POLSCI 9590: Methods I

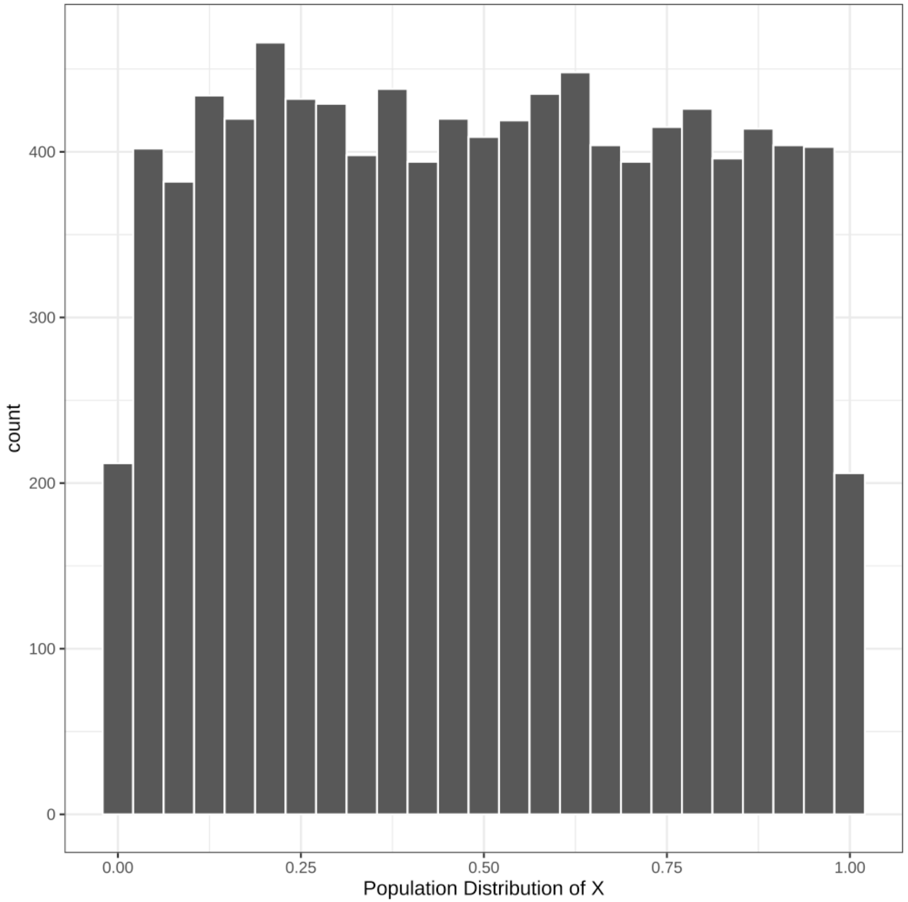
Sampling and Generalization

Dave Armstrong

Population Distribution

```r
set.seed(2543)
pop <- runif(10000, 0, 1)
```

```r
ggplot() +
  geom_histogram(aes(x=pop),
                 bins=25,
                 col="white") +
  theme_bw() +
  labs(x="Population Distribution of X")
```



The true population mean μ is:

```r
mean(pop)
```

```
## [1] 0.49728
```

and the true population SD σ is:

```r
sqrt(sum((pop - mean(pop))^2)/length(pop))
```

```
## [1] 0.2872507
```

Random Sample

Let's take a sample of size 100 at random from the population

```
samp <- sample(pop, 100, replace=TRUE)
# sample mean
mean(samp)
```

```
## [1] 0.4871019
```

```
# sample sd
sd(samp)
```

```
## [1] 0.2748858
```

The **sampling error** is $\bar{x} - \mu$:

