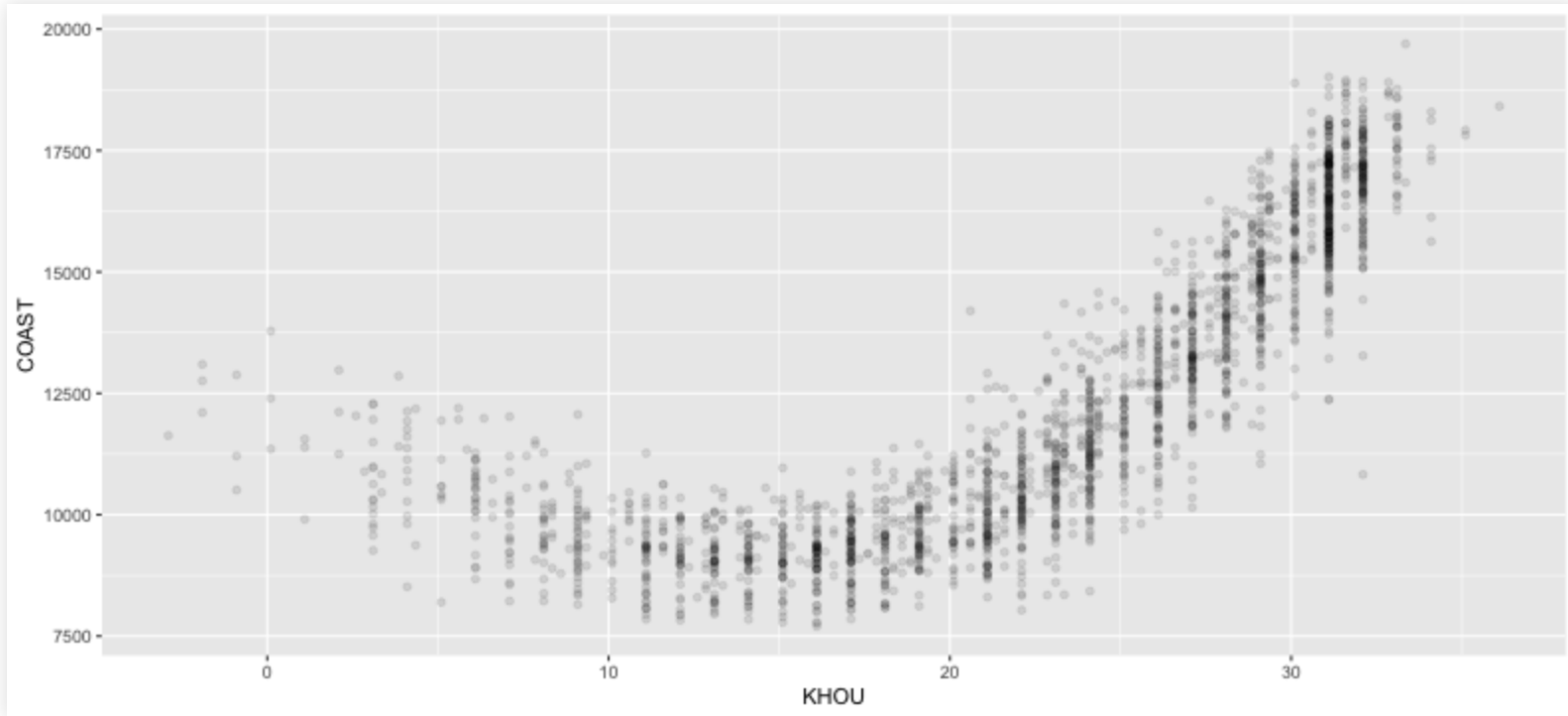
This gives us B draws from the bootstrapped sampling distribution of $\hat{f}(x)$.

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

An example

```
library(tidyverse)
loadhou = read.csv('../data/loadhou.csv')

ggplot(loadhou) + geom_point(aes(x=KHOU, y=COAST), alpha=0.1) +
  theme_set(theme_bw(base_size=18))
```



An example

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
```

An example

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
```

