Problem set 5

Your name here

Due 11/1/2022 at 5pm

NOTE1: Start with the file ***ps5_2022.Rmd*** *(available from the github repository at https://github.com/UChicago-pol-methods/IntroQSS-F22/tree/main/assignments). Modify that file to include your answers. Make sure you can "knit" the file (e.g. in RStudio by clicking on the* ***Knit*** *button). Submit both the Rmd file and the knitted PDF via Canvas*

Question 1

(1a) **Write code to draw one sample of size 5 from a normal distribution with mean 3 and variance of 2.**

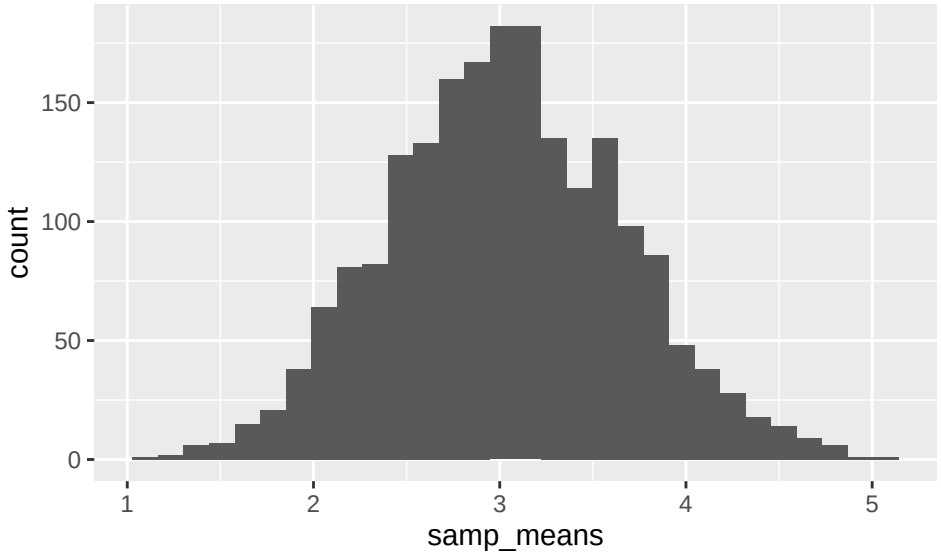
```
rnorm(n = 5, mean = 3, sd = sqrt(2))
```

```
## [1]  3.002096  4.821360 -1.386288  2.935653  2.466032
```

(1b) **Draw $m = 2000$ such samples, compute the sample mean for each sample, and plot a histogram of the result.**

```
samp_means <- replicate(2000, mean(rnorm(n = 5, mean = 3, sd = sqrt(2))))
tibble(samp_means) |>
  ggplot(aes(x = samp_means)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



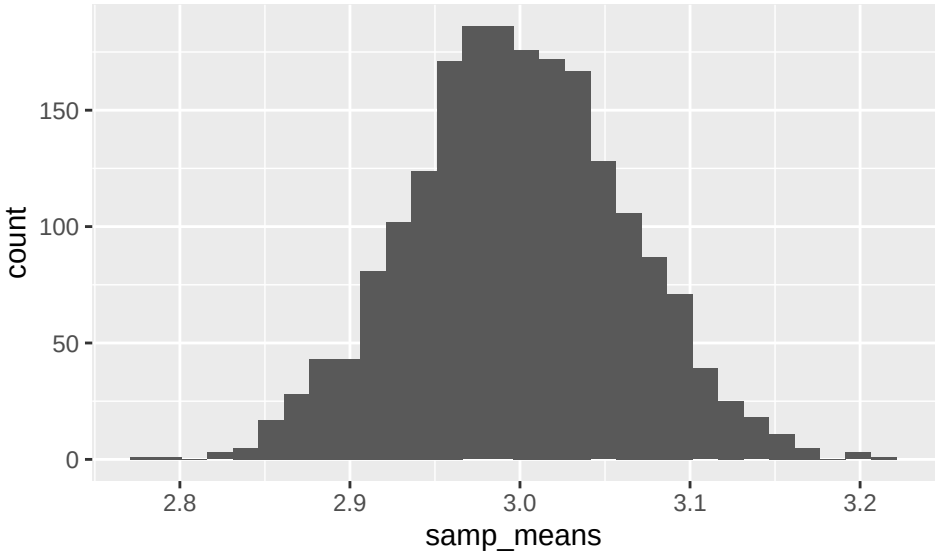
(1c) **Compute the variance across your sample means. Compare it to the theoretical sampling variance of the sample mean. Explain any difference you find.**

Answer: The variance of the sample means as computed via `var(samp_means)` is 0.3991. The theoretical sampling variance ($\mathrm{V}[X]/n$) is 2/5.

(1d) **Repeat (1b) and (1c) but set the sample size to 500.**

```
samp_means <- replicate(2000, mean(rnorm(n = 500, mean = 3, sd = sqrt(2))))
tibble(samp_means) |>
  ggplot(aes(x = samp_means)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Answer: The variance of the sample means as computed via `var(samp_means)` is 0.004. The theoretical sampling variance ($\mathrm{V}[X]/n$) is $2/500 = .004$.

