

# The Bootstrap Method

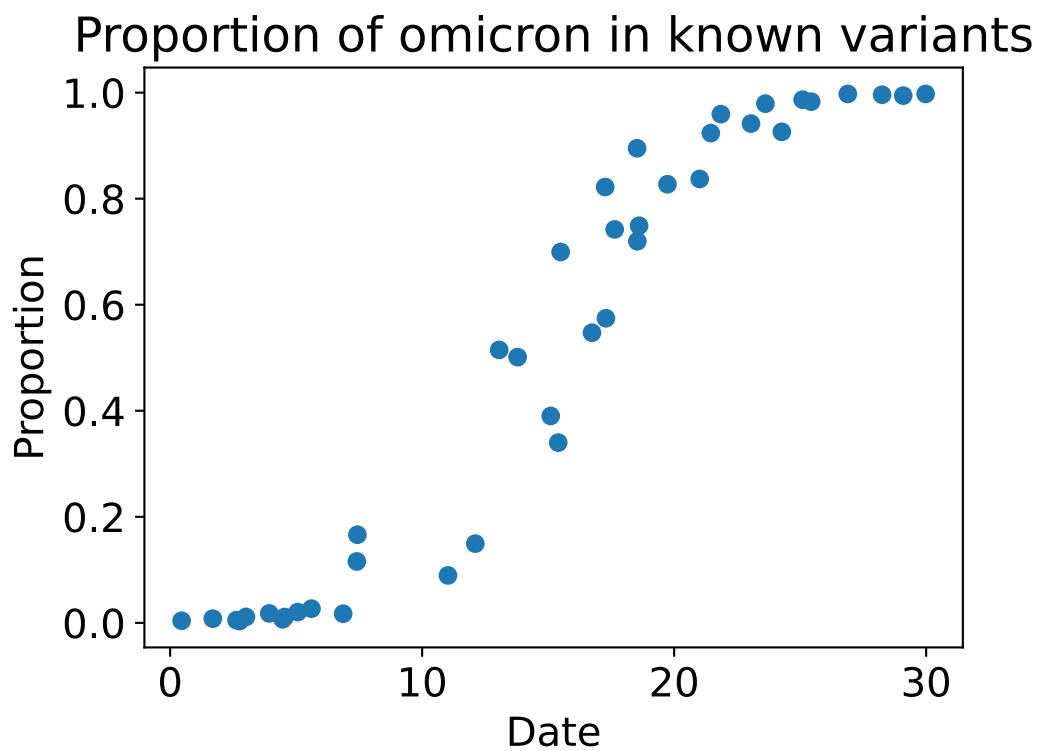
Dr. Devan Becker

Wilfrid Laurier University

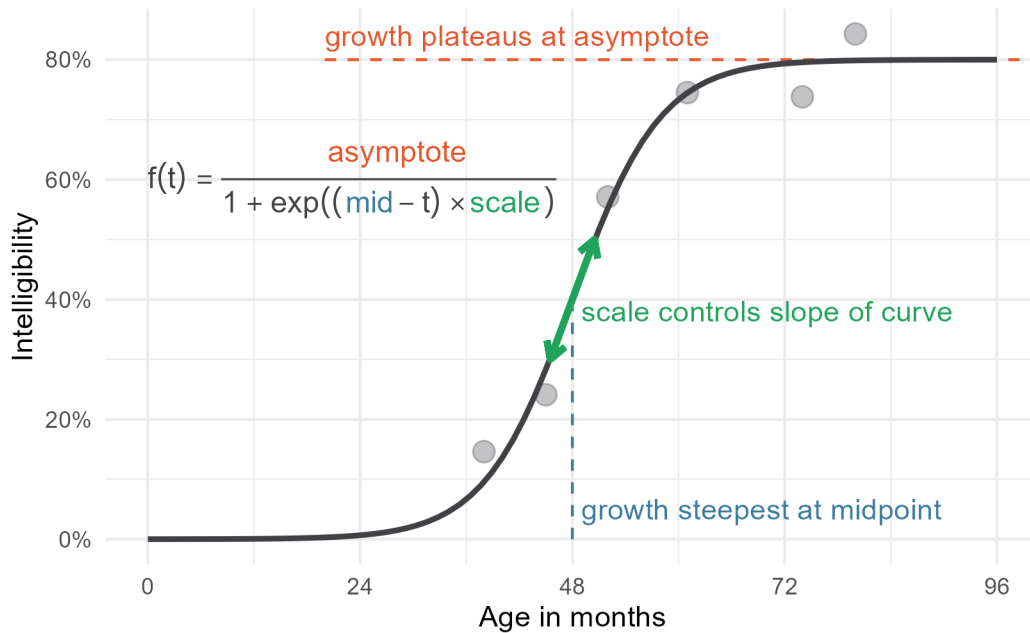
March 24, 2022

## Invasion of Omicron

How quickly did omicron become dominant?



## Logistic Growth Curve



Source: <https://www.tjmahr.com/anatomy-of-a-logistic-growth-curve/>

In this context, a logistic growth model is just the assumed shape of the curve. Just like how the normal distribution is completely characterized by its mean and standard deviation, the growth curve is characterized by the following three values:

- The asymptote, which is the maximum value that the curve will reach. In our case, we know that this is 1.
- The midpoint, which is the point at which the curve reaches half of the asymptote.
- The scale, which can be seen as the slope of the curve at the midpoint.

