

TP 2 - Estratificacion Mesas Electorales Hogares

Gomez Vargas Andrea, Iummato Luciana, Pesce Andrea Gisele

2024-12-01

Tabla de contenidos

Paquetes de trabajo	1
Ejercicio 1	2
Ejercicio 2	33
Ejercicio 3	39
Ejercicio 4	41

Paquetes de trabajo

```
library(tidyverse)
library(survey)
library(readxl)
library(gt)
library(sampling)
library(VIM)
library(binom)
library(openxlsx)
library(DT)
library(stratification)
library(kableExtra)
options(scipen = 999)
```

Ejercicio 1

```
#Importar datos
```

```
marco <- read.csv("data/MESAS_ESCRUTADAS_Cierre.csv", encoding = "UTF-8")
```

```
#Explorar datos
```

```
names(marco)
```

```
[1] "Agrupacion"      "Cargo"           "Codigo"          "Distrito"
[5] "Establecimiento" "Fecha"           "IdCargo"          "IdCircuito"
[9] "IdDistrito"      "IdSeccion"       "Mesa"            "Seccion"
[13] "electores"       "envio"           "idAgrupacion"    "idAgrupacionInt"
[17] "tipoVoto"        "votos"
```

```
glimpse(marco)
```

```
Rows: 2,665,130
Columns: 18
$ Agrupacion      <chr> "", "", "", "", "", "JUNTOS POR EL CAMBIO", "FRENTE DE...
$ Cargo           <chr> "DIPUTADOS PROVINCIALES", "DIPUTADOS PROVINCIALES", "D...
$ Codigo          <int> 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 74, 74, 74, 74...
$ Distrito        <chr> "Ciudad Autónoma de Buenos Aires", "Ciudad Autónoma de...
$ Establecimiento <chr> "Colegio Nac. Nº2 D. F. Sarmiento", "Colegio Nac. Nº2 ...
$ Fecha           <chr> "14-11-2021 18:30", "14-11-2021 18:30", "14-11-2021 18...
$ IdCargo         <int> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, ...
$ IdCircuito      <chr> "00006", "00006", "00006", "00006", "00006", "00006", ...
$ IdDistrito      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ IdSeccion       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ Mesa            <chr> "09060E", "09060E", "09060E", "09060E", "09060E", "090...
$ Seccion         <chr> "Comuna 01", "Comuna 01", "Comuna 01", "Comuna 01", "C...
$ electores       <int> 346, 346, 346, 346, 346, 346, 346, 346, 346, 346, 346, 350,...
$ envio           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ idAgrupacion    <int> NA, NA, NA, NA, NA, 501, 502, 504, 503, 187, NA, NA, N...
$ idAgrupacionInt <int> NA, NA, NA, NA, NA, 11, 12, 14, 13, 7, NA, NA, NA, NA,...
$ tipoVoto        <chr> "blancos", "nulos", "recurridos", "comando", "impugnad...
$ votos           <int> 0, 1, 0, 0, 0, 9, 6, 3, 2, 0, 1, 0, 0, 0, 0, 29, 13, 1...
```

```
marco$Agrupacion<-as.factor(marco$Agrupacion)
```

1. Qué cantidad mínima de variables identifica una mesa electoral? Y un colegio?

- a. Mesa electoral: Distrito, sección, circuito, establecimiento y el propio código de la mesa
- b. Establecimiento: Distrito, sección, circuito, y el propio nombre del establecimiento

