Have taken

- sample stat (point estimate of some parameter): $\hat{p}$ for p
- had SE

Calculated

margin of error: (for 95%) $ME = (2)(SE)$

Obtained boundaries

lower: $\hat{p} - ME$

upper: $\hat{p} + ME$

$\Rightarrow (\hat{p} - ME, \hat{p} + ME)$

Task:

- Using sampled data (changes from sample to sample)

  $\Rightarrow$ Next time your $\hat{p}$ is different

- Estimating a non-moving target p

Once we have our CI - perhaps $(0.23, 0.31)$ we might say:

> The procedure I used to construct this CI succeeds in capturing the population parameter 95% of time.

Introduction to Bootstrapping

Exam: Content
1.1 – 1.3
2.1 – 2.6
3.1 – 3.2
Friday

Thomas Scofield

September 27, 2021

The context

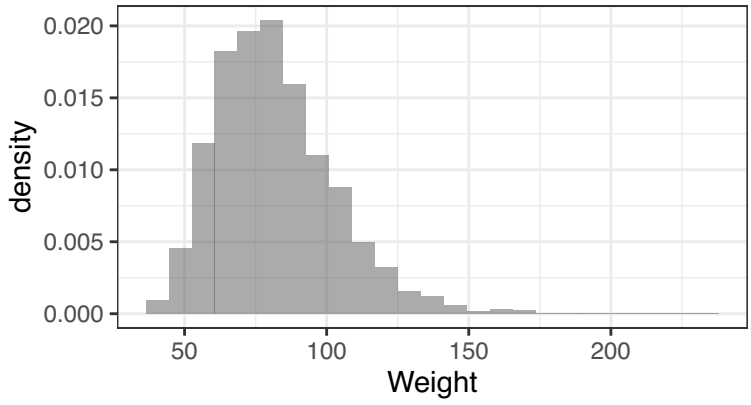
There is a population parameter we seek. Perhaps it is the average `weight` μ for people in America. For the moment, let's take our population to be the people in the **NHANES** dataset over the age of 17.

```
myPopulation <- filter(NHANES, Age > 17 & Weight > 0)
nrow(myPopulation)
```

```
## [1] 7420
```

We see there are 7420 (out of the original 10000) cases that made it through our filter. Let's look at the distribution of `Weight` as well as the population mean μ:

```
gf_dhistogram(~Weight, data=myPopulation)
```




```
mean(~Weight, data=myPopulation)
```

```
## [1] 82.00212
```

The distribution is somewhat right-skewed, and has population mean $\mu = 82.0$ (in Kg). I will now make a container for weights called `purpleBag`, evoking the image that we have written the weights of all 7420 members of the population on individual slips of paper and placed them in a purple (*purple* for *population*) bag.

```
purpleBag <- myPopulation$Weight
```

Let's draw an SRS of $n = 50$ weights from the population (purple bag) of weights. As I will use this sample several times, I will name it `orangeBag` (*orange* for *original sample*).

```
orangeBag <- sample(purpleBag, size=50)
orangeBag    # displays contents of list orangeBag
```

```
##  [1]  99.6  68.4  98.8  83.8  97.3  83.8  93.5 106.5  83.8  86.8 136.3 161.5
```

```
## [13]  82.8  55.9  88.7 105.2  77.9  98.5  62.3  78.1  66.1  82.8  68.7  86.1
## [25]  86.2  97.7  56.3  70.8  68.0  72.5  51.5 112.0 100.4  53.0  70.1 121.5
## [37]  93.9 100.0  70.2  91.2 113.7  65.4  72.4 119.7  83.5  92.9  96.8  81.9
## [49]  61.8  89.8
```

```
mean(orangeBag)
```


```
## [1] 86.928
```

The mean of the sample in this bag is calculated here. As this is the mean of the original sample, let's call it originalXbar:

```
originalXbar <- mean(orangeBag)
```

Note that it does not equal $\mu = 82.0$. It rarely does. If we want to know how often a sample mean differs by this much from μ, that can be assessed by looking at the sampling distribution of $\overline{x}$ for random samples of size $n = 50$ taken from this population.

