day 2.6 notes

T.Scofield

Sampling distributions for $\hat{p}$

Let's create a population. For our purposes, let's make this population contain 70% successes and 30% failures:

```
population <- c(rep(0,3), rep(1,7))
population
```

```
 [1] 0 0 0 1 1 1 1 1 1 1
```

This, throughout the section, will be the population I sample from.

Let's also decide on a sample size, say $n = 12$.

To draw a single sample of size 12 and compute the proportion of successes in that sample, we might use

```
x <- resample(population, size=12)
x
```

```
 [1] 1 1 0 1 1 1 1 1 0 1 0 0
```

```
prop(~ (x==1))
```

```
prop_TRUE
0.6666667
```

A streamlined (into a single line) version looks like this:

```r
prop(~ (resample(population, size=12)==1 ))
```

```
prop_TRUE
0.5833333
```

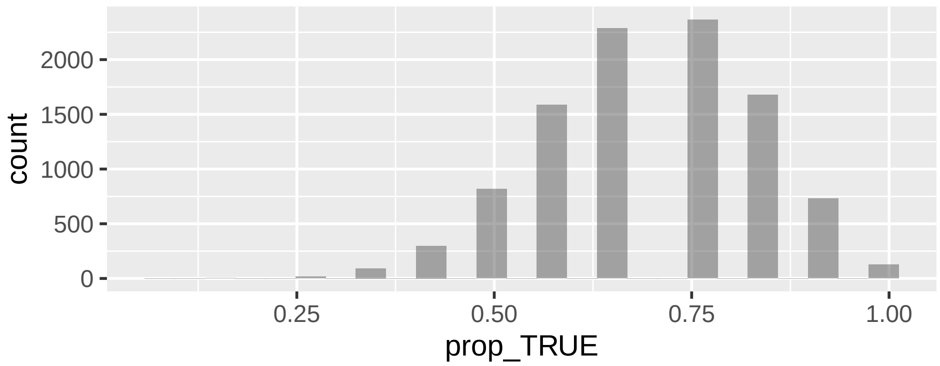
To simulate the sampling distribution of $\hat{p}$ (i.e., sample proportions when a sample of size 12 is drawn, with replacement, from this population), I need to repeat this process many times. Here, I use `do(10000) *` to repeat it 10000 times.

```r
manyPhats <- do(10000) * prop(~ (resample(population, size=12)==1 ))
head(manyPhats)    # shows the first few results
```

```
  prop_TRUE
1 0.8333333
2 0.5833333
3 0.9166667
4 1.0000000
5 0.5000000
6 0.8333333
```

Let's view all 10000 of the outcomes using a histogram:

```r
gf_histogram(~prop_TRUE, data=manyPhats, numbins=13)
```



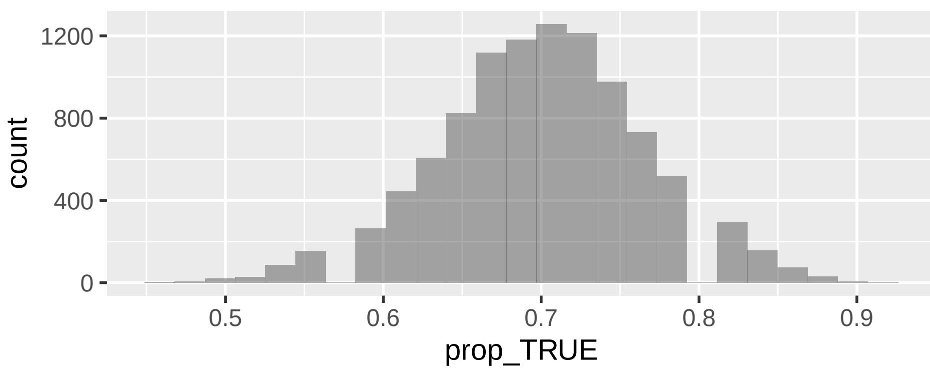
Same idea, but with a different sample size

Let's again sample from this population, but draw samples of greater size, say $n = 48$.

```
manyPhats48 <- do(10000) * prop(~ (resample(population, size=48)==1 ))
head(manyPhats48)
```

```
  prop_TRUE
1 0.6041667
2 0.6875000
3 0.7916667
4 0.6666667
5 0.6458333
6 0.7291667
```

```
gf_histogram(~prop_TRUE, data=manyPhats48)
```



Observe

- For both simulated distributions of $\hat{p}$ (the one with $n = 12$, the other with $n = 48$), the mean appears to be around $0.7 = 70\%$, which is the overall proportion of successes in the population.
- The simulated distribution is more symmetric and bell-shaped for the $n = 48$ case than for the $n = 12$ case. The $n = 48$ histogram is less spread out than is the $n = 12$ case.