2. Hallar la proporción de votos a diputados por cada uno de los cuatro agrupamientos de partidos (son 3 segun el enunciado inicial del TP:FdT, JxC y FIT)

```
#Filtro votos positivos + votos diputados
marco_positivos<-marco %>% filter(tipoVoto=="positivo"& Cargo=="DIPUTADOS
NACIONALES")
```

```
# Elimino algunas variables
marco_positivos$Codigo <- NULL
marco_positivos$Fecha <- NULL
marco_positivos$Cargo <- NULL
marco_positivos$envio <- NULL
marco_positivos$idAgrupacion <- NULL
marco_positivos$idCargo <- NULL
marco_positivos$idAgrupacionInt <- NULL
```

```
total_votos <- marco_positivos %>%
  summarise(total_votos = sum(votos)) %>%
  pull(total_votos)
```

```
total_votos
```

```
[1] 23238621
```

```
#Recodificar partidos
marco_positivos$Partido <- "Resto"
marco_positivos$Partido <-
  ifelse(substr(marco_positivos$Agrupacion,1,15)=="FRENTE DE
TODOS", "FdT",marco_positivos$Partido)
marco_positivos$Partido <-
  ifelse(substr(marco_positivos$Agrupacion,1,19)=="
FRENTE DE IZQUIERDA", "FIT",marco_positivos$Partido)
marco_positivos$Partido <- ifelse(substr(marco_positivos$Agrupacion,1,6)=="
JUNTOS", "Juntos",marco_positivos$Partido)
```

```
#proporción de voto a diputados por partidos
proporciones<-marco_positivos %>%
  group_by(Partido) %>%
  summarise(votos_agrupacion= sum(votos),
            proporcion= round(votos_agrupacion/total_votos*100,1)
            )

proporciones
```

```
# A tibble: 4 × 3
  Partido votos_agrupacion proporcion
  <chr>          <int>          <dbl>
1 FIT             1194947             5.1
2 FdT             7339755            31.6
3 Juntos          7973281            34.3
4 Resto           6730638            29
```

3. Tabular la cantidad de colegios electorales, mesas electorales y electores por Estrato

6 estratos: CABA, Partidos del Gran Buenos Aires, Resto de Buenos Aires, Región Pampeana (Córdoba, Santa Fé, La Pampa, Entre Ríos), NEA - NOA, Resto

```
#creación variable estratos
table(marco_positivos$Distrito)
```

```
Buenos Aires
217470
Catamarca
5015
Chaco
14020
Chubut
6565
Ciudad Autónoma de Buenos Aires
36670
Córdoba
62013
Corrientes
10344
Entre Ríos
23373
Formosa
4224
Jujuy
```

```

5112
La Pampa
4400
La Rioja
4585
Mendoza
29344
Misiones
13755
Neuquén
11074
Río Negro
10038
Salta
21875
San Juan
6860
San Luis
6135
Santa Cruz
4370
Santa Fe
73152
Santiago del Estero
16359
Tierra del Fuego, Antártida e Islas del Atlántico Sur
3409
Tucumán
18770

```

```

GBA <- c("Avellaneda", "Almirante Brown", "Berazategui", "Esteban Echeverría",
        "Ezeiza", "Florencio Varela", "General San
Martín", "Hurlingham", "Ituzaingó",
        "José C. Paz", "La Matanza", "Lanús", "Lomas de Zamora",
        "Malvinas Argentinas", "Merlo", "Moreno", "Morón",
        "Quilmes", "San Fernando", "San Isidro", "San Miguel", "Tigre", "Tres de
Febrero", "Vicente López")

```

```

marco_positivos<- marco_positivos %>% mutate(estrato =case_when(
  Distrito== "Ciudad Autónoma de Buenos Aires" ~ "CABA",
  Distrito== "Córdoba" | Distrito== "Entre Ríos" | Distrito== "La Pampa" |
Distrito== "Santa Fe" ~ "Región_pampeana",
  Distrito== "Chaco" | Distrito== "Catamarca" | Distrito== "Corrientes" |
Distrito== "Formosa" | Distrito== "Santiago del Estero" |
  Distrito== "Jujuy" | Distrito== "Misiones" | Distrito== "La Rioja" | Distrito==
"Salta" | Distrito== "Tucumán"~ "NEA_NOA",
  Seccion %in% GBA & Distrito== "Buenos Aires"~ "GBA",

```

```
!Seccion %in% GBA & Distrito== "Buenos Aires" ~ "Resto_BsAs",
TRUE~"Resto_país"))

table(marco_positivos$Distrito,marco_positivos$estrato)
```