(1e) **Show the histogram of Democratic vote share by county in the US in the 2020 presidential election. (You used this data in problem set 2. You might use the `pull()` function to extract a single column.) Does the distribution look symmetric?**

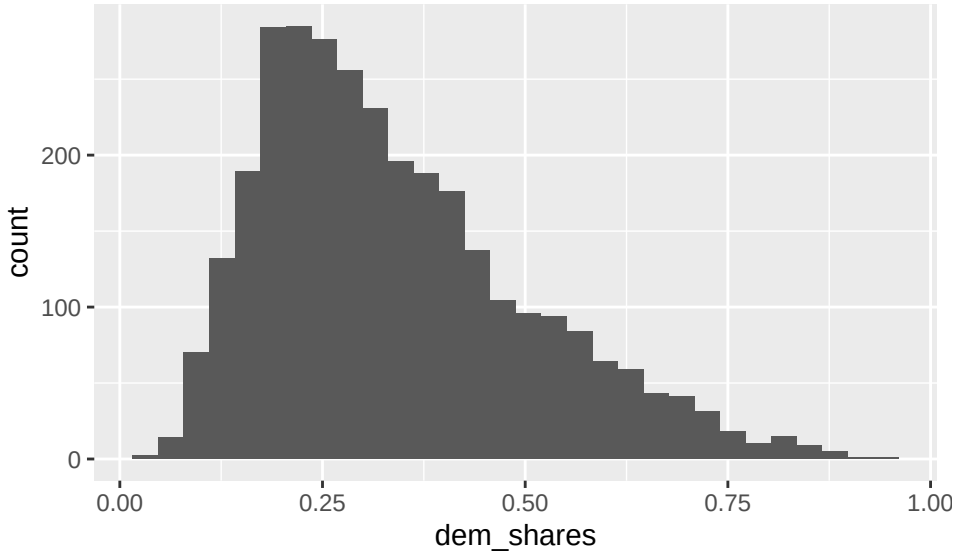
```
data_path <- "./../hw2/"
df <- read_csv(str_c(data_path, "tidy_county_pres_results.csv"))
```

```
## Rows: 49776 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): county, state
## dbl (5): FIPS, year, total_vote, dem_vote, rep_vote
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dem_shares <- df |>
  mutate(dem_share = dem_vote/(dem_vote + rep_vote)) |>
  filter(year == 2020) |>
  pull()

tibble(dem_shares) |>
  ggplot(aes(x = dem_shares)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

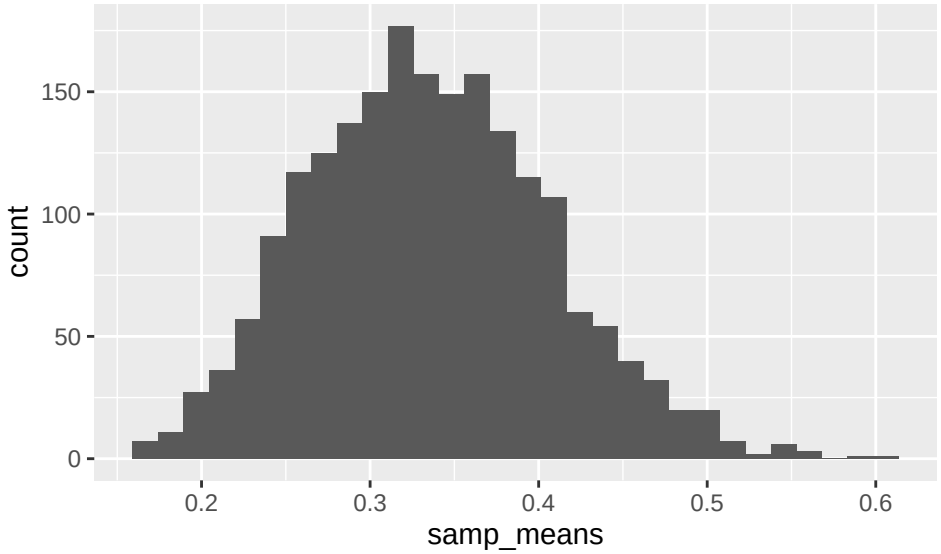


No it is not symmetric. The modal county gave only about 25% support to Biden, which is well below the average (0.34). The sample size is too small for the CLT to really "kick in", so the distribution of sample means has the same skew as the distribution of X.

(1f) **Obtain 2000 random samples of 5 counties (without replacement), compute the mean of Democratic vote share for each sample, and plot the distribution of sample means. Does the distribution look symmetric?**