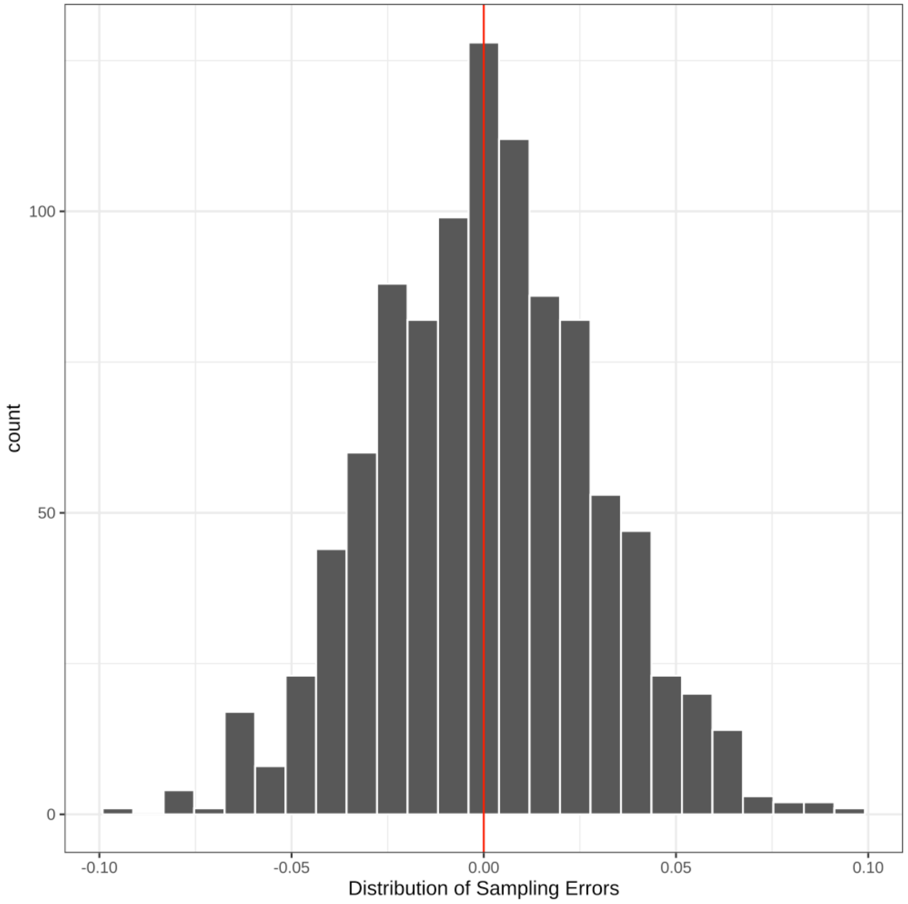
```
mean(samp) - mean(pop)
```

```
## [1] -0.0101781
```

Repeat

What if we did this 1000 times?

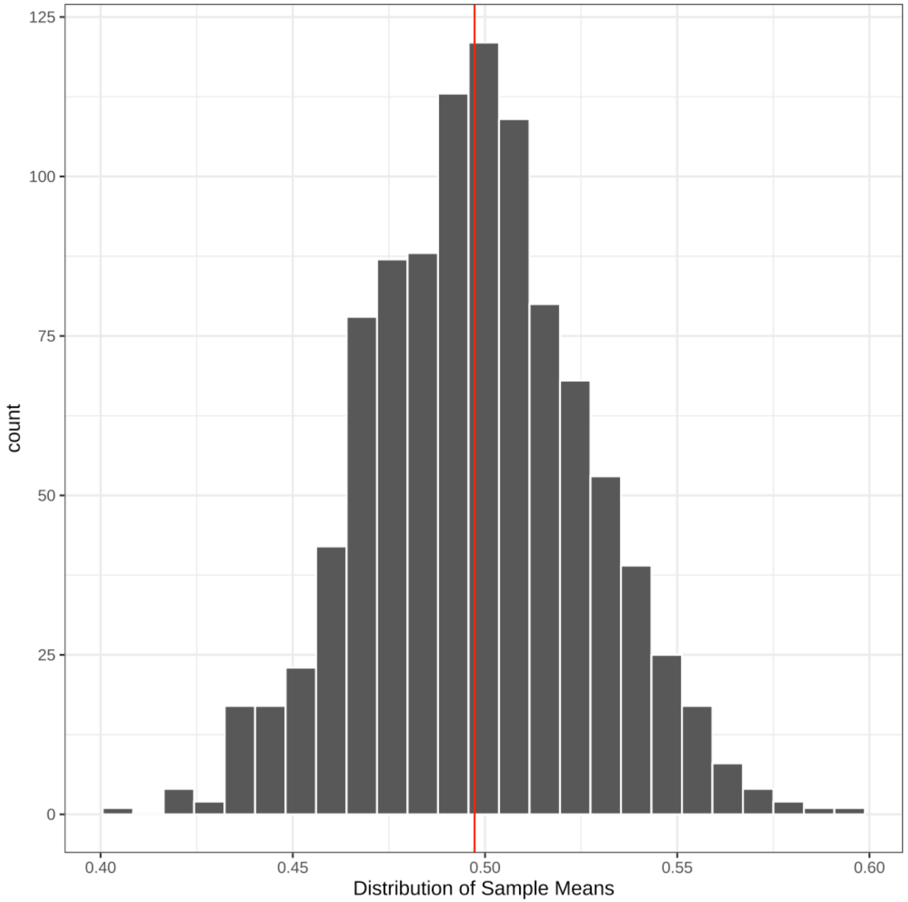
```
means <- replicate(1000,
       mean(sample(pop, 100, replace=TRUE)))
samp.errors <- means - mean(pop)
ggplot() +
  geom_histogram(aes(x=samp.errors),
                 bins=25,
                 col="white") +
  theme_bw() +
  geom_vline(xintercept=0, col="red") +
  labs(x="Distribution of Sampling Errors")
```



