

Tarea 07

```
library(tidyverse)
library(gridExtra)
library(estcomp)
```

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:

- se particionan las casillas de la población en estratos (cada casilla pertenece a exactamente un estrato), y
- 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:

```
strata_sample_2006
```

```
## # A tibble: 300 x 3
##   stratum      n      N
##   <dbl> <int> <int>
## 1       1     20    369
## 2       2     23    420
## 3       3     24    440
## 4       4     31    570
## 5       5     29    528
## 6       6     37    664
## 7       7     26    474
## 8       8     21    373
## 9       9     25    457
## 10      10     24    430
## # ... with 290 more rows
```

```
colSums(strata_sample_2006)
```

```
## stratum      n      N
##   45150    7200 130777
```

- 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`.

```
sample_2006
```

```
## # A tibble: 7,200 x 11
##   polling_id stratum edo_id rural pri_pvem  pan panal prd_pt_conv  psd otros
##   <int>    <dbl> <int> <dbl>    <int> <int> <int>    <int> <int> <int>
## 1     74593     106     16     1      47    40     0      40     0     9
## 2    109927     194     27     0     131    10     0     147     1     8
## 3    112039     199     28     0      51    74     2      57     2     2
## 4     86392     141     20     1     145    64     2     139     1    14
## 5    101306     176     24     0      51   160     0      64    14     1
## 6     86044     140     20     1     150    20     0     166     1    11
## 7     56057      57     15     1     117   119     2      82     0    24
## 8     84186     128     19     0     118   205     8      73     9    13
## 9     27778     283      9     0      26    65     5     249     7     2
## 10    29892     289      9     0      27    32     0     338    14     7
## # ... with 7,190 more rows, and 1 more variable: total <int>
```

```
## Verifying the number of votes taken from each poll-box
pollbox_cts <- sample_2006 %>%
  select(polling_id, stratum) %>%
  group_by(stratum) %>%
  summarise(count=n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
pollbox_cts
```

```
## # A tibble: 300 x 2
##   stratum count
##   <dbl> <int>
## 1      1     20
## 2      2     23
## 3      3     24
## 4      4     31
## 5      5     29
## 6      6     37
## 7      7     26
## 8      8     21
## 9      9     25
## 10     10     24
## # ... with 290 more rows
```

```
## Reviewing data from specific "stratums"
strat_ents <- sample_2006 %>%
  filter(stratum == 1)
strat_ents
```

```
## # A tibble: 20 x 11
##   polling_id stratum edo_id rural pri_pvem pan panal prd_pt_conv psd otros
##   <int> <dbl> <int> <dbl> <int> <int> <int> <int> <int> <int>
## 1      9      1      1      0      44    268      4      59      7     11
## 2    123      1      1      1     129    136      3      37      2     19
## 3    101      1      1      0      47    134      0      83      7      6
## 4     53      1      1      0      47    118      5      77      9      9
## 5    152      1      1      0     107    160      7      68     14      6
## 6    232      1      1      1     126    135      5      61      3      7
## 7    129      1      1      0     106    163      5      48     12     11
## 8    202      1      1      0     116    112     12     120     21      7
## 9    273      1      1      0      91    121      3      31      7      5
## 10     95      1      1      0      32    242      7      35     14     11
## 11    137      1      1      0     117    153      6      44      5     10
## 12     30      1      1      0      94    100      4      80      7     12
## 13    253      1      1      0      60    126     11      60     15     13
## 14    311      1      1      0     121    124     15      71      4      8
## 15     60      1      1      0      62    128      5      68      9      0
## 16    329      1      1      0      62    121      2      76      8      8
## 17    344      1      1      0      87    137      3      76      5     12
## 18    269      1      1      0      81    121     11      85     22      3
## 19     23      1      1      1      56    200      8      86      2     14
## 20    133      1      1      0      92    142      3      55     14      9
## # ... with 1 more variable: total <int>
```