```
samp_means <- replicate(n = 2000, mean(sample(dem_shares, size = 5)))
tibble(samp_means) |>
  ggplot(aes(x = samp_means)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



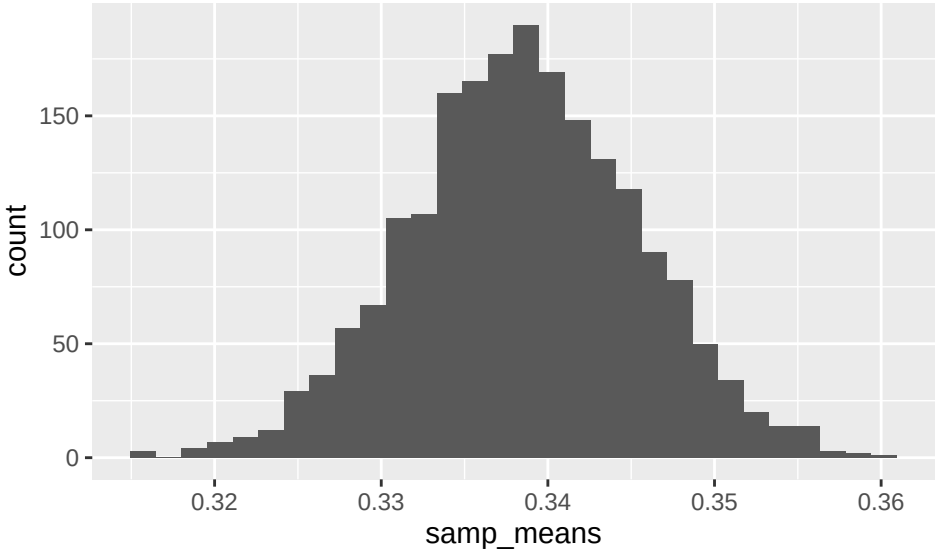
(1g) **Compute the variance across your sample means. Compare it to the theoretical variance of the sample mean. Explain any difference you find.**

Answer: The variance of the sample means is 0.005. The theoretical variance is 0.0053034.

(1h) **Repeat (1f) and (1g) but set the sample size to 500.**

```
samp_means <- replicate(n = 2000, mean(sample(dem_shares, size = 500)))
tibble(samp_means) |>
  ggplot(aes(x = samp_means)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Now it looks symmetric.

The variance of the sample means is 0.000047. The theoretical variance is 0.000053.

The reason for the difference is that we no longer have i.i.d. samples because 500 is pretty large compared to 3111 counties total. An easy way to see why the variance of the sample mean will be lower than its theoretical value as the sample size gets close to the population size is to consider what happens when you sample without replacement and the sample size *equals* the population size – then the variance across sample means will be zero even though the theoretical variance for an iid sample will be $\mathrm{V}[X]/n > 0$.

Question 2

(2a) **Using the plug-in sample variance (Definition 3.2.18 in Aronow & Miller) as a guide, write down the formula for plug-in sample covariance.**

Answer: Because covariance of X and Y can be written $\mathrm{Cov}[X, Y] = E[XY] - E[X]E[Y]$, the plug-in sample covariance is $\overline{XY} - \overline{X}\,\overline{Y}$.

(2b) **Pretending for a moment that U.S. county election results in 2020 are a sample from a super-population, use the formula you wrote above to estimate the covariance between the total votes cast in a county in 2020 and the proportion of votes cast for the Democrat in 2020. Compare it to the unbiased sample covariance, which you can compute using `R`'s `cov()` function.**

```
df |>
  mutate(dem_share = dem_vote/(dem_vote + rep_vote)) |>
  filter(year == 2020) |>
  summarize(`Plug-in sample covariance` = mean(total_vote*dem_share) -
              mean(total_vote)*mean(dem_share),
            `Unbiased sample covariance` = cov(total_vote, dem_share)) -> comparison
comparison |> t()
```

```
##                                [,1]
## Plug-in sample covariance  9364.934
## Unbiased sample covariance 9367.946
```

Just as with the sample variance, the plug-in version is slightly lower than the unbiased version. Recall that covariance is $\mathrm{E}\left[(X - \mathrm{E}\left[X\right])(Y - \mathrm{E}\left[Y\right])\right]$, which can be thought of as the average product of X's deviations from its mean and Y's deviations from its mean. But these means must be estimated, which leads to the deviations being underestimated, and therefore the average product of deviations being underestimated.

The unbiased version corrects this by multiplying by $n/(n-1)$:

```
as.numeric(comparison[1]*(3111/3110))
```

```
## [1] 9367.946
```

```
as.numeric(comparison[2])
```

```
## [1] 9367.946
```