Sampling Distribution

```
ggplot() +
  geom_histogram(aes(x=means),
                 bins=25,
                 col="white") +
  theme_bw() +
  geom_vline(xintercept=mean(pop),
             col="red") +
  labs(x="Distribution of Sample Means")
```

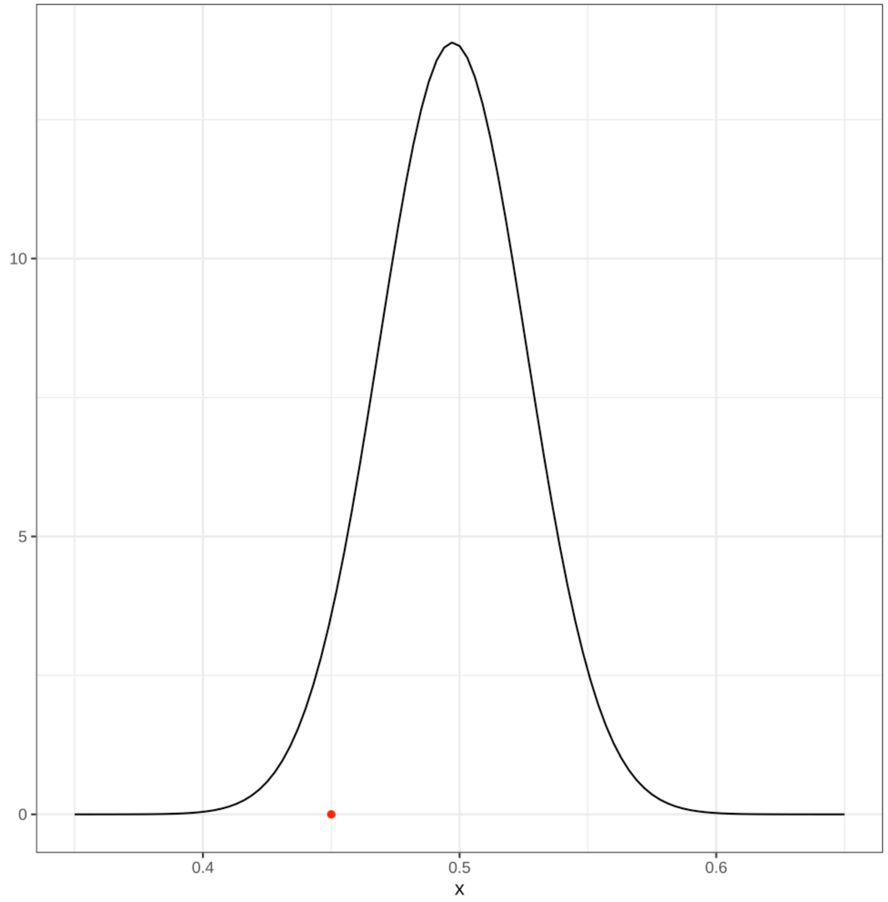
The **sampling distribution** is the distribution of sample means around the true, but generally unknown, population mean.



Sample Statistics

Sample Statistics

We only ever have **one** sample statistic.
How do we learn about the population from one value?

