Quantifying Sampling Error

2017-02-27

Preparation

We will eventually use the data in the *mnSchools.csv* file. These data include institutional-level attributes for several Minnesota colleges and universities. The source of these data is: http://www.collegeresults.org. The attributes include:

- `id`: Institution ID number
- `name`: Institution name
- `gradRate`: Six-year graduation rate. This measure represents the proportion of first-time, full-time, bachelor's or equivalent degree-seeking students who started in Fall 2005 and graduated within 6 years.
- `public`: Dummy variable indicating educational sector (0 = private institution; 1 = public institution)
- `sat`: Estimated median SAT score for incoming freshmen at the institution
- `tuition`: Cost of attendance for full-time, first-time degree/certificate-seeking in-state undergraduate students living on campus for academic year 2013-14.

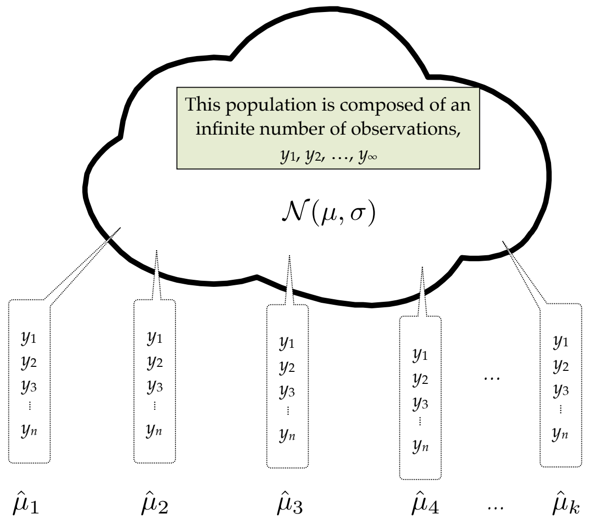
```
mn = read.csv(file = "~/Google Drive/Documents/github/EPsy-8252/data/mnSchools.csv")
head(mn)
```

id	name	gradRate	public	sat	tuition
1	Augsburg College	65.2	0	1030	39294
3	Bethany Lutheran College	52.6	0	1065	30480
4	Bethel University, Saint Paul, MN	73.3	0	1145	39400
5	Carleton College	92.6	0	1400	54265
6	College of Saint Benedict	81.1	0	1185	43198
7	Concordia College at Moorhead	69.4	0	1145	36590

Sampling Distributions

A *sampling distribution* is the probability distribution of a particular statistic based on drawing ALL possible random samples from a population and computing that statistic for each of them. When the population is infinitely large, the sampling distribution is a pure theoretical construct and can only be derived from theoretical results (e.g., mathematical proof) when they are known, or approximated using other methods (e.g., simulation). The goal for statisticians is to **quantify the variation** in the statistic that is due to random sampling—quantify the sampling error.

To understand this, let's work with a more concrete example. Let's assume we have FULL information about the population we are sampling from: Normally distributed with a mean of 0 and a standard deviation of 1. Let's also assume we want to draw samples of size 100 from this population, and for each sample we will collect the sample mean. Our goal is to describe/quantify the variation in the mean that is due to sampling error. Visually, this looks like the following:


This population is composed of an infinite number of observations, $y_1, y_2, \ldots, y_\infty$
$\mathcal{N}(\mu, \sigma)$
y_1 y_2 y_3 $\vdots$ y_n
$\hat{\mu}_1$ $\hat{\mu}_2$ $\hat{\mu}_3$ $\hat{\mu}_4$ ... $\hat{\mu}_k$

Statistical Theory

Because we have full information about the population and it is normally distributed, we can use statistical theory to help with achieving our goal. Theory says that if the population is normally distributed, that the sampling distribution of the sample mean will also be normally distributed. The mean of the sampling distribution of the sample mean (that's a mouthful, so statisticians use the term *expected value*) will be identical to the population mean ($E(\bar{x}) = \mu$). The standard deviation of the sampling distribution of the sample mean (the standard error) will be equal to the standard deviation of the population divided by the square root of the sample size ($\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$). Thus, using theory, for our example, the sampling distribution for the sample mean will be

- Normally distributed
- $E(\bar{x}) = 0$
- $\text{SE}_{\bar{x}} = \frac{1}{\sqrt{100}} = 0.1$

The goal was to quantify the sampling error, which we did when we computed the standard error. This, along with the expected value, tells us that, on average, the sample mean (computed from a random sample of size 100 drawn from our known population) will have a value of 0. However, we should not be surprised if the sample mean deviates from zero by as much as 0.2 (twice the SE).

