

Tarea 07

Ita Santiago

```
library(tidyverse)
library(estcomp)
library(patchwork)
```

Conteo rápido

En México, las elecciones tienen lugar un domingo, los resultados oficiales del proceso se presentan a la población una semana después. A fin de evitar proclamaciones de victoria injustificadas durante ese periodo el INE organiza un conteo rápido.

El conteo rápido es un procedimiento para estimar, a partir de una muestra aleatoria de casillas, el porcentaje de votos a favor de los candidatos en la elección.

En este ejercicio deberás crear intervalos de confianza para la proporción de votos que recibió cada candidato en las elecciones de 2006. La inferencia se hará a partir de una muestra de las casillas similar a la que se utilizó para el conteo rápido de 2006.

El diseño utilizado es *muestreo estratificado simple*, lo que quiere decir que:

- i) se particionan las casillas de la población en estratos (cada casilla pertenece a exactamente un estrato),
y
- ii) dentro de cada estrato se usa *muestreo aleatorio* para seleccionar las casillas que estarán en la muestra.

En este ejercicio (similar al conteo rápido de 2006):

- Se seleccionó una muestra de 7,200 casillas
- La muestra se repartió a lo largo de 300 estratos.
- La tabla `strata_sample_2006` contiene en la columna N el número total de casillas en el estrato y en n el número de casillas que se seleccionaron en la muestra, para cada estrato:

```
glimpse(strata_sample_2006)
```

```
## Rows: 300
## Columns: 3
## $ stratum <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
## $ n      <int> 20, 23, 24, 31, 29, 37, 26, 21, 25, 24, 22, 24, 25, 21, 25, ...
## $ N      <int> 369, 420, 440, 570, 528, 664, 474, 373, 457, 430, 408, 432, ...
```

- La tabla `sample_2006` en el paquete `estcomp` contiene para cada casilla:
 - el estrato al que pertenece: `stratum`
 - el número de votos que recibió cada partido/coalición: `pan`, `pri_pvem`, `panal`, `prd_pt_convergencia`, `psd` y la columna `otros` indica el número de votos nulos o por candidatos no registrados.

– el total de votos registrado en la casilla: total.

```
glimpse(sample_2006)
```

```
## Rows: 7,200
## Columns: 11
## $ polling_id <int> 74593, 109927, 112039, 86392, 101306, 86044, 56057, 841...
## $ stratum <dbl> 106, 194, 199, 141, 176, 140, 57, 128, 283, 289, 77, 82...
## $ edo_id <int> 16, 27, 28, 20, 24, 20, 15, 19, 9, 9, 15, 15, 19, 27, 3...
## $ rural <dbl> 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1...
## $ pri_pvem <int> 47, 131, 51, 145, 51, 150, 117, 118, 26, 27, 120, 47, 1...
## $ pan <int> 40, 10, 74, 64, 160, 20, 119, 205, 65, 32, 119, 41, 113...
## $ panal <int> 0, 0, 2, 2, 0, 0, 2, 8, 5, 0, 4, 1, 3, 1, 0, 2, 1, 1, 0...
## $ prd_pt_conv <int> 40, 147, 57, 139, 64, 166, 82, 73, 249, 338, 149, 187, ...
## $ psd <int> 0, 1, 2, 1, 14, 1, 0, 9, 7, 14, 12, 7, 7, 3, 0, 10, 14,...
## $ otros <int> 9, 8, 2, 14, 1, 11, 24, 13, 2, 7, 7, 9, 8, 7, 25, 11, 2...
## $ total <int> 136, 297, 188, 365, 290, 348, 344, 426, 354, 418, 411, ...
```

```
sample_2006 %>%
  select(polling_id, stratum, pri_pvem:total) %>%
  pivot_longer(names_to = "party", values_to = "votes",
    cols = pri_pvem:otros) %>%
  group_by(stratum, party) %>%
  summarise(Y = sum(votes),
    X = sum(total), .groups = 'drop')
```

```
## # A tibble: 1,800 x 4
##   stratum party      Y      X
##   <dbl> <chr>   <int> <int>
## 1      1 otros     181  6425
## 2      1 pan      2941  6425
## 3      1 panal     119  6425
## 4      1 prd_pt_conv 1320  6425
## 5      1 pri_pvem   1677  6425
## 6      1 psd       187  6425
## 7      2 otros     204  7790
## 8      2 pan      3153  7790
## 9      2 panal      86  7790
## 10     2 prd_pt_conv 2064  7790
## # ... with 1,790 more rows
```

Una de las metodologías de estimación, que se usa en el conteo rápido, es *estimador de razón* y se contruyen intervalos de 95% de confianza usando el método normal con error estándar bootstrap. En este ejercicio debes construir intervalos usando este procedimiento.