We don't need to go into too much detail for the model, other than to say that it is our assumption about the shape of the relationship and we haven't made any distributional assumptions.

## Fitting the Curve

Least squares, but no assumption of normality!

$$(\widehat{scale}, \widehat{mid}) = \hat{\theta} = \operatorname{argmin}_{\theta} (y_i - f(x_i|\theta))^2$$

```

. . .
1  import numpy as np
2  import pandas as pd
3  from scipy.optimize import curve_fit
4  omicron = pd.read_csv("data.csv")
5
6  def logistic_growth(x, r, k): # assume asymptote is 1
7      y = 1/(1 + np.exp((k - x) * r))
8      return y
9
10 theta_hat, covs = curve_fit(f = logistic_growth,
11                             xdata = omicron.date,
12                             ydata = omicron.prop,
13                             p0 = [0.3, 15])

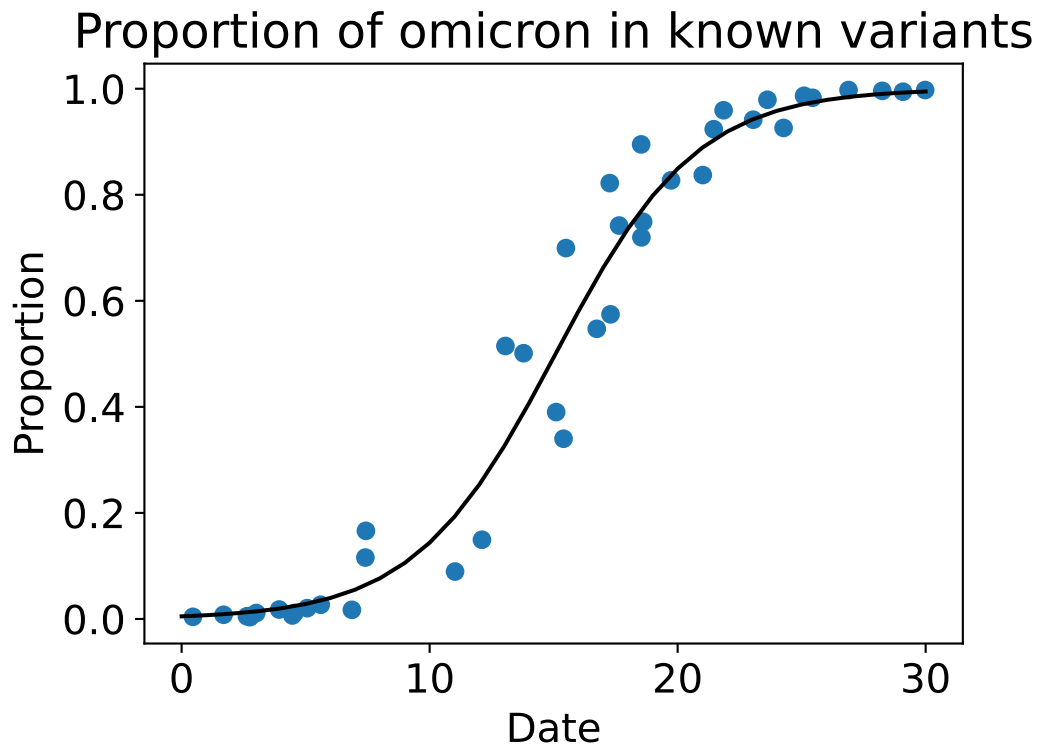
```