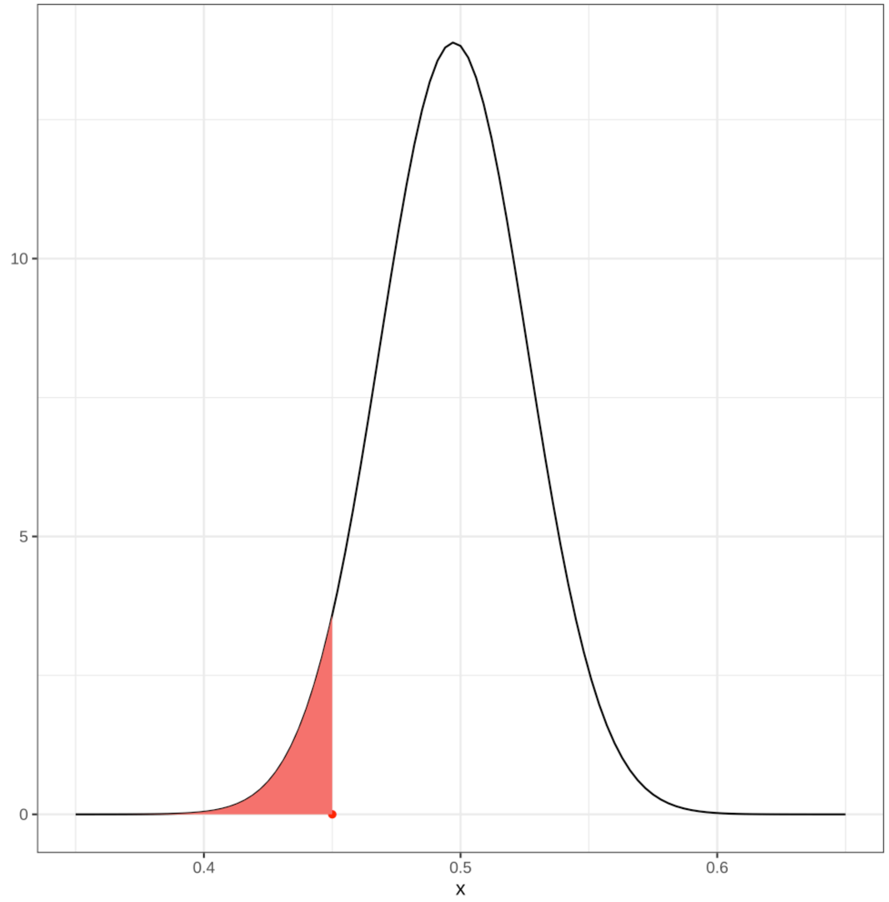


What we want to do

We would like to be able to identify how likely it is that we would observe our sample statistic $(\bar{x} = .45)$ in the population.

```
pnorm(.45, mean(pop), true.se)
```

```
## [1] 0.04988717
```



What we have.

We have a couple of things.

- A value for the sample mean, $\bar{x}$ (which is an estimate of the true population mean μ.)
- A value for the sample standard deviation s (which is an estimate of the true population standard deviation σ.)

Confidence Interval

A slightly circuitous definition:

A confidence is an interval created such that $(1-\alpha)\%$ *of the intervals we could make from different samples will cover the true, but unknown population value*.

Important points:

1. We can't make any interesting probability statements about a single interval. It either contains the true value or it does not.
   - The interesting probability statements are about the set of intervals of which we have, but one instance.
2. We don't know which of the above situations we're in.
   - if α is sufficiently small, then we are *willing to bet* that we are in one of the "good" samples.

Confidence Interval Formulae

If we know $\sigma_{\bar{x}}$ (unlikely)

$$\bar{x} \pm z_{\text{crit}} \frac{\sigma_x}{\sqrt{n}}$$

If we have to estimate $\hat{\sigma}_{\bar{x}} = s_{\bar{x}}$, then:

$$\bar{x} \pm z_{\text{crit}} \frac{s_x}{\sqrt{n}}$$

If we have small samples $(n < 120)$:

$$\bar{x} \pm t_{\text{crit}} \frac{s_x}{\sqrt{n}}$$

Note: the formula doesn't require us to know μ, which is nice.