Simulation

We can also approximate the shape, center, and standard error of the sampling distribution of the sample mean using simulation. In R, to draw from a given normal distribution we use the `rnorm()` function. Below, we draw 100 random observations from the normal distribution having a mean of 0 and a standard deviation of 1 and also compute its mean.

```
mean(rnorm(n = 100, mean = 0, sd = 1))
```

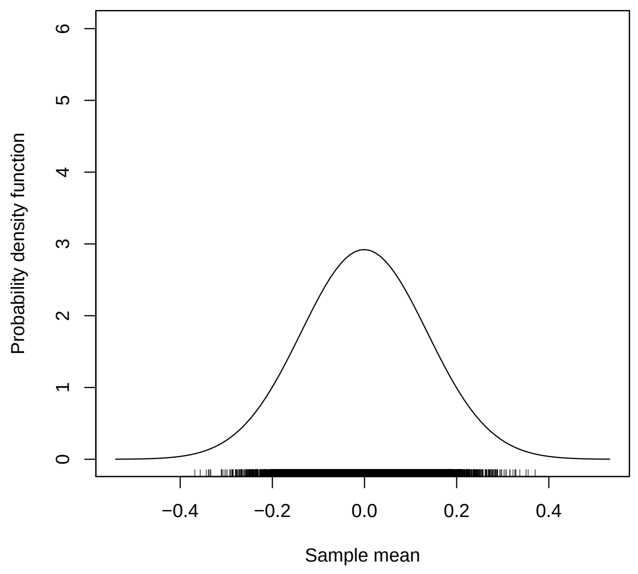
```
## [1] 0.065
```

If we do this many, many times, each time recording the sample mean, we can then examine the probability distribution of these means (to approximate the shape), compute the mean value of all these means (to approximate the expected value), and compute the standard deviation of all these means (to approximate the standard error). The `do()` function from the **mosaic** package will help us record and collect these means. Below we carry out this sampling process 10,000 times.

```
library(mosaic)

myMeans = do(10000) * mean(rnorm(n = 100, mean = 0, sd = 1))
#head(myMeans)

# Plot the distribution of sample means
library(sm)
sm.density(myMeans$mean, xlab = "Sample mean")
```



```
# Compute the mean of the sample means
mean(myMeans$mean)
```

```
## [1] -0.000541
```

```
# Compute the sd of the sample means
sd(myMeans$mean)
```

```
## [1] 0.0998
```

Remember that these are approximations of the values that we got using statistical theory. But, you can see that they are reasonable approximations of these values. The shape looks approximately normal, the mean of the simulated means is close to 0, and the approximation of the standard error is close to 0.1.

Partial Information

To simulate, or to use theory, we needed to have FULL information about the population, namely its shape, the mean, and the standard deviation. What happens when we don't have full information? Well, in those cases, the theory leads only to approximations as well (and only under certain conditions). For example, when the standrad deviation of the population is unknown, then we have to use the standard deviation of each random sample as an approximation of this value. In that case, the resulting sampling distribution of the sample mean is not normally distributed. The additional uncertainty added from having to use the sample standard deviation as an approximation to the population standard deviation makes the sampling distribution t-distributed. Or, when the population shape is not normally distributed, we have to rely on a result from theory called the *Central Limit Theorem* which says that if the sample size is large enough, then even though the population wasn't normally distributed, the sampling distribution of the sample mean will still be approximately normal.