Since we're not making any distributional assumptions, we cannot use maximum likelihood. We can still find the least squares between the line and the points, but we don't have the convenient calculus behind linear models. Instead, it's an optimization routine. Note that *asym* is known to be one, so we only need to consider the slope and the scale.

Note that we can't even say that the observations are just random error around the line - this assumes additivity, which is not one of our assumptions! There's a theme here: we've assumed the shape of the relationship and absolutely nothing else.

## Challenge: Variance?

Point estimates are easy!



There are no distribution assumptions! How can we find an estimate for the variance around this curve?

Let's build up an intuition.

## Recap

What is the standard error?

1. The sd of the distribution of sample means
2. The sd of the observed sample mean
3. The sd of a sample

...

**i** Answer

Option 1 is the (simplified) definition.

The concept we're going to cover today relates to sampling distributions. So we'll start with a simple question: what is the standard error?

Since you're reading the pdf version, it's hard to stop you from reading the answer without thinking about it. I still recommend thinking about it for a minute or two.

Okay, so the answer is 1: the standard deviation of the distribution of sample means. By knowing the behaviour of sample means, we can say something about the population mean. That is, we can do *inference*.

## Recap

Which is a reasonable estimate of the population mean?

1. The sample mean
2. The sample median
3. Halfway between  $Q_1$  and  $Q_3$
4. All have their merits and pitfalls

...

**i** Answer

There's more than one way to estimate the centre!

Clearly, the correct answer is "All have their merits". The point of this question is to prime you for the next:

## Question of the Day

Which is a reasonable estimate of the population?

1. The sample mean
2. The sample mean and sample sd
3. The sample

...

**i** Answer

If our sample is representative, then it can be used as an estimate!

Today's lesson focuses on this idea: the sample can be seen as an estimate of the population, and can be used to construct a sampling distribution!

## Bootstrapping: Learning Outcomes

If the sample is a **plug-in estimate** of the population...

...

then we can build the **sampling distribution** from nothing but the **sample**!

The name “bootstrap” comes from the idea of “pulling yourself up by your bootstraps”, which refers to the act of pulling yourself out of poverty using only what you have and no other supports. The phrase was initially used sarcastically to highlight how patently ridiculous the idea was, but some people use it to simply mean “succeed using only what you have.”

The main purpose of bootstrapping is using the sample in place of the population. This allows us to estimate properties of the sampling distribution without making parametric assumptions about the population. It's actually kind of magical.