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.

```
## Saving procedure in formula to calculate the reason estimate (estimador de razón) for
a sample
```

```
#### Procedure based on Rob's version of the code
```

```
calc_p_result <- function(sample, strata_sample){

  p_result <- sample %>%
    select(-edo_id, -rural) %>%
    group_by(stratum) %>%
    summarize( ## Adding all the votes per stratum per candidate
      no_casillas = n(),
      pri_pvem_vsum = sum(pri_pvem),
      pan_vsum = sum(pan),
      panal_vsum = sum(panal),
      prd_pt_conv_vsum = sum(prd_pt_conv),
      psd_vsum = sum(psd),
      otros_vsum = sum(otros),
      total_vsum = sum(total),
    ) %>%
    left_join(strata_sample, by="stratum") %>% ## Obtaining the total number of poll-box
es in a stratum
    mutate( ## Expanding the total number of votes
      pri_pvem_vexp = pri_pvem_vsum*(N/no_casillas),
      pan_vexp = pan_vsum*(N/no_casillas),
      panal_vexp = panal_vsum*(N/no_casillas),
      prd_pt_conv_vexp = prd_pt_conv_vsum*(N/no_casillas),
      psd_vexp = psd_vsum*(N/no_casillas),
      otros_vexp = otros_vsum*(N/no_casillas),
      total_vexp = total_vsum*(N/no_casillas),
    ) %>%
    select(!ends_with("vsum")) %>% ## Selecting relevant columns
    summarise_all(funs(sum)) %>% ## Adding results for all stratum
    select(ends_with("vexp")) %>% ## Selecting only results from candidates and total
    mutate( ## Obtaining the reason estimate (estimador de razón)
      p_pri_pvem = pri_pvem_vexp/total_vexp*100,
      p_pan = pan_vexp/total_vexp*100,
      p_panal = panal_vexp/total_vexp*100,
      p_prd_pt_conv = prd_pt_conv_vexp/total_vexp*100,
      p_psd = psd_vexp/total_vexp*100,
      p_otros = otros_vexp/total_vexp*100,
    ) %>%
    select(starts_with("p_"))

  p_result

}
```

```
## Trying function for given dataset
```

```
p_candidates <- calc_p_result(sample_2006, strata_sample_2006)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
p_candidates
```

```
## # A tibble: 1 x 6
##   p_pri_pvem p_pan p_panal p_prd_pt_conv p_psd p_otros
##   <dbl> <dbl>   <dbl>         <dbl> <dbl>   <dbl>
## 1      22.3  35.9    0.952         35.2  2.71    2.90
```

```
## Tere's sublime version of the code
```

```
# El siguiente código estima las proporciones para todos los partidos
# puedes utilizarlo o escribir tu propio código
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)
            ) %>%
  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))
```

```
## `summarise()` regrouping output by 'stratum' (override with `.groups` argument)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 6 x 2
##   party      p_hat
##   <chr>      <dbl>
## 1 otros      2.90
## 2 pan       35.9
## 3 panal     0.952
## 4 prd_pt_conv 35.2
## 5 pri_pvem  22.3
## 6 psd       2.71
```

```
## Function to take bootstrap samples from voting data per stratum
```

```
bootstrap_by_strat <- function(data){

  ## Initializing list where samples for each stratum will be stored
  s_samp_list = list()

  ## Loop through each stratum
  for (val in unique(data$stratum)){

    ## Taking sample of stratum
    s_samp <- data %>%
      filter(stratum == val) %>%
      sample_n(size = nrow(.), replace=TRUE)

    ## Appending sample of stratum to list
    s_samp_list[[val]] <- s_samp
  }

  ## Concatenating all the samples
  s_samp_res <- dplyr::bind_rows(s_samp_list)

  s_samp_res

}

## Evaluating sampling function
# bootstrap_by_strat(sample_2006)
```

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

- Genera 1000 muestras bootstrap.
- 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.

```
## Simulating analysis results with bootstrap
n_sims <- 1000
bootstrap <- map_df(1:n_sims, ~ calc_p_result(bootstrap_by_strat(sample_2006), strata_s
ample_2006))
```



```
## Plotting bootstrap results for each candidate and adding reason estimate (estimador d
e razón) as red line

## Creating list and counter to store created plots
plot_list = list()
i <- 1

## Loop to iterate over every candidate
for (col in names(bootstrap)){

  plot <- ggplot(
    bootstrap,
    aes_string(x = col)
  ) +
  geom_histogram() +
  geom_vline(
    xintercept = (select(p_candidates, col) %>% pull()),
    colour = "red"
  ) +
  labs(subtitle = col) +
  xlab("") + ylab("")

  plot_list[[i]] = plot
  i <- i + 1
}
```