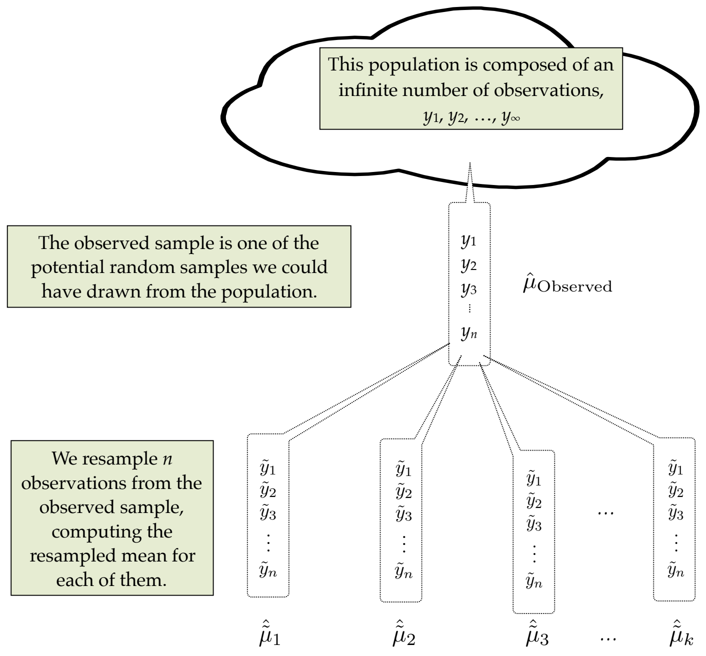
But, the known theory under partial information is quite limited.

- **It mostly requires a normal distribution of the population.** When the population is not normally distributed, the theory we have is much more limited and weaker. There is a reason that the asusmptions for most of the tests/CIs you have learned to this point have ana assumption that the population is normally distributed.
- **We only have theory for some statistics.** We know a lot about the sampling distribution of the sample mean, regression coefficients, and some others. There is little theory about the sampling distribution for other statistics. For example, there is no formula to compute the standard error from the sampling distribution of the sample median. Nor can we compute from a formula the SE for statistics such as R^2.

Bootstrapping: An Alternative Method to Approximate the SE

Bootstrapping is a simulation-based approach to approximating the SE. The advantages of bootstrapping are that we do not need full information about the population (in fact, it has even been used when there is essentially no information about the population). It is also a method that works to approximate the SE across almost all statistics you may want to find the SE for.

The basic premise of bootstrapping is that any one sample is a smaller reflection of the population. In that sense, we can substitute the sample for the population and use simulation to *resample* from this sample. This is called the "plug-in" principle. (We are plugging-in the sample for the population.) Visually we depict this concept below.


This population is composed of an infinite number of observations, $y_1, y_2, \ldots, y_\infty$
The observed sample is one of the potential random samples we could have drawn from the population.
y_1 y_2 y_3 ⋮ y_n
$\hat{\mu}_{\text{Observed}}$
We resample n observations from the observed sample, computing the resampled mean for each of them.
$\tilde{y}_1$ $\tilde{y}_2$ $\tilde{y}_3$ ⋮ $\tilde{y}_n$
$\hat{\hat{\mu}}_1$ $\hat{\hat{\mu}}_2$ $\hat{\hat{\mu}}_3$ … $\hat{\hat{\mu}}_k$

If we were to resample n observations out of a sample that itself has n observations, we would get the same n observations each time. To avoid this problem, when we resample from the observed sample, we resample n observations WITH REPLACEMENT. To illustrate this, let's sample 100 observations from the population we have been working with. We will call this our observed sample. (The `set.seed()` function allows us to replicate this by giving us the same observed sample.)

```
set.seed(567)
observed = rnorm(n = 100, mean = 0, sd = 1)
observed
```

```
##   [1]  0.6567  0.3392 -0.6408 -1.4401  0.2208 -0.4098 -0.3725 -2.1234
##   [9]  0.0599 -1.9398 -1.1382 -0.7564 -2.1480  1.3594 -0.2696  0.0888
##  [17] -1.9292  0.5701  1.4159 -0.5537 -0.4523 -0.4653  0.6126 -0.5122
##  [25]  0.7333  0.4441 -0.3589  0.9153  1.0341 -0.6560  0.0129  1.4749
##  [33]  0.0437  0.5619  0.3563 -1.0012  1.4028  0.4163 -1.1409  0.4602
##  [41]  0.9921  0.5015 -0.2341  0.0208 -0.9579  0.5868 -0.0370 -0.8947
##  [49] -0.2553 -0.5143  1.2909  0.0774 -1.7378 -0.3613  1.4134 -2.7795
##  [57] -1.8887 -0.9533 -1.1119 -0.3165  0.3335  0.5993  0.7462 -1.7402
##  [65] -0.5338 -0.2854  1.8822  0.9357 -0.0636 -0.8839  1.3923  0.0910
##  [73] -1.8015  0.3384 -0.7796 -0.8799  1.1825  0.1086  0.7399  1.0745
##  [81]  1.3038 -0.4666  0.3456 -1.1259  1.5963  0.6135 -0.2190  0.3745
##  [89]  1.8172 -0.7918  1.2540 -0.7949 -0.6786 -0.9056 -2.3957  0.7801
##  [97] -2.0927  0.8731  0.1396  0.2510
```

The `resample()` function from the **mosaic** package draws a bootstrap sample (samples with replacement).