Sampling distributions for $\bar{x}$

Means require a quantitative variable. In sports many quantitative variables are measured. In what follows, I will use as my population Major League Baseball players in 2018 (ones who batted at least 100 times). In the first line of code, I import a data set on these players.

```
mlbStats18 <- read.csv("https://scofield.site/teaching/data/csv/mlb18abEligible.csv")
head(mlbStats18, n=3)
```

```
  X          name team position games  AB   R   H doubles triples HR RBI walks
1 1      Betts, M  BOS       RF   136 520 129 180      47       5 32  80    81
2 2   Martinez, J  BOS       LF   150 569 111 188      37       2 43 130    69
3 3     McNeil, J  NYM       2B    63 225  35  74      11       6  3  19    14
  strike_outs stolen_bases caught_stealing_base   AVG   OBP   SLG   OPS
1          91           30                    6 0.346 0.438 0.640 1.078
2         146            6                    1 0.330 0.402 0.629 1.031
3          24            7                    1 0.329 0.381 0.471 0.852
```

When I use `resample()` on a data frame, it results in the selection of cases, but with all variables intact for those cases. Here, I choose $n = 10$ players at random, with replacement:

```
resample(mlbStats18, size=10)
```

```
      X          name team position games  AB  R   H doubles triples HR RBI
367 367    Frazier, T  NYM       3B   115 408 54  87      18       0 18  59
224 224  Pederson, J  LAD       LF   148 395 65  98      27       3 25  56
55   55     Peraza, J  CIN       SS   157 632 85 182      31       4 14  58
34   34    Martini, N  OAK       LF    55 152 26  45       9       3  1  19
178 178     Andrus, E  TEX       SS    97 395 53 101      20       3  6  33
281 281    Heredia, G  SEA       LF   125 292 29  69      14       1  5  19
275 275    Wieters, M  WSH        C    76 235 24  56       8       0  8  30
340 340      Smith, D  NYM       1B    56 143 14  32      11       1  5  11
376 376      Jones, J  DET       LF   129 429 54  89      22       6 11  34
167 167  Castillo, W  CWS        C    49 170 17  44       7       0  6  15
    walks strike_outs stolen_bases caught_stealing_base   AVG   OBP   SLG   OPS
367    48         112            9                    4 0.213 0.303 0.390 0.693
224    40          85            1                    5 0.248 0.321 0.522 0.843
55     29          75           23                    6 0.288 0.326 0.416 0.742
34     21          36            0                    0 0.296 0.397 0.414 0.811
178    28          66            5                    3 0.256 0.308 0.367 0.675
```

```
281    32           52            2                       4 0.236 0.318 0.342 0.661
275    30           45            0                       1 0.238 0.330 0.374 0.704
340     4           47            0                       0 0.224 0.255 0.420 0.675
376    24          142           13                       5 0.207 0.266 0.364 0.630
167     9           46            1                       0 0.259 0.304 0.406 0.710
    orig.id
367     367
224     224
55       55
34       34
178     178
281     281
275     275
340     340
376     376
167     167
```

The variable I've chosen to focus on is `hits`, the column labeled "H". I'm demonstrating the sampling distribution for the sample mean ($\bar{x}$), so I need code that draws a random sample of size $n = 10$ and computes the mean number of hits for the 10 players selected.

```
mean(~H, data=resample(mlbStats18, size=10))  # calculates a single mean
```

```
[1] 103.7
```

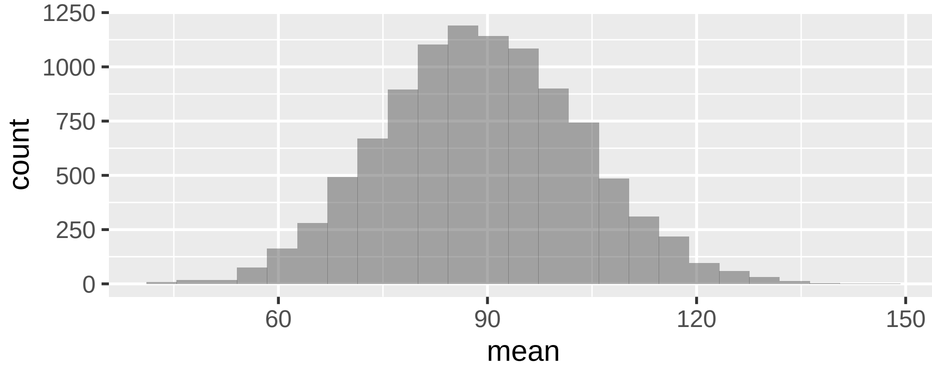
To draw 5 samples, calculating the sample mean number of hits for each one.

```
do(5) * mean(~H, data=resample(mlbStats18, size=10))
```

```
   mean
1 111.0
2  87.8
3  81.0
4  73.3
5  95.7
```