	CABA	GBA	NEA_NOA
Buenos Aires	0	129474	0
Catamarca	0	0	5015
Chaco	0	0	14020
Chubut	0	0	0
Ciudad Autónoma de Buenos Aires	36670	0	0
Córdoba	0	0	0
Corrientes	0	0	10344
Entre Ríos	0	0	0
Formosa	0	0	4224
Jujuy	0	0	5112
La Pampa	0	0	0
La Rioja	0	0	4585
Mendoza	0	0	0
Misiones	0	0	13755
Neuquén	0	0	0
Río Negro	0	0	0
Salta	0	0	21875
San Juan	0	0	0
San Luis	0	0	0
Santa Cruz	0	0	0
Santa Fe	0	0	0
Santiago del Estero	0	0	16359
Tierra del Fuego, Antártida e Islas del Atlántico Sur	0	0	0
Tucumán	0	0	18770

	Región_pampeana
Buenos Aires	0
Catamarca	0
Chaco	0
Chubut	0
Ciudad Autónoma de Buenos Aires	0
Córdoba	62013
Corrientes	0
Entre Ríos	23373
Formosa	0
Jujuy	0
La Pampa	4400
La Rioja	0
Mendoza	0
Misiones	0

Neuquén	0	
Río Negro	0	
Salta	0	
San Juan	0	
San Luis	0	
Santa Cruz	0	
Santa Fe	73152	
Santiago del Estero	0	
Tierra del Fuego, Antártida e Islas del Atlántico Sur	0	
Tucumán	0	
	Resto_BsAs	Resto_país
Buenos Aires	87996	0
Catamarca	0	0
Chaco	0	0
Chubut	0	6565
Ciudad Autónoma de Buenos Aires	0	0
Córdoba	0	0
Corrientes	0	0
Entre Ríos	0	0
Formosa	0	0
Jujuy	0	0
La Pampa	0	0
La Rioja	0	0
Mendoza	0	29344
Misiones	0	0
Neuquén	0	11074
Río Negro	0	10038
Salta	0	0
San Juan	0	6860
San Luis	0	6135
Santa Cruz	0	4370
Santa Fe	0	0
Santiago del Estero	0	0
Tierra del Fuego, Antártida e Islas del Atlántico Sur	0	3409
Tucumán	0	0

```
#Creación de tabla de estratos
tabla_estratos0 <-marco_positivos %>%
  group_by(estrato,Distrito,IdSeccion,IdCircuito, Establecimiento,Mesa) %>%
  summarise(electores = first(electores), .groups = "drop")

tabla_estratos1<-tabla_estratos0 %>%
  group_by(estrato,Distrito,IdSeccion,IdCircuito, Establecimiento) %>%
  summarise(mesas= n(),
            electores = sum(electores))
```

```

tabla_estratos_final <- tabla_estratos1 %>%
  group_by(estrato) %>%
  summarise(
    Establecimientos = n(),
    TotalMesas = sum(mesas),
    TotalElectores = sum(electores)
  )

tabla_estratos_final

```

```

# A tibble: 6 × 4
  estrato      Establecimientos TotalMesas TotalElectores
  <chr>          <int>         <int>         <int>
1 CABA              1021           7334       2535912
2 GBA                3270          21579       7547222
3 NEA_NOA           3832          22389       7503865
4 Región_pampeana  3757          21206       7137693
5 Resto_BsAs       2840          14666       4983390
6 Resto_país       2222          13063       4318698

```

4. Construir a partir del archivo dado una tabla de mesas electorales (lo necesitaremos más adelante), cada una con el total de votos a cada partido, el total de votos positivos y las variables de identificación.

```

tabla_votos_mesa <- marco_positivos %>%
  group_by(estrato, Distrito, IdSeccion, IdCircuito,
    Establecimiento, Mesa, Partido) %>%
  summarise(Voto = sum(votos, na.rm = TRUE)) %>%
  pivot_wider(names_from = Partido,
    values_from = Voto) %>%
  mutate(
    # Reemplaza NAs por 0 en las columnas de votos
    FdT = if_else(is.na(FdT), 0, FdT),
    Juntos = if_else(is.na(Juntos), 0, Juntos),
    FIT = if_else(is.na(FIT), 0, FIT),
    Resto = if_else(is.na(Resto), 0, Resto),
    # Calcula el total con los valores de votos
    Total = FdT + Juntos + FIT + Resto
  ) %>%
  ungroup()

head(tabla_votos_mesa) # mostramos los primeros 10 resultados