An example

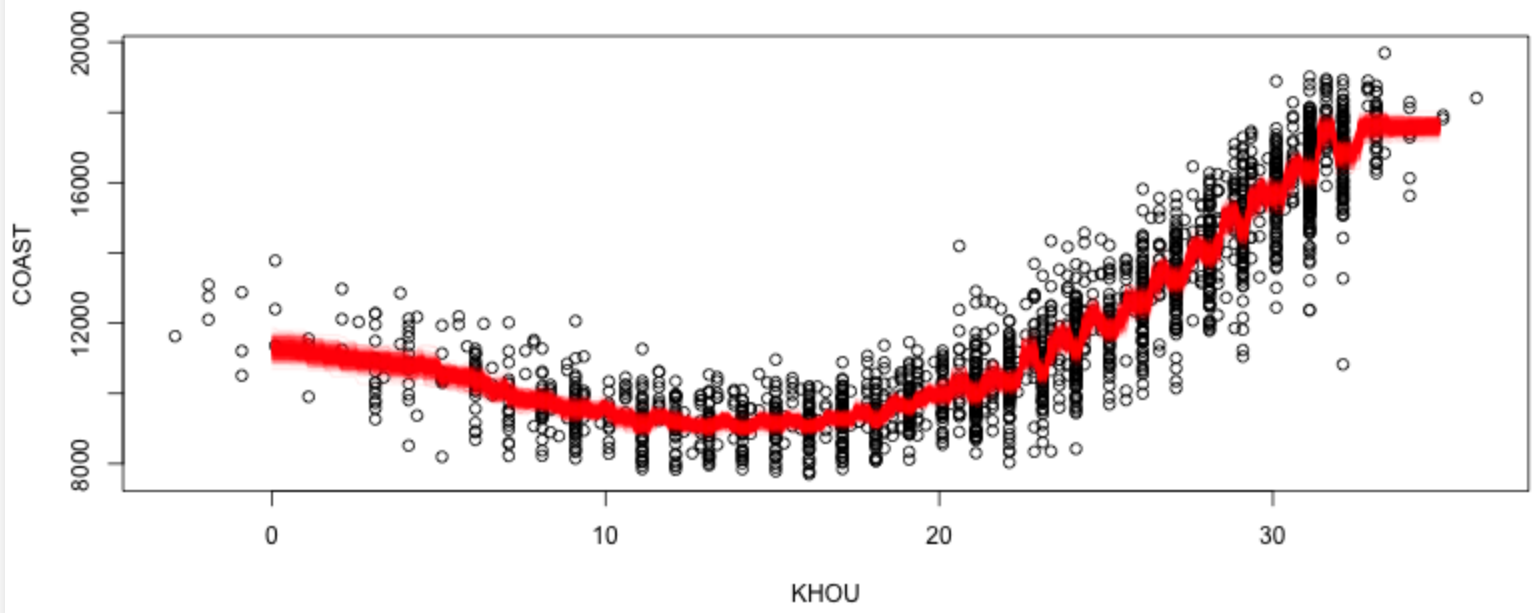
- Shortcut:

```
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
```

Spaghetti plot: using base R graphics

```
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))
}
```



The residual-resampling bootstrap

- Question: "how might my estimate $\hat{f}(x)$ have been different if the error terms/residuals had been different?"
- Assumption: each residual e_i is a random sample from a probability distribution $P(e)$ describing the noise in your data set.
- Problem: We don't know $P(e)$.
- Solution: Approximate $P(e)$ by $\hat{P}(e)$, the empirical distribution of the residuals from your fitted model.
- Key fact for implementation: sampling from $\hat{P}(e)$ is equivalent to sampling with replacement from residuals of your fitted model.