Aside 1: Sampling distributions using SRS and i.i.d. samples

We proceed as in our sampling distribution activity Tuesday. As we have drawn one random sample of size $n = 50$ from the `purpleBag` and computed $\overline{x}$, so now we do it many times. Our original sample was an SRS, so it seems we should carry out the process as similarly as possible.

```
samplingDistSRS <- do(5000) * mean( sample( purpleBag, size=50 ))
head(samplingDistSRS)
```

```
##     mean
## 1 78.290
## 2 84.208
## 3 79.064
## 4 80.162
## 5 75.814
## 6 82.012
```

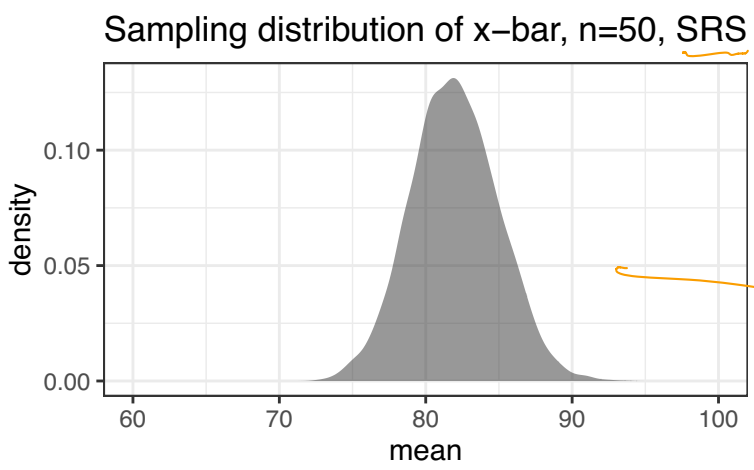
At this stage, note that we might have drawn i.i.d. samples, instead. I will do so here.

```
samplingDistIID <- do(5000) * mean( sample( purpleBag, size=50, replace=TRUE ))
head(samplingDistIID)
```

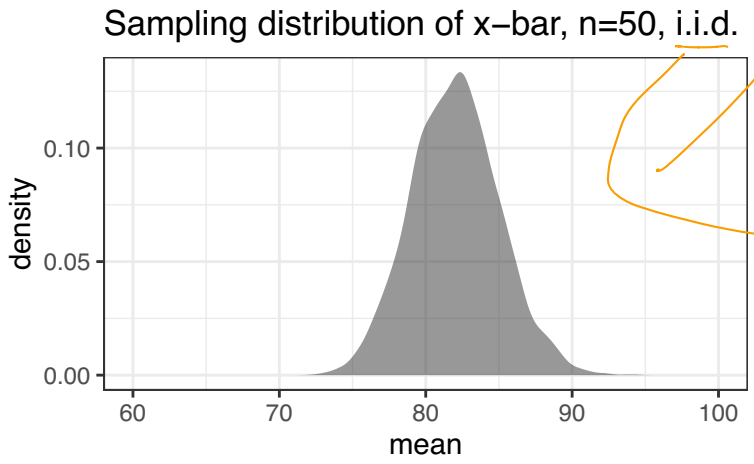
```
##     mean
## 1 85.014
## 2 81.158
## 3 84.674
## 4 79.606
## 5 77.106
## 6 75.254
```

Of course, you can expect to see any difference between the two simply by looking at the first few rows of each! Below, I produce density plots of the two for comparison. Now, can you see a difference?

```
p1 <- gf_density(~mean, data=samplingDistSRS) %>%
  gf_labs(title = "Sampling distribution of x-bar, n=50, SRS") %>%
  gf_refine(scale_x_continuous(limits=c(60,100)))
p2 <- gf_density(~mean, data=samplingDistIID) %>%
  gf_labs(title = "Sampling distribution of x-bar, n=50, i.i.d.") %>%
  gf_refine(scale_x_continuous(limits=c(60,100)))
grid.arrange(p1, p2, nrow=2)
```



(i.e., samples w/out replacement)



Nearly identical, since sample size n is insignificant compared to the size of the population.

samples with replacement

How about looking at the means (technically the mean of a distribution of means)?

```
mean(~mean, data=samplingDistSRS)
```

```
## [1] 81.97264
```

```
mean(~mean, data=samplingDistIID)
```

```
## [1] 82.01404
```

Given our previous work with sampling distributions, it should not be surprising that both are centered almost exactly at μ, and that they appear symmetric, bell-shaped. What may be surprising is how little difference there is between repeatedly sampling with replacement vs. doing so without replacement. The reason for this is the relatively small size of our samples ($n = 50$) in comparison to the size of our population (7420).