```



```
# A tibble: 6 × 11
  estrato Distrito IdSeccion IdCircuito Establecimiento Mesa FIT FdT Juntos
  <chr> <chr> <int> <chr> <chr> <chr> <dbl> <dbl> <dbl>
1 CABA Ciudad ... 1 00001 CENOF Centro I... 0002... 22 76 79
2 CABA Ciudad ... 1 00001 CENOF Centro I... 0002... 20 72 93
3 CABA Ciudad ... 1 00001 CENOF Centro I... 0002... 26 70 80
4 CABA Ciudad ... 1 00001 Esc. Integral ... 0001... 28 81 69
5 CABA Ciudad ... 1 00001 Esc. Integral ... 0002... 25 70 98
6 CABA Ciudad ... 1 00001 Esc. Integral ... 0002... 23 82 94
# i 2 more variables: Resto <dbl>, Total <dbl>
```

5. A partir de la tabla de mesas electorales construir la tabla de colegios electorales, cada uno con el total de votos a cada partido, el total de votos positivos y las variables de identificación.

```
tabla_votos_establecimiento <- marco_positivos %>%
  group_by(estrato, Distrito, IdSeccion, IdCircuito, Establecimiento, Partido) %>%
  summarise(Voto = sum(votos, na.rm = TRUE)) %>%
  pivot_wider(names_from = Partido,
              values_from = Voto) %>%
  mutate(
    # Reemplaza NAs por 0 en las columnas de votos
    FdT = if_else(is.na(FdT), 0, FdT),
    Juntos = if_else(is.na(Juntos), 0, Juntos),
    FIT = if_else(is.na(FIT), 0, FIT),
    Resto = if_else(is.na(Resto), 0, Resto),
    # Calcula el total con los valores de votos
    Total = FdT + Juntos + FIT + Resto
  ) %>%
  ungroup()

head(tabla_votos_establecimiento)
```

```
# A tibble: 6 × 10
  estrato Distrito IdSeccion IdCircuito Establecimiento FIT FdT Juntos Resto
  <chr> <chr> <int> <chr> <chr> <dbl> <dbl> <dbl> <dbl>
1 CABA Ciudad ... 1 00001 CENOF Centro I... 68 218 252 152
2 CABA Ciudad ... 1 00001 Esc. Integral ... 76 233 261 127
3 CABA Ciudad ... 1 00001 Esc. N°4 Baldo... 65 227 265 154
4 CABA Ciudad ... 1 00001 Esc. N°26 Hipó... 212 717 807 386
5 CABA Ciudad ... 1 00001 Esc. N°27 Dean... 80 229 274 153
6 CABA Ciudad ... 1 00001 Esc. N°3 Berna... 57 235 274 156
# i 1 more variable: Total <dbl>
```

6. Tabular y graficar los tres totales y los tres porcentajes poblacionales a estimar

```
#Tabla
totales <- tabla_votos_establecimiento %>%
  summarise(
    Total_FdT = sum(FdT, na.rm = TRUE),
    Total_Juntos = sum(Juntos, na.rm = TRUE),
    Total_FIT = sum(FIT, na.rm = TRUE),
    Total_Resto = sum(Resto, na.rm = TRUE),
    Total_votos = sum(Total, na.rm = TRUE),
    Porcentaje_FdT = round(Total_FdT/Total_votos*100,2),
    Porcentaje_Juntos = round(Total_Juntos/Total_votos*100,2),
    Porcentaje_FIT = round(Total_FIT/Total_votos*100,2),
    Porcentaje_Resto = round(Total_Resto/Total_votos*100,2)
  )

totales
```