## Bootstrapping in Action

Re-fit with random indices (keeping  $(x, y)$  pairs together).

```
1  n_boot = 5000
2  scale_tracker = np.zeros(n_boot)
3  midpoint_tracker = np.zeros(n_boot)
4
5  for i in range(n_boot):
6      np.random.seed(i)
7
8      indices = np.random.randint(
9          low = 0,
10         high = omicron.shape[0],
11         size = omicron.shape[0]) # WITH REPLACEMENT
12     omi = omicron.iloc[indices]
13     theta, covs = curve_fit(logistic_growth,
14                             omi.date, omi.prop, p0 = [0.3, 15])
15     scale_tracker[i] = theta[0]
16     midpoint_tracker[i] = theta[1]
```

A (non-parametric) bootstrap sample is just a simple random sample from the sample that you already have. In practice, this means sampling  $n$  integers from 1 to  $n$ , and then taking these indices from your dataset. Of course, this means that we're sampling with replacement - we will almost always have repeated values in our sample.

Just like with samples from the population, each time we do this we'll have a different mean value. Just like with samples from the population, the standard deviation of these means is an estimate for the standard error of the mean.

Aside: We are making some assumptions about the population, though. The bootstrap doesn't work when the population has infinite variance or when there are, say, outliers in the population that don't exist in our sample.

## Results

```
xseq = np.linspace(0, 30, 31)
yseq = logistic_growth(xseq, theta_hat[0], theta_hat[1])

figure, axis = plt.subplots(2, 2, dpi = 100)
figure.tight_layout(h_pad=2)

axis[0, 0].plot(omicron.date, omicron.prop, "o",
               xseq, yseq, "-k")
axis[0, 0].set_title("Data with all resampled curves")

for i in range(midpoint_tracker.shape[0]):
    yi = logistic_growth(xseq,
                        scale_tracker[i], midpoint_tracker[i])
    axis[0, 0].plot(xseq, yi, color = "grey", alpha = 0.01, lw = 0.1)

#kr = np.polyfit(scale_tracker, midpoint_tracker, deg = 1)
#kseq = np.array([np.min(midpoint_tracker), np.max(midpoint_tracker)])
#rseq = np.array([np.min(scale_tracker), np.max(scale_tracker)])
#axis[0, 1].plot(scale_tracker, midpoint_tracker, "o")
#axis[0, 1].set_title("Scatterplot of scale versus midpoint")
#axis[0, 1].plot(rseq, kr[1] + kr[0]*rseq, "-r")

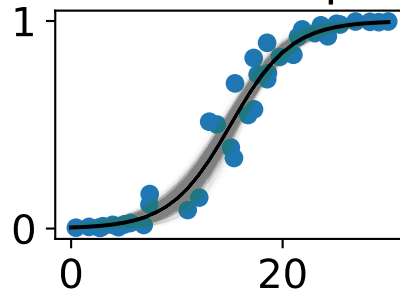
axis[0, 1].set_visible(False)

axis[1, 0].hist(scale_tracker, bins = 70)
axis[1, 0].set_title("Histogram of scale")
axis[1, 0].vlines(theta_hat[0], 0, 400, color = "red")

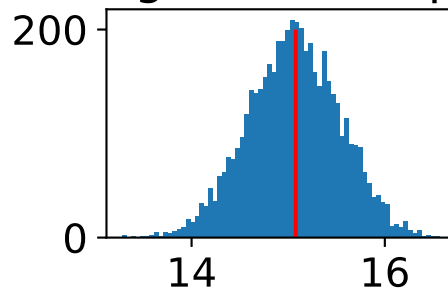
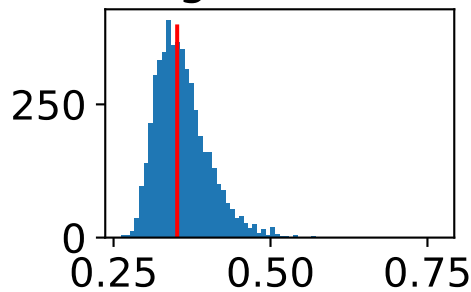
axis[1, 1].hist(midpoint_tracker, bins = 70)
axis[1, 1].set_title("Histogram of midpoint")
axis[1, 1].vlines(theta_hat[1], 0, 200, color = "red")
plt.show()
```



## Data with all resampled curves



## Histogram of scale Histogram of midpoint



Here are the results of the bootstrap sampling.