Aside 2: A 95% confidence interval for μ

What we might do, at this point, is calculate the standard error, which as we observed is about the same regardless of whether we compute it from the sampling distribution for SRS or for i.i.d. samples. Here it is from the distribution of $\overline{x}$ using i.i.d. random samples of size $n = 50$.

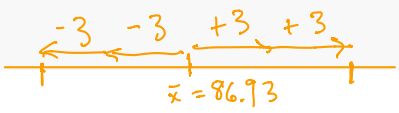
```
sd(~mean, data=samplingDistIID)
```

```
## [1] 2.996611
```

std error ≈ 3

Our **point estimate** for μ is $\overline{x} = 86.928$ from the original sample. To build a 95% CI for μ we compute the margin of error (twice the standard error), then add/subtract it from our original sample mean:

```
marginOfError <- 2 * sd(~mean, data=samplingDistIID)
originalXbar + c(-1,1) * marginOfError
```




```
## [1] 80.93478 92.92122
```

There it is, our 95% CI is $(80.93, 92.92)$. It is always possible the population parameter is not contained inside the confidence interval, but we know it is this time, since $\mu = 82.0$.

But wait! Isn't this all rather silly? We built a confidence interval in the effort to estimate μ, when we already knew it by direct calculation from the population! Let's now make the situation more realistic.

Bootstrapping

It **is** realistic to obtain a sample sample from the population (`purpleBag`) like the one we have in `orangeBag`.

It **is not** realistic for us to go back to the population several thousand times to draw more samples like the one in `orangeBag`. The implications of this include

- losing the ability to view the sampling distribution
- losing the ability to calculate $\text{SE}_{\overline{x}}$.

One idea for dealing with this problem had to await the rise of modern computers; it is known as **bootstrapping**. What we do is draw several thousand **bootstrap samples**, as they are called, not from `purpleBag`, but from `orangeBag`. As before, the size of each sample must match that of the original sample. In our current setting, they must be drawn with replacement (like an i.i.d. sample) in order to provide a variety of results. Given our work above, we can get one bootstrap sample like this:

```
resample(orangeBag)   # or, alternatively,  sample(orangeBag, replace=TRUE)
```

```
##  [1]  66.1  62.3  83.8  96.8  86.1  65.4  83.8  83.8  98.5  99.6  92.9 113.7
## [13]  96.8  86.2  82.8  92.9  93.5  97.7  55.9 136.3  78.1  81.9  78.1  91.2
## [25]  72.4  70.2 121.5  65.4 119.7  81.9  86.2  55.9  83.5  98.8  89.8  77.9
## [37]  68.0  82.8  98.5 161.5  86.8  72.5  86.1  83.5  51.5  70.8  68.7  68.7
## [49]  62.3  65.4
```

We use a bootstrap sample such as this to compute a **bootstrap statistic**, in this case a mean.

```
mean(resample(orangeBag))
```

```
## [1] 86.818
```