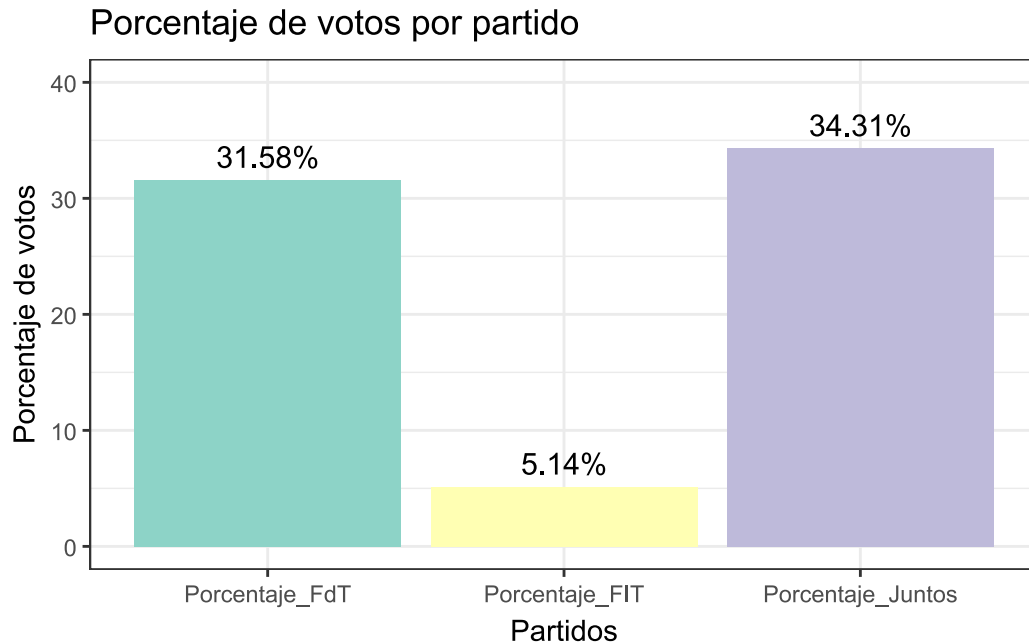
```
# A tibble: 1 × 9
  Total_FdT Total_Juntos Total_FIT Total_Resto Total_votos Porcentaje_FdT
    <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1  7339755    7973281    1194947    6730638    23238621    31.6
# i 3 more variables: Porcentaje_Juntos <dbl>, Porcentaje_FIT <dbl>,
#   Porcentaje_Resto <dbl>
```

```
#Gráfico
totales_long <- totales %>%
  select(Porcentaje_FdT, Porcentaje_FIT, Porcentaje_Juntos) %>%
  pivot_longer(cols = everything(), names_to = "Partido", values_to =
    "Porcentaje")

# Limpiar los nombres de los partidos
totales_long$Partido <- gsub("_Porcentaje", "", totales_long$Partido)

ggplot(totales_long, aes(x = Partido, y = Porcentaje, fill = Partido)) +
  geom_col() + # Gráfico de barras
  geom_text(aes(label = paste0(Porcentaje, "%")), # Agregar etiquetas con
    porcentajes
    vjust = -0.5, size = 4) + # Ajustar posición y tamaño
    del texto
  ylim(c(0,40)) +
  labs(
    title = "Porcentaje de votos por partido",
    x = "Partidos",
    y = "Porcentaje de votos"
  ) +
  theme_bw() +
```

```
scale_fill_brewer(palette = "Set3") + # Escala de colores agradable
theme(legend.position = "none")
```