The first plot shows all of curves that were estimated by resamples as thin grey lines. Since there are 5,000 of them, they appear as one single line. Notice how the line fit to the original data isn't straight through the middle of this cloud of lines. The two outliers around  $x = 8$  seem to have a large effect on our parameter estimates! Note that we don't have a rigorous definition of influence in this model, but the bootstrap has revealed this to us!

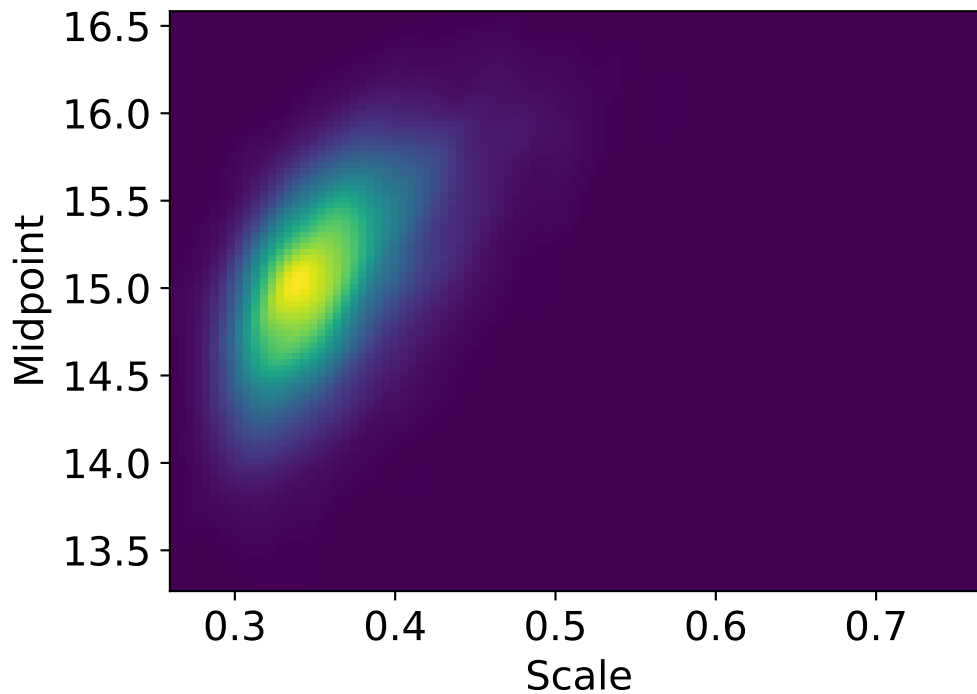
The third and fourth plots show the estimated sampling distributions along with a red line indicating the estimate from the original sample. The scale clearly has a non-normal, skewed distribution for this sample, whereas the midpoint does seem somewhat normal looking.

## Covariance between parameters

```
from scipy.stats import kde
k = kde.gaussian_kde(np.stack([scale_tracker, midpoint_tracker]))
xi, yi = np.mgrid[scale_tracker.min():scale_tracker.max():100*1j, midpoint_tracker.min():midpoint_tracker.max():100*1j]
zi = k(np.vstack([xi.flatten(), yi.flatten()]))

plt.pcolormesh(xi, yi, zi.reshape(xi.shape))
plt.xlabel("Scale")
plt.ylabel("Midpoint")
plt.show()
```

/tmp/ipykernel\_46243/3596105871.py:2: DeprecationWarning: Please use `gaussian\_kde` from the  
k = kde.gaussian\_kde(np.stack([scale\_tracker, midpoint\_tracker]))



As we saw in the previous slide, there is covariance between the parameters. This KDE estimate shows that this is not a linear correlation: there is a slight curve to this relationship, and the most common bootstrap estimates are not in the center of the distribution.