Confidence Intervals in Software

R Python Stata

```
library(uwo4419)
confidenceInterval(samp, distr = "norm")
```

```
##   Estimate   CI lower   CI upper Std. Error
## 0.48710189 0.43322527 0.54097851 0.02748858
```

```
confidenceInterval(samp, distr = "t")
```

```
##   Estimate   CI lower   CI upper Std. Error
## 0.48710189 0.43255859 0.54164519 0.02748858
```

Or `mean_cl_normal()` function from `ggplot2`.

```
mean_cl_normal(samp)
```

```
##           y      ymin      ymax
## 1 0.4871019 0.4325586 0.5416452
```

CES data

R   Python   Stata

```r
library(rio)
library(tidyr)
ces <- import("ces19.dta")
confidenceInterval(ces$market)
```

```
##     Estimate     CI lower     CI upper   Std. Error
## -0.285597485 -0.300882915 -0.270312055  0.007798832
```

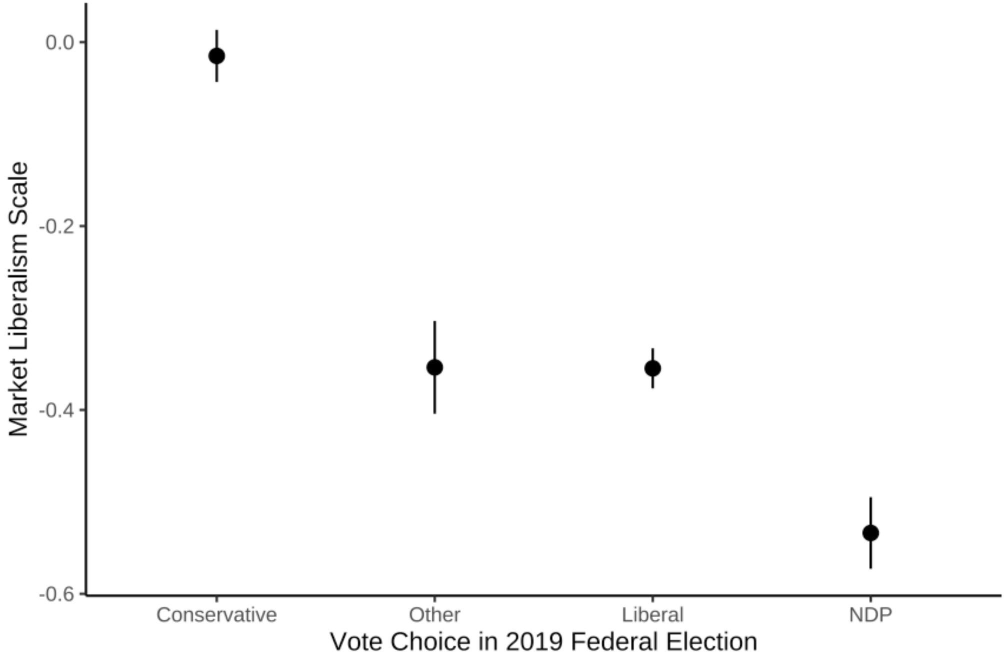
```r
x <- ces %>%
  filter(!is.na(vote) & !is.na(market)) %>%
  mutate(vote = factorize(vote)) %>%
  group_by(vote) %>%
  summarise(ci = list(mean_cl_normal(market))) %>%
  unnest(ci)
x
```

```
## # A tibble: 4 × 4
##   vote                y    ymin    ymax
##   <fct>           <dbl>   <dbl>   <dbl>
## 1 Liberal      -0.355  -0.377  -0.333
## 2 Conservative -0.0149 -0.0433  0.0134
## 3 NDP          -0.534  -0.573  -0.495
## 4 Other        -0.354  -0.404  -0.303
```

Plotting Confidence Intervals

R | Python | Stata

```
ggplot(x, aes(x=reorder(vote, -y, mean), y=y,
              ymin=ymin,
              ymax=ymax)) +
  geom_pointrange() +
  theme_classic() +
  labs(x="Vote Choice in 2019 Federal Election",
       y="Market Liberalism Scale")
```



Exercises

Using the GSS data, do the following:

1. Calculate the confidence interval of `resilience`. Make both 95% and 99% confidence intervals.
2. Plot the confidence intervals of `resilience` for the different groups identified by `SRH_115`.