Again, a simulated sampling distribution requires a lot more iterations than 5 of them. So, let's up the number to 10000, this time viewing a histogram (simulated sampling distribution) of means:

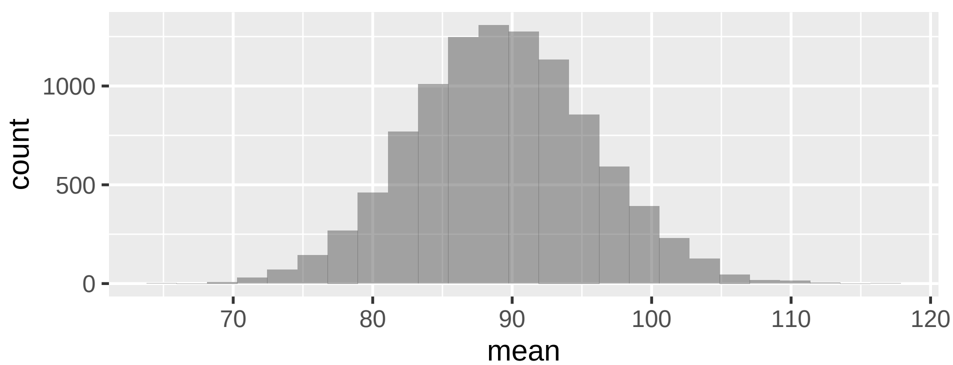
```
manyXbars <- do(10000) * mean(~H, data=resample(mlbStats18, size=10))
gf_histogram(~mean, data=manyXbars)
```



Repeating for samples of size $n = 50$

Let's go to samples of size 50.

```
manyXbars <- do(10000) * mean(~H, data=resample(mlbStats18, size=50))
gf_histogram(~mean, data=manyXbars)
```



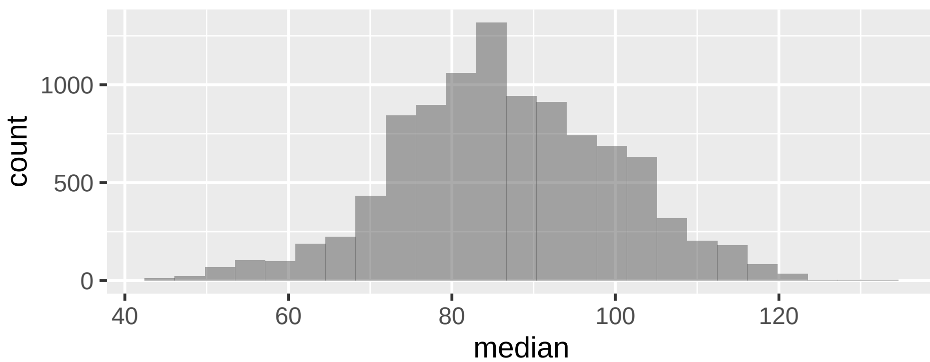
Observe

- For both simulated distributions of $\bar{x}$ (the one with $n = 10$, the other with $n = 50$), the mean appears to be around 88 or so.
- The simulated distribution is more symmetric and bell-shaped for the $n = 50$ case than for the $n = 10$ case. The $n = 50$ histogram is less spread out than is the $n = 10$ case.

Sampling distribution for the sample median

When our variable is quantitative, there are lots of sample statistics that might be computed. The mean $\bar{x}$ was done in the last section (and is by far the most popular), but one might produce a sampling distribution for the sample median, or for the 0.3-quantile, or for the sample standard deviation, just to name a few. Here, I'll do it for the median, when samples drawn from `mlbStats18` are of size $n = 30$. Look over the differences in code.

```
manyMedians <- do(10000) * median(~H, data=resample(mlbStats18, size=30))
gf_histogram(~median, data=manyMedians)
```



Sampling distribution for the sample correlation

If we are to compute a correlation from a sample, we will need **two** variables. Perhaps we are interested in correlation between homeruns (`HR`) and strike-outs (`strike_outs`). We will draw samples from `mlbStats18` as before, this time calculating r, the correlation coefficient each time. Code that does so just once, using a sample size of $n = 20$, follows:

```
cor(strike_outs ~ HR, data=resample(mlbStats18, size=20))
```

```
[1] 0.6916527
```

Again, we obtain a simulated sampling distribution for the sample correlation r by carrying this out many times:

```
manyCors <- do(10000) *  cor(strike_outs ~ HR, data=resample(mlbStats18, size=20))
head(manyCors)
```

```
        cor
1 0.7512196
2 0.7682710
3 0.6469283
4 0.8055510
5 0.7961862
6 0.8247882
```

I've stored the results in a data frame called `manyCors`. The column where they are stored is called `cor`. So, a histogram displaying the distribution of values of `r`:

```
gf_histogram(~ cor, data=manyCors)
```