7. Seleccionar con sampling una muestra de colegios con cada estrategia

Estrategia 1 con MAS

```
# Cantidad de estratos
H=6
#tamaño muestra establecimientos
n=400
#tamaño universo establecimientos
N=sum(tabla_estratos_final$Establecimientos)
```

```
#Creo variables para ponderar la muestra en mi marco
tabla_estratos_final<-tabla_estratos_final %>%
  mutate(pesos=Establecimientos/N,
         n_prop= round(n*TotalElectores/sum(TotalElectores)))
# Control
sum(tabla_estratos_final$n_prop)
```

```
[1] 401
```

```
#Agrego fpc a marco muestral (establecimientos por estrato?)
tabla_votos_establecimiento <- tabla_votos_establecimiento %>%
  group_by(estrato) %>%
  mutate(fpc=n())
```

```
#Ordeno el marco de muestreo por estrato (lo pide sampling)
tabla_votos_establecimiento <-
tabla_votos_establecimiento[order(tabla_votos_establecimiento$estrato),]
```

```
#Selección de muestra
smplMAS = sampling::strata(tabla_votos_establecimiento, stratanames =
c("estrato") ,
                           size=tabla_estratos_final$n_prop , description=TRUE,
                           method = "srswor")
```

Stratum 1

Population total and number of selected units: 1021 30

Stratum 2

Population total and number of selected units: 3270 89

Stratum 3

Population total and number of selected units: 3832 88

Stratum 4

Population total and number of selected units: 3757 84

Stratum 5

Population total and number of selected units: 2840 59

Stratum 6

Population total and number of selected units: 2222 51

Number of strata 6

Total number of selected units 401

```
# Recupero datos del marco de muestreo
muestra_casos <- sampling::getdata(tabla_votos_establecimiento,smplMAS)
```

```
# Calculo el factor de expansion utilizando probabilidad de seleccion calculada
muestra_casos$pondera <- 1/muestra_casos$Prob
```

8. Calcular con survey las estimaciones pedidas, junto a sus CV, IC(90%) y deff.

```
# Le indico a survey el diseno de muestra
# Indico fpc por estrato
diseno <- survey::svydesign(id=~1, strata = ~estrato,
                           weights = ~pondera, data=muestra_casos,
                           fpc = ~fpc)

diseno
```

```
Stratified Independent Sampling design
survey::svydesign(id = ~1, strata = ~estrato, weights = ~pondera,
  data = muestra_casos, fpc = ~fpc)
```

```
#Estimaciones de totales
# FdT
EstimTotalFdT <- survey::svytotal(~FdT, diseno, deff=TRUE, cv=TRUE )
EstimTotalFdT
```

	total	SE	DEff
FdT	6752484	235059	0.8216

```
cv(EstimTotalFdT)
```

	FdT
FdT	0.03481074

```
deff(EstimTotalFdT)
```

	FdT
	0.8215972

```
confint(EstimTotalFdT)
```

	2.5 %	97.5 %
FdT	6291777	7213192

```
# FIT
EstimTotalFIT <- survey::svytotal(~FIT, diseno, deff=TRUE, cv=TRUE )
EstimTotalFIT
```

	total	SE	DEff
FIT	1178818	65434	0.8871

```
cv(EstimTotalFIT)
```

	FIT
FIT	0.055508

```
deff(EstimTotalFIT)
```

	FIT
	0.8870591

```
confint(EstimTotalFdT)
```

	2.5 %	97.5 %
FdT	6291777	7213192

```
# Juntos
EstimTotalJuntos <- survey::svytotal(~Juntos, diseno, deff=TRUE, cv=TRUE )
EstimTotalJuntos
```