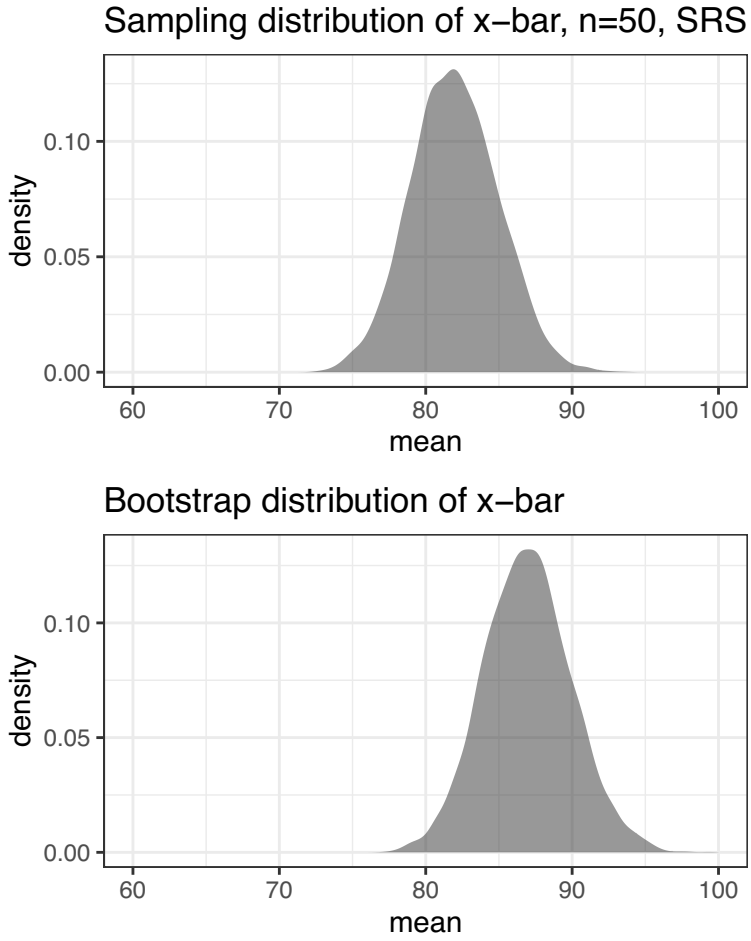
When we repeat this process over and over, the collection of means produces what we call a **bootstrap distribution**.

```
bootstrapDistMeans <- do(5000) * mean(resample(orangeBag))
head(bootstrapDistMeans)
```

```
##     mean
## 1 90.374
## 2 82.488
## 3 89.398
## 4 90.420
## 5 90.894
## 6 88.228
```

Let's look at this distribution, and compare it with the sampling distribution we viewed before.

```
p3 <- gf_density(~mean, data=bootstrapDistMeans) %>%
  gf_labs(title = "Bootstrap distribution of x-bar") %>%
  gf_refine(scale_x_continuous(limits=c(60,100)))
grid.arrange(p1, p3, nrow=2)
```



– repeated from above, given here for comparison's sake

The two are similar, but not identical, as borne out by `favstats()`:

```
favstats(~mean, data=samplingDistIID)
```

```
##    min     Q1 median     Q3   max     mean       sd    n missing
##  72.36 79.923 82.006 83.994 94.33 82.01404 2.996611 5000       0
```

```
favstats(~mean, data=bootstrapDistMeans)
```

```
##     min    Q1 median     Q3    max     mean       sd    n missing
##  77.346 84.89  86.89 88.905 99.346 86.94828 2.987145 5000       0
```

Note they have different means. The center of the sampling distribution is almost exactly μ, reflecting that $\overline{x}$ is an unbiased estimator of μ. The center of the bootstrap distribution is almost exactly $\overline{x} = 86.928$, the mean of our original sample (i.e., the mean of values in `orangeBag`). When you think about it, that isn't so surprising, given that every bootstrap sample was *drawn* from `orangeBag`.

But though these distributions are centered at different places, they have approximately the same spreads. In the results above, the standard error (`sd` from the sampling distribution) is $\text{SE}_{\overline{x}} = 2.997$. The `sd` from the bootstrap distribution is 2.987. While the two numbers are not always *this* close to each other, the reason for bootstrapping is precisely so we can use its `sd` to approximate the standard error.

You try it

The Lock5 dataset **CommuteAtlanta** contains variables related to travel times for 500 people who work in Atlanta, Georgia. This is a sample, already drawn for you from a larger population of commuters working in Atlanta. So treat the `Distance` variable like `orangeBag`. Without access to the larger population, we cannot obtain the parameter μ. Instead, use bootstrapping to find an approximately 95% confidence interval for μ.

Once you have achieved this, visit the StatKey website. From the apps listed there, select "CI for Single Mean, Median, St.Dev." in the "Bootstrap Confidence Intervals" column. In the top left there is a drop-down menu that has "Mustang Price" pre-selected. Use it to select "Atlanta Commute (Distance)". Try your hand at generating a bootstrap distribution and producing a point estimate and standard error that could be used to build a 95% confidence interval like you have just done using RStudio.

Try this (fixed from what I wrote on the board during class)

```
bootstrapDist <- do(3000) * mean(resample(CommuteAtlanta$Distance))
favstats(~mean, data = bootstrapDist)
```

use s.d. as the approximate SE

```
mean(~Distance, data = CommuteAtlanta)
```

this computes the mean $\bar{x}$ of the original sample which should be about the same as the mean of the bootstrap distribution reported by favstats