## Calculating a CI

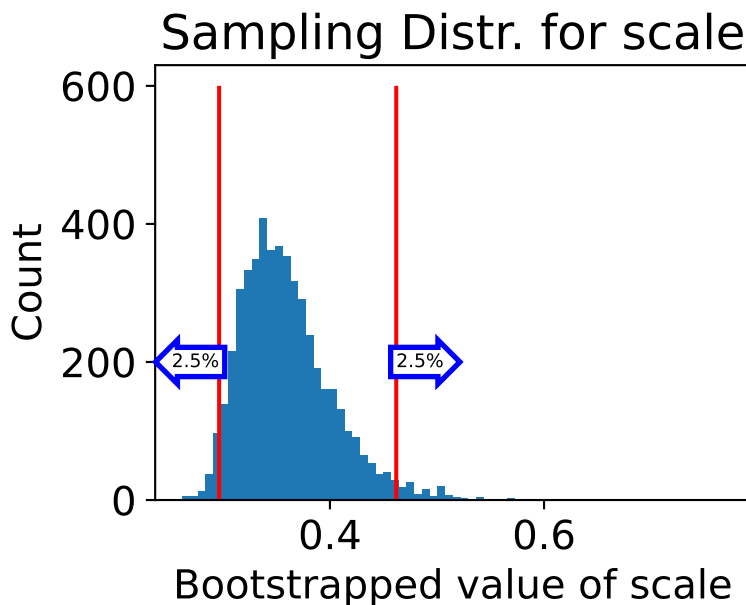
### 1. Quantiles

0.025 and 0.975

```
rtils = np.quantile(scale_tracker, [0.025, 0.975])

plt.figure(figsize = (4,3))

plt.hist(scale_tracker, bins = 70)
plt.vlines(rtils, 0, 600, colors = "red")
plt.title("Sampling Distr. for scale")
plt.xlabel("Bootstrapped value of scale")
plt.ylabel("Count")
plt.text(rtils[0], 200, "2.5%",
        ha = "right", va = "center", size = 7,
        bbox = dict(boxstyle="larrow,pad=0.3", fc = "white",
                    ec = "b", lw = 2))
plt.text(rtils[1], 200, "2.5%",
        ha = "left", va = "center", size = 7,
        bbox = dict(boxstyle="rarrow,pad=0.3", fc = "white",
                    ec = "b", lw = 2))
plt.show()
```



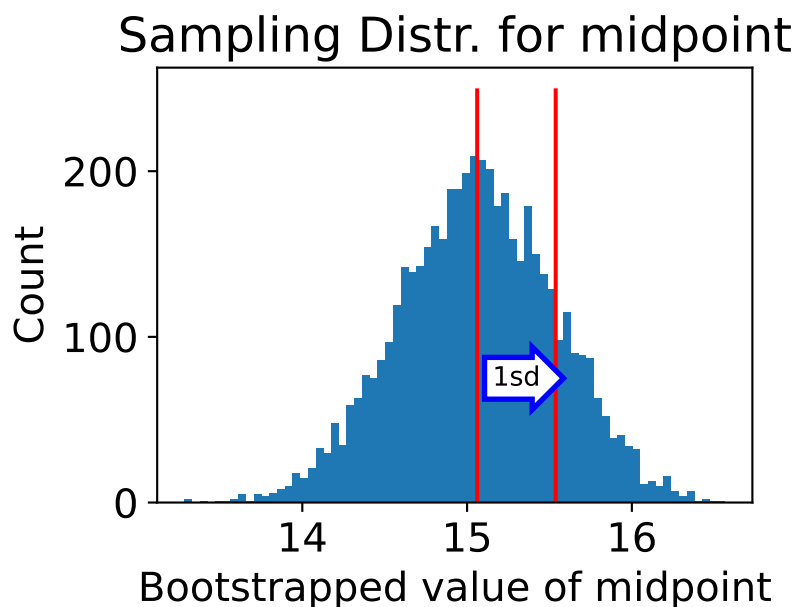
Estimates sampling distr.