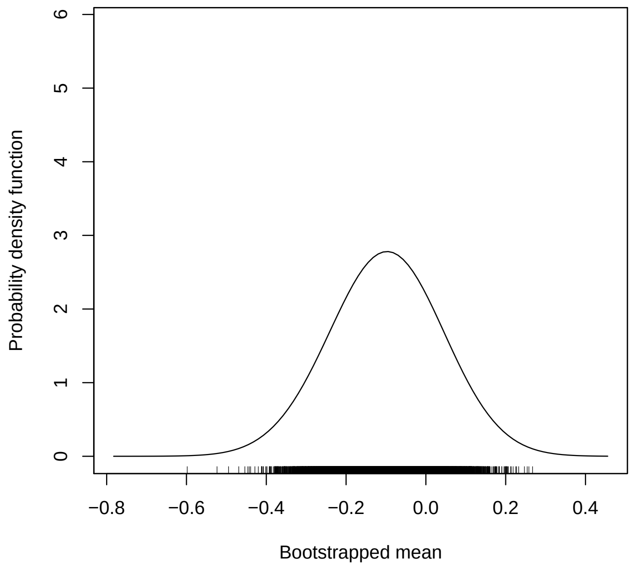
```
resample(observed)
```

```
##   [1] -2.1480 -2.3957 -1.4401 -2.7795  0.7801  1.4159 -0.8839 -2.7795
##   [9] -0.4653  1.4159  0.0774  1.3594 -2.1234  0.0599 -0.7949 -0.2190
##  [17] -0.8799 -0.7949  0.6567  1.3923 -0.5143  0.0888  0.1396 -1.9398
##  [25]  0.3335 -1.1119 -1.7378 -0.2190 -0.4666 -0.0370  1.1825 -1.1119
##  [33] -0.8839  1.4028  0.0129  0.1396  0.0129 -1.7402  0.8731 -0.7918
##  [41]  1.5963  0.7462  0.0129  0.5015  1.4028 -0.7564 -0.4098 -0.3589
##  [49]  0.9357 -0.3613 -1.8887  1.1825 -0.4666 -0.6786 -0.5143  0.4602
##  [57] -1.1119  1.8822 -0.7796  1.0341  0.3745  0.5993  0.5619  0.5701
##  [65] -0.7796  1.4028 -1.7402 -1.1259 -1.9398  0.0208 -1.4401  1.4028
##  [73] -0.0636  0.5015 -2.7795  0.7801 -0.3613 -0.4098  0.7801 -1.9398
##  [81] -0.6408 -1.1119 -0.2553  0.9921 -0.5122 -1.9292  1.5963 -0.5537
##  [89] -0.6408 -0.2341 -1.0012  0.1396  0.4602 -0.4653 -0.5338 -0.3165
##  [97]  1.3923  0.3745 -0.5338 -1.9398
```

It randomly sampled the same number of observations as was in the observed sample ($n = 100$). However, it is not sampling form the population, but from the observed sample. It is also sampling with replacement. We will use follow the same simulation procedure we did with sampling from the population, but instead we will resample from our observed data.

```
myBootstrap = do(10000) * mean(resample(observed))
#head(myBootstrap)

# Plot the distribution of bootstrapped means
sm.density(myBootstrap$mean, xlab = "Bootstrapped mean")
```



```
# Compute the mean of the bootsrapped means
mean(myBootstrap$mean)
```

```
## [1] -0.0996
```

```
# Compute the sd of the bootstrapped means
sd(myBootstrap$mean)
```

```
## [1] 0.102
```

Here the bootstrap distribution of resampled means has qualities which are based on the observed sample. In other words, the expected value of the bootstrapped means is approximately equal to the mean of the observed sample. The shape of the bootstrap distribution of resampled means is relateed to the shape of the observed sample. Those are byproducts of the resampling process. Recall that our goal was to estimate the sampling error (not the shape or mean). That is where bootstrapping shines. Notice that the bootstrapped SE is approximately equal to the SE we computed from theory (and approximated by sampling from the population through simulation), 0.1. This means, all we need is a sample of data from which we can bootstrap to obtain an approximation of the SE; we do not need the population from which that sample was drawn from.