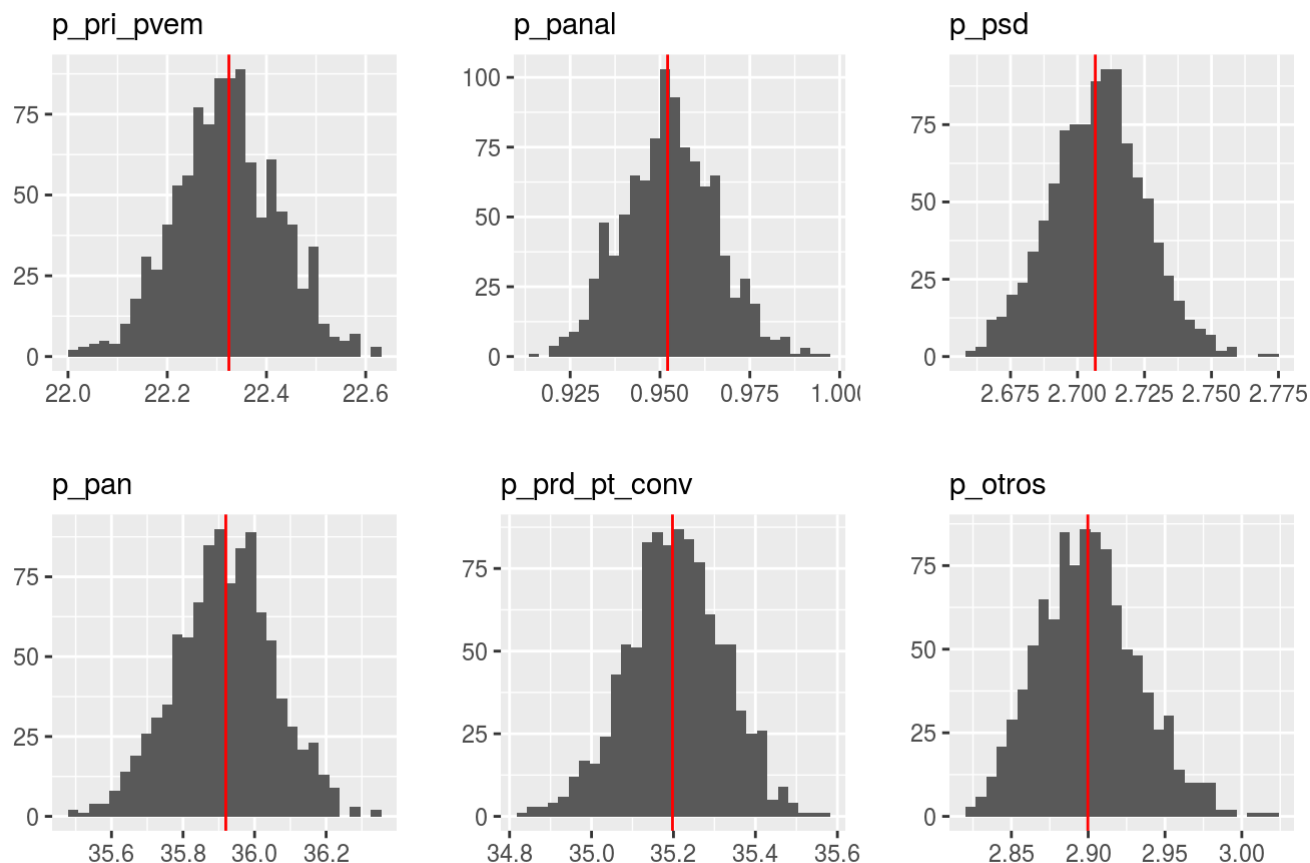
```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(col)` instead of `col` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
## Plotting results in single matrix
marrangeGrob(plot_list, nrow = 2, ncol = 3)
```

```
## `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`.
```

page 1 of 1



```
## Calculating standard error for every candidate
candidates_ee <- bootstrap %>%
  summarise_all(sd)
candidates_ee
```

```
## # A tibble: 1 x 6
##   p_pri_pvem p_pan p_panal p_prd_pt_conv p_psd p_otros
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1      0.104 0.138 0.0129      0.118 0.0177 0.0328
```