The residual-resampling bootstrap

This leads to the following algorithm. First fit the model, yielding

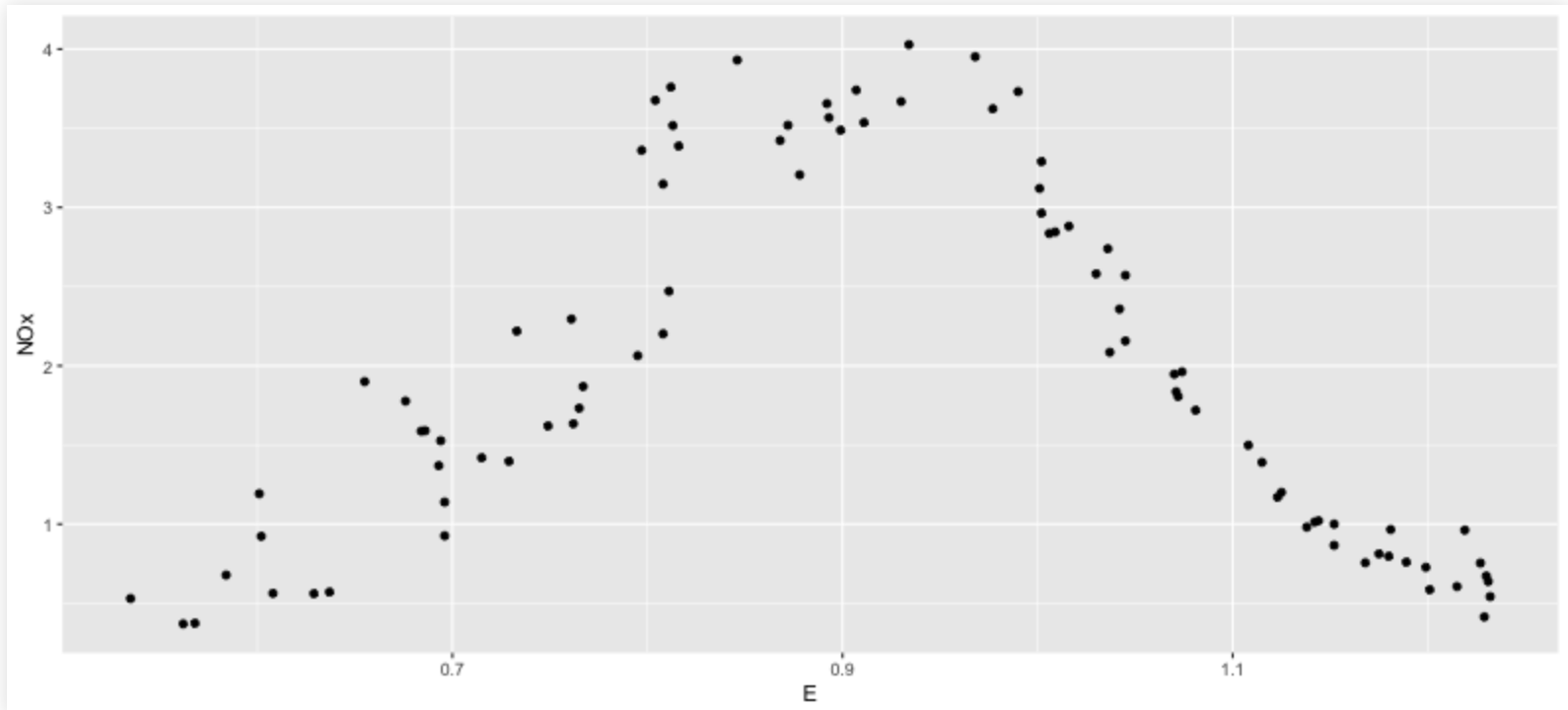
$$y_i = \hat{f}(x_i) + e_i$$

Then, for b = 1 to B:

- Construct a sample from $\hat{P}(e)$ by sampling N residuals $e_i^{(b)}$ *with replacement* from the original residuals $e_1, \ldots, e_N$.
- Construct synthetic outcomes $y_i^{(b)}$ by setting
$$y_i^{(b)} = \hat{f}(x_i) + e_i^{(b)}$$
- Refit the model to the $(x_i, y_i^{(b)})$ pairs, yielding $\hat{f}^{(b)}$.

An example

```
ethanol = read.csv('ethanol.csv')
ggplot(ethanol) + geom_point(aes(x=E, y=NOx)) +
  theme_set(theme_bw(base_size=18))
```



Key fact: the (x_i, y_i) are not random samples here! *The x_i points are fixed as part of the experimental design.*