Bootstrapping Example

Let's use the Minnesota College data to see how we can use bootstrapping in practice. Say we want to estimate the median graduation rate and we want to give an range (interval) of plausible values for this estimate. In a sense this is akin to finding a confidence interval for the median. First, we compute the sample median:

```
median(mn$gradRate)
```

```
## [1] 63.5
```

Now we want to obtain an estimate how much the sample median varies because of sampling error. To do this, we will draw bootstrap resamples from the observed data and compute the median for each of these. Then we can find the standard deviation of these bootstrapped medians.

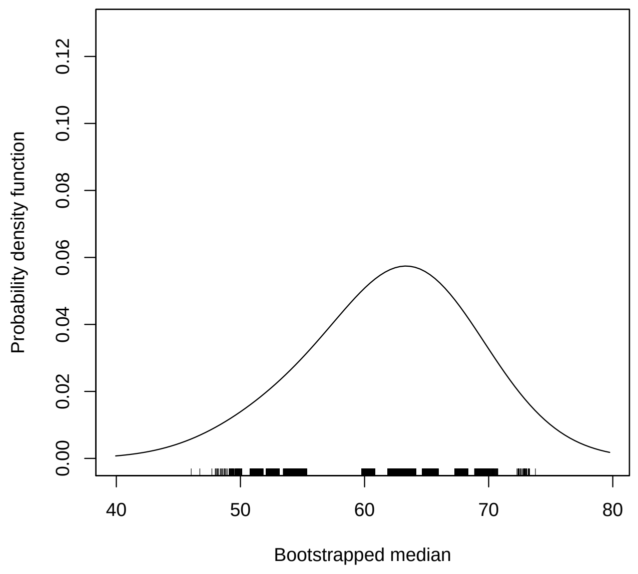
```
myBootstrap = do(10000) * median(resample(mn$gradRate))
#head(myBootstrap)

# Compute the sd of the bootstrapped medians
se_median = sd(myBootstrap$median)
se_median
```

```
## [1] 4.66
```

Combining this with our observed estimate of 63.5, we can produce an interval estimate. Typically we add/subtract two standard errors to the observed estimate in order to obtain our interval estimate. This is appropriate if the sampling distribution is symmetric. Although we don't have the sampling distribution, the bootstrap distribution should have some of the same qualities, so we will plot that distribution in order to evaluate its symmetry.

```
sm.density(myBootstrap$median, xlab = "Bootstrapped median")
```



Percentile Interval

The bootstrap distribution seems slightly left-skewed. To compensate for that, we will compute a *percentile interval* rather than adding/subtracting two SEs to the observed median. In a normal distribution, adding and subtracting two SEs is akin to encompassing the middle 95% of the sampling distribution. The percentile interval mimics this, but finds the middle 95% of the bootstrap distribution. To do this, we arrange the bootstrapped medians in order from smallest to largest. Then we drop the lowest 2.5% and the highest 2.5% (leaving the middle 95%). The smallest and largest value left indicate the endpoints of our interval. We can quickly access these values by using the `quantile()` function.

```
lower_limit = quantile(myBootstrap$median, probs = 0.025)
lower_limit
```

```
## 2.5%
## 52.6
```

```
upper_limit = quantile(myBootstrap$median, probs = 0.975)
upper_limit
```

```
## 97.5%
##  69.4
```

The percentile interval estimate is then [52.6, 69.4].

Note if we had used the $\pm 2(SE)$ method our interval would have been [54.2, 72.8]. While this is close to what we got for the percentile interval, it is different because of the asymmetry in the bootstrap distribution. When the sampling/bootstrap distribution is symmetric, the intervals prodeuce using these two methods give very similar endpoints (in the case of the normal distribution, they are exactly the same). The more asymmetric the distributio the more different the results produced by these two methods.