Para cada candidato:

1. Calcula el estimador de razón combinado, para muestreo estratificado la fórmula es:

$$\hat{p} = \frac{\sum_h \frac{N_h}{n_h} \sum_i Y_{hi}}{\sum_h \frac{N_h}{n_h} \sum_i X_{hi}}$$

donde:

- \hat{p} es la estimación de la proporción de votos que recibió el candidato en la elección.
- Y_{hi} es el número total de votos que recibió el candidato en la i -ésima casillas, que pertenece al h -ésimo estrato.
- X_{hi} es el número total de votos en la i -ésima casilla, que pertenece al h -ésimo estrato.
- N_h es el número total de casillas en el h -ésimo estrato.
- n_h es el número de casillas del h -ésimo estrato que se seleccionaron en la muestra.

El siguiente código estima las proporciones para todos los partidos puedes utilizarlo o escribir tu propio código

```
prop_x_party <- sample_2006 %>%
  select(polling_id, stratum, pri_pvem:total) %>% # columnas relevantes
  pivot_longer(names_to = "party", values_to = "votes",
               cols = pri_pvem:otros) %>% # alargamos
  group_by(stratum, party) %>%
  summarise(Y = sum(votes),
            X = sum(total),
            .groups = 'drop'
  ) %>%
  left_join(strata_sample_2006, by = "stratum") %>% # unimos tabla de pesos
  group_by(party) %>%
  summarise(p_hat = 100 * sum(N / n * Y) / sum(N / n * X), .groups = 'drop')

sum(prop_x_party$p_hat)
```

```
## [1] 100
```

2. Utiliza **bootstrap** para calcular el error estándar, y reporta tu estimación del error.

- Genera 1000 muestras bootstrap.

```
muestra_x_est <- function(){
  p <- sample_2006 %>%
    select(polling_id, stratum, pri_pvem:total) %>%
    pivot_longer(names_to = "party", values_to = "votes",
                 cols = pri_pvem:otros) %>%
    group_by(stratum, party) %>%
    sample_frac(0.3, replace = TRUE) %>%
    summarise(Y = sum(votes),
              X = sum(total),
              .groups = 'drop'
            ) %>%
    left_join(strata_sample_2006, by = "stratum") %>% # unimos tabla de pesos
    group_by(party) %>%
    summarise(p_hat = 100 * sum(N / n * Y) / sum(N / n * X), .groups = 'drop')
}

p_estrato <- rerun(1000, muestra_x_est())
names_ <- c("p_otros", "p_pan", "p_panal", "p_prd_pt_conv", "p_pri_pvem", "p_psd")
p_df <- as.data.frame(matrix(unlist(p_estrato),
```

```

                                nrow = length(p_estrato),
                                byrow = TRUE))

p_df <- select(p_df, V7:V12)
colnames(p_df) <- names_
i <- sapply(p_df, is.factor)
p_df <- as.data.frame(apply(p_df, 2, as.numeric))
ee_boot <- apply(p_df, 2, FUN = sd)
ee_boot

```

```

##      p_otros      p_pan      p_panal p_prd_pt_conv  p_pri_pvem
## 0.06592384 0.24784266 0.02336096 0.20703211 0.18922768
##      p_psd
## 0.03218079

```

+ Recuerda que las muestras bootstrap tienen que tomar en cuenta la metodología que se utilizó en la selección de la muestra original, en este caso, lo que implica es que debes tomar una muestra aleatoria independiente dentro de cada estrato.

3. Construye un intervalo del 95% de confianza utilizando el método normal. Revisa si el supuesto de normalidad es razonable.