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

```
## Evaluating normality suposition
plot_list = list()
i <- 1

## Loop to iterate over every column
for (col in names(bootstrap)){

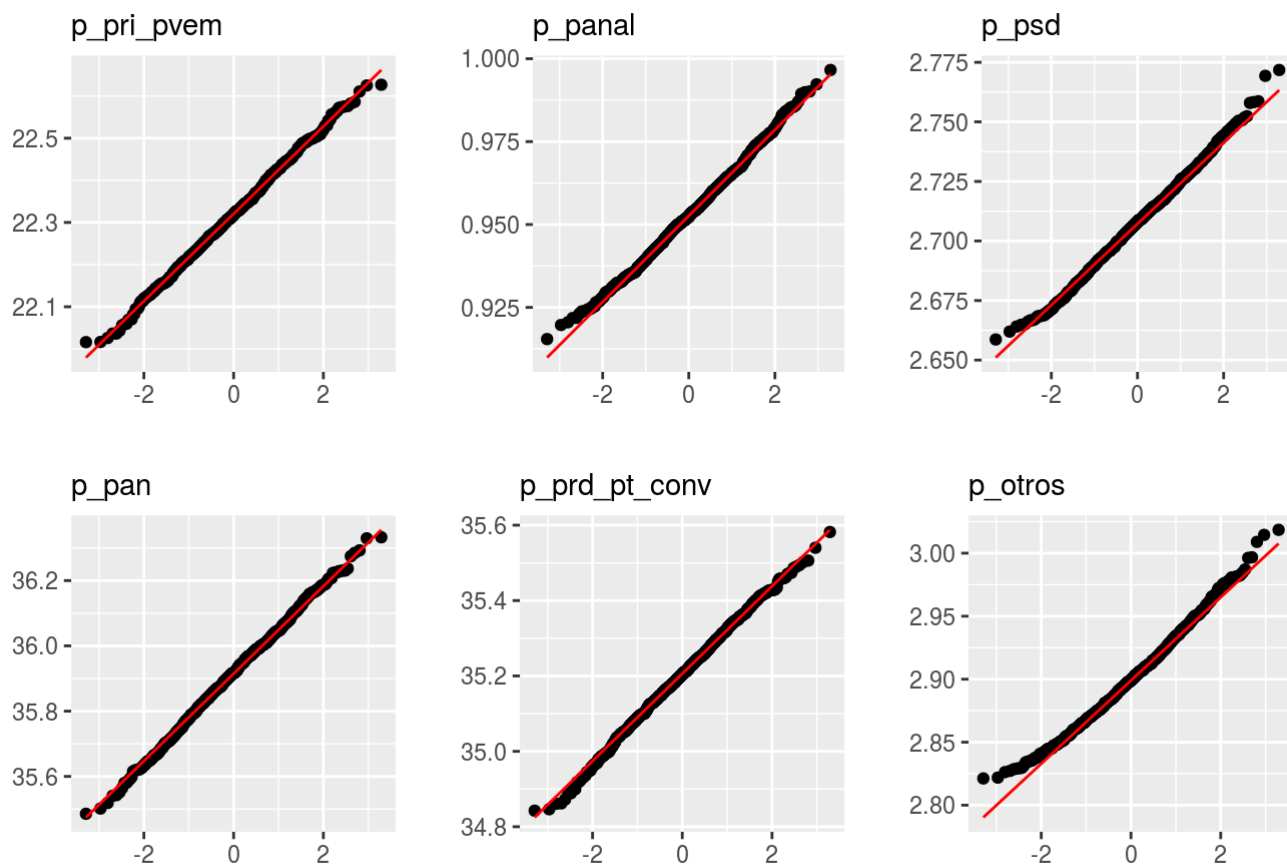
  ## Creating plot for selected column
  plot <- ggplot(
    bootstrap,
    aes_string(sample = col)
  ) +
    geom_qq() +
    geom_qq_line(colour = "red") +
    labs(subtitle = col) +
    xlab("") + ylab("")

  ## Storing plot in list
  plot_list[[i]] = plot
  i <- i + 1

}

## Plotting results in single matrix
marrangeGrob(plot_list, nrow = 2, ncol = 3)
```

page 1 of 1



Reporta tus intervalos en una tabla.

```
## Calculating bootstrap confidence intervals for each candidate (95% confidence)
trust_intervals <- names(p_candidates) %>%
  tibble() %>%
  mutate(
    ee = as.numeric(as.vector(candidates_ee[1, ])),
    lower_lim = as.numeric(as.vector(p_candidates[1, ]) - 2*ee),
    reason_estimate = as.numeric(as.vector(p_candidates[1, ])),
    upper_lim = as.numeric(as.vector(p_candidates[1, ]) + 2*ee)
  )

trust_intervals
```

```
## # A tibble: 6 x 5
##   .               ee lower_lim reason_estimate upper_lim
##   <chr>          <dbl>    <dbl>          <dbl>    <dbl>
## 1 p_pri_pvem    0.104      22.1           22.3      22.5
## 2 p_pan         0.138      35.6           35.9      36.2
## 3 p_panal       0.0129     0.926          0.952     0.978
## 4 p_prd_pt_conv 0.118      35.0           35.2      35.4
## 5 p_psd         0.0177     2.67           2.71      2.74
## 6 p_otros       0.0328     2.83           2.90      2.97
```