	total	SE	DEff
Juntos	7546343	291351	0.7226

```
cv(EstimTotalJuntos)
```

	Juntos
Juntos	0.03860818

```
deff(EstimTotalJuntos)
```

Juntos
0.7226383

```
confint(EstimTotalFdT)
```

2.5 % 97.5 %
FdT 6291777 7213192

```
#Dataframe de totales
#FdT
# Extraer el total y el error estándar
total <- coef(EstimTotalFdT) # Estimación del total
se <- SE(EstimTotalFdT)      # Error estándar

# Extraer coeficiente de variación y diseño efectivo
cv_val <- cv(EstimTotalFdT)  # Coeficiente de variación
deff_val <- deff(EstimTotalFdT) # Diseño efectivo

# Extraer el intervalo de confianza
conf <- confint(EstimTotalFdT) # Intervalo de confianza

# Crear un data.frame con los resultados
resultados_totalesFdT <- data.frame(
  Total = total,
  SE = se,
  CV = cv_val,
  Deff = deff_val,
  IC_Lower = conf[, 1], # Límite inferior del intervalo de confianza
  IC_Upper = conf[, 2]  # Límite superior del intervalo de confianza
)

#FIT
# Extraer el total y el error estándar
total <- coef(EstimTotalFIT) # Estimación del total
se <- SE(EstimTotalFIT)      # Error estándar

# Extraer coeficiente de variación y diseño efectivo
cv_val <- cv(EstimTotalFIT)  # Coeficiente de variación
deff_val <- deff(EstimTotalFIT) # Diseño efectivo

# Extraer el intervalo de confianza
conf <- confint(EstimTotalFIT) # Intervalo de confianza

# Crear un data.frame con los resultados
resultados_totalesFIT <- data.frame(
```

```

Total = total,
SE = se,
CV = cv_val,
Deff = deff_val,
IC_Lower = conf[, 1], # Límite inferior del intervalo de confianza
IC_Upper = conf[, 2]  # Límite superior del intervalo de confianza
)

#Juntos
# Extraer el total y el error estándar
total <- coef(EstimTotalJuntos) # Estimación del total
se <- SE(EstimTotalJuntos)      # Error estándar

# Extraer coeficiente de variación y diseño efectivo
cv_val <- cv(EstimTotalJuntos)  # Coeficiente de variación
deff_val <- deff(EstimTotalJuntos) # Diseño efectivo

# Extraer el intervalo de confianza
conf <- confint(EstimTotalJuntos) # Intervalo de confianza

# Crear un data.frame con los resultados
resultados_totalesJuntos <- data.frame(
  Total = total,
  SE = se,
  CV = cv_val,
  Deff = deff_val,
  IC_Lower = conf[, 1], # Límite inferior del intervalo de confianza
  IC_Upper = conf[, 2]
)

#Renombrar columnas
colnames(resultados_totalesFdT) <- c(
  "Total",
  "Error Estándar (SE)",
  "Coeficiente de Variación (CV)",
  "Diseño Efectivo (Deff)",
  "IC Inferior",
  "IC Superior"
)

# Ver los resultados
resultados_totalesFdT

```

	Total	Error Estándar (SE)	Coeficiente de Variación (CV)
FdT 6752484	235059	0.03481074	
	Diseño Efectivo (Deff)	IC Inferior	IC Superior
FdT	0.8215972	6291777	7213192


```
colnames(resultados_totalesFIT) <- c(
  "Total",
  "Error Estándar (SE)",
  "Coeficiente de Variación (CV)",
  "Diseño Efectivo (Deff)",
  "IC Inferior",
  "IC Superior"
)
# Ver los resultados
resultados_totalesFIT
```

	Total	Error Estándar (SE)	Coeficiente de Variación (CV)
FIT	1178818	65433.81	0.055508

	Diseño Efectivo (Deff)	IC Inferior	IC Superior
FIT	0.8870591	1050570	1307066

```
colnames(resultados_totalesJuntos) <- c(
  "Total",
  "Error Estándar (SE)",
  "Coeficiente de Variación (CV)",
  "Diseño Efectivo (Deff)",
  "IC Inferior",
  "IC Superior"
)
# Ver los resultados
resultados_totalesJuntos
```

	Total	Error Estándar (SE)	Coeficiente de Variación (CV)
Juntos	7546343	291350.6	0.03860818

	Diseño Efectivo (Deff)	IC Inferior	IC Superior
Juntos	0.7226383	6975306	8117379

```
#Estimaciones de proporciones
# FdT
Proporcion_FdT <- survey::svyratio(~FdT , ~Total, diseno, deff=TRUE )
Proporcion_FdT
```

```
Ratio estimator: svyratio.survey.design2(~FdT, ~Total, diseno, deff = TRUE)
Ratios=
      Total
FdT 0.3046485
SEs=
```

```
Total
FdT 0.008046654
```

```
df_ratio_FdT <- data.frame(Proporcion_FdT[[1]])
df_SE_FdT    <- data.frame( sqrt(Proporcion_FdT[[2]]))
CV_FdT       <- survey::cv(Proporcion_