2. Normal (or  $t$ ) CI

$$\hat{k} \pm t_{\alpha/2} s_{k^*}$$

```
k_mean = np.mean(midpoint_tracker)
k_sd = np.sqrt(np.var(midpoint_tracker))

plt.figure(figsize = (4,3))
plt.hist(midpoint_tracker, bins = 70)
plt.vlines([k_mean, k_mean + k_sd], ymin = 0, ymax = 250, color = "red")
plt.text(k_mean + k_sd/2, 75, "1sd",
         ha = "center", va = "center", size = 10,
         bbox = dict(boxstyle="arrow,pad=0.3", fc = "white",
                     ec = "b", lw = 2))
plt.title("Sampling Distr. for midpoint")
plt.xlabel("Bootstrapped value of midpoint")
plt.ylabel("Count")
plt.show()
```



Estimates standard error.

Now that we have a way of estimating the sampling distribution, what do we do with it? There are two main approaches.

First, we can just use the bootstrap sampling distribution as a plug-in estimate for the true sampling distribution. This means looking at all bootstrap samples and finding, say, the 2.5 and 97.5 percentiles.

Second, we can use the standard deviation from the sampling distributions as the standard error and calculate our usual confidence interval as the sample mean plus and minus the standard error. If the bootstrap sampling distribution looks skewed we can make appropriate adjustments.

There are other approaches out there, such as calculating the average width of parametric (normal) confidence intervals built from each re-sample or incorporating bias, but these can be explored in assignments or in your spare time.

## The Bootstrap Principle

**The sample is a plug-in estimate of the population.**

...

 The sample must be representative of the population.


Bootstrapping doesn't fix biased sampling or small sample sizes.

...

 Centered near the mean of the sample not the mean of the population.


Bootstrapping does not improve the point estimate.

...

 Correct assumptions about the population can be powerful.

Today's lesson has been on *non-parametric* bootstrapping.

...

 Samples must be independent!

No time series models!

For bootstrap, we're using the sample as a plug-in estimate of the population. This heavily relies on the assumption that we have a sample that is representative of the population.

Assuming we have this assumption, we don't need to make assumptions about the parametric form of the population!

It is important to note that the bootstrap doesn't improve on our current estimate. For the sample mean, the bootstrap distribution is centered on the sample mean, *NOT* the population mean. This is why it's useful for confidence intervals, but don't for a second think that it will get us more information about the population mean.

## Discussion

Is the bootstrap good for medians?

## Try it Yourself!

1. Sample from a population.
2. Calculate the statistic
3. Calculate the bootstrap version of that statistic

Try it for:

Quantiles

Median, 0.025, 0.975

The t-statistic

Bootstrap is robust to deviations from normal populations!

Mean of skewed distributions

What's your intuition? (This may be an assignment question...)

The bootstrap is not a magic bullet that works for everything. For the median, there is code in the repo to estimate the sampling distribution from a known (non-normal) population. You'll need to take a sample, then calculate a bootstrap confidence interval for that sample.

Alternatively, we can look at confidence intervals for the mean, but when the population is heavily skewed.

One of these will be part of a future assessment!

## Summary

- Bootstrapping is simply **re-sampling** from your sample (**Plug-in Principle**)
- Bootstrapping works well for **complicated or unknown/unassumed sampling distributions**
- Bootstrap confidence intervals can be great, but **don't bootstrap blindly!**

## Next Steps

- Bias-corrected and accelerated bootstraps
- Parametric and smoothed bootstrap
- Permutation tests

### Further Reading:

- Bootstrap for time series models

In the next lesson, we'll look at the bootstrap method for hypothesis testing, in particular how the bootstrap t-statistic can be *better* because it incorporates the covariance between the mean and the standard deviation. However, it is also too small by a factor of  $\sqrt{(n-1)/n}$ .

In the lesson after that, we'll look at two extensions of bootstrapping, including when we assume something about the shape of the population and a hybrid between no assumptions and parametric assumptions.