An example

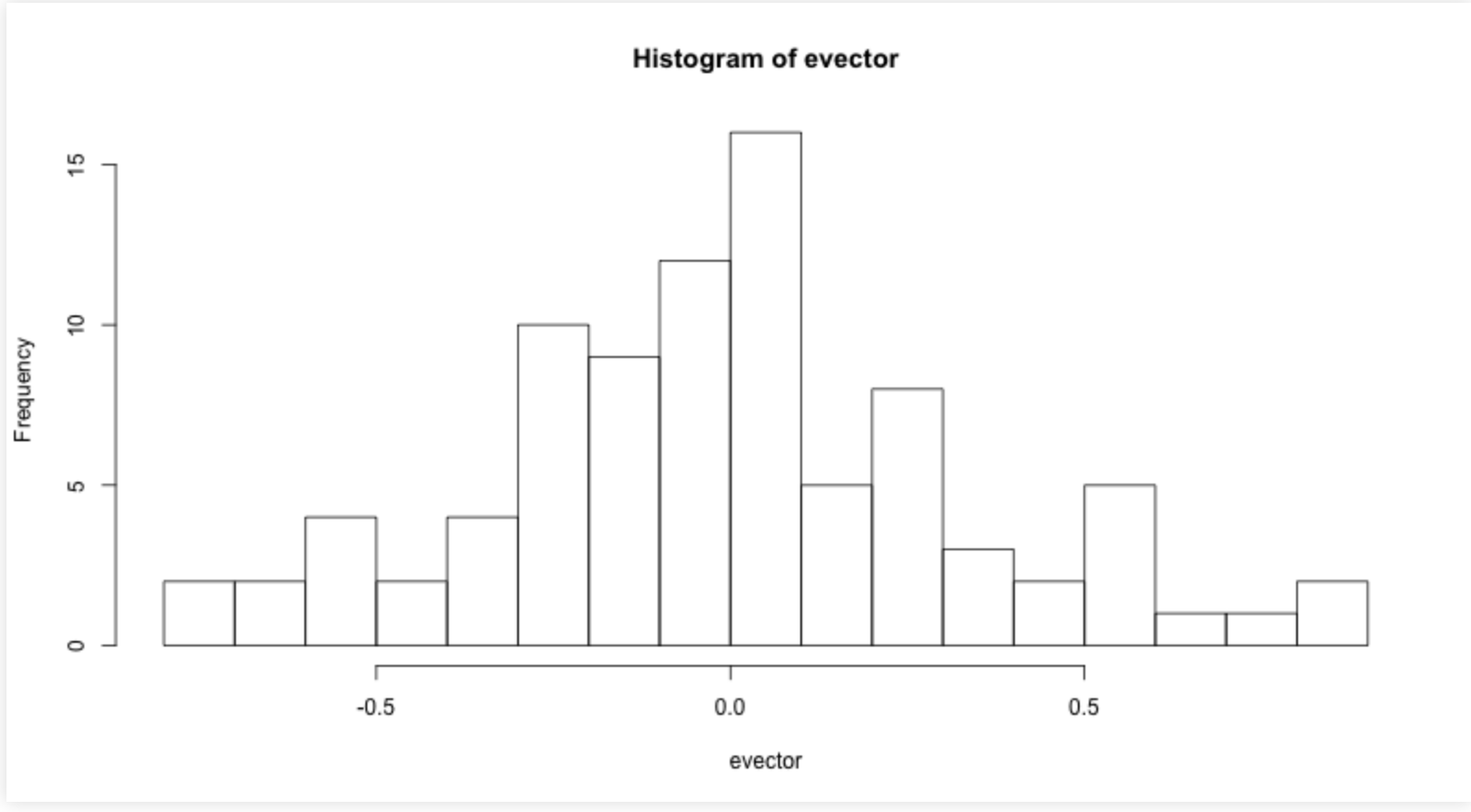
Let's quantify uncertainty for $f(0.7)$ and $f(0.95)$, under 5th-order polynomial model, via the residual resampling bootstrap.

First, let's look at the empirical distribution of the residuals:

```
poly5 = lm(NOx ~ poly(E, 5), data=ethanol)
yhat = fitted(poly5)
evector = ethanol$NOx - yhat  # empirical distribution of residuals
```

An example

```
hist(evector, 20)
```



This is our estimate $\hat{P}(e)$ for the probability distribution of the residuals.

An example

Now we bootstrap:

```
X_test = data.frame(E=c(0.7, 0.95))
boot5 = do(500)*{
  e_boot = resample(evector)
  y_boot = yhat + e_boot  # construct synthetic outcomes
  ethanol_boot = ethanol
  ethanol_boot$NOx = y_boot  # substitute real outcomes with synthetic ones
  poly5_boot = lm(NOx ~ poly(E, 5), data=ethanol_boot)
  fhat_boot = predict(poly5_boot, X_test)
  fhat_boot
}
head(boot5, 3)  # first column is f(0.7), second is f(0.95)
```

```
        X1       X2
1 1.479639 3.634581
2 1.576081 3.536532
3 1.506203 3.615962
```

An example

As before, we can get standard errors and/or confidence intervals.

- Standard errors:

```
se_hat = apply(boot5, 2, sd)
se_hat
```

```
        X1         X2
0.07574009 0.07527173
```

- Confidence intervals: calculate quantiles for each column

```
apply(boot5, 2, quantile, probs=c(0.025, 0.975))
```

```
             X1       X2
2.5%  1.359714 3.390013
97.5% 1.653690 3.688565
```

The parametric bootstrap

- Question: "how might my estimate $\hat{f}(x)$ have been different if my outcomes were a different realization from the same underlying conditional distribution $P(y_i \mid x_i)$?
- Assumption: each outcome y_i is a random sample from a conditional probability distribution $P(y \mid x)$.
- Problem: We don't know $P(y \mid x)$.
- Solution: Approximate $P(y \mid x)$ by $\hat{P}(y \mid x)$, the family of conditional distributions estimated from your sample.
- Key fact for implementation: $\hat{P}(y \mid x)$ is a parametric probability model parametrized by $\hat{f}(x)$, the fitted function.

The parametric bootstrap

Solution: simulate hypothetical “alternative data universes" from a parametric model fitted to your data.

Example: see `predimed_bootstrap.R`

Bootstrapping for risk estimation

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.** (See portfolio.R)

Bootstrapping for risk estimation

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:

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

Bootstrapping for risk estimation

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:

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

Bootstrapping for risk estimation

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:

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

Bootstrapping for risk estimation

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 $t = T$:

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

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

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

Bootstrapping for risk estimation

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.

Bootstrapping for risk estimation

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?