Reporta tus intervalos en una tabla.

```

h_otros <- ggplot(p_df, aes(x = p_otros)) +
  geom_histogram()

h_pan <- ggplot(p_df, aes(x = p_pan)) +
  geom_histogram()

h_panal <- ggplot(p_df, aes(x = p_panal)) +
  geom_histogram()

h_prd_pt_conv <- ggplot(p_df, aes(x = p_prd_pt_conv)) +
  geom_histogram()

h_pri_pvem <- ggplot(p_df, aes(x = p_pri_pvem)) +
  geom_histogram()

h_psd <- ggplot(p_df, aes(x = p_psd)) +
  geom_histogram()

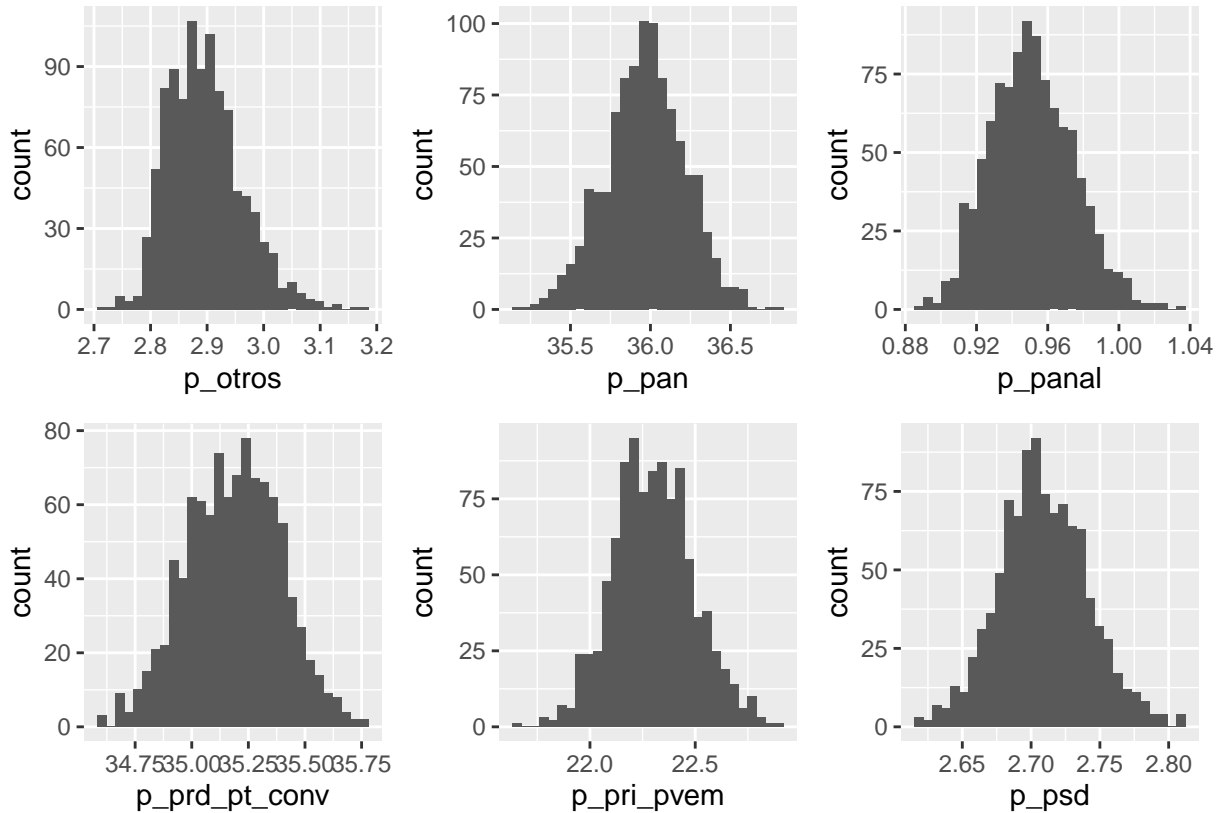
h_otros + h_pan + h_panal + h_prd_pt_conv + h_pri_pvem + h_psd

```

```

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

```



```
qq_otros <- ggplot(p_df, aes(sample = p_otros)) +
  geom_qq() +
  geom_qq_line(colour = "red")

qq_pan <- ggplot(p_df, aes(sample = p_pan)) +
  geom_qq() +
  geom_qq_line(colour = "red")

qq_panal <- ggplot(p_df, aes(sample = p_panal)) +
  geom_qq() +
  geom_qq_line(colour = "red")

qq_prd_pt_conv <- ggplot(p_df, aes(sample = p_prd_pt_conv)) +
  geom_qq() +
  geom_qq_line(colour = "red")

qq_pri_pvem <- ggplot(p_df, aes(sample = p_pri_pvem)) +
  geom_qq() +
  geom_qq_line(colour = "red")

qq_psd <- ggplot(p_df, aes(sample = p_psd)) +
  geom_qq() +
  geom_qq_line(colour = "red")

qq_otros + qq_pan + qq_panal + qq_prd_pt_conv + qq_pri_pvem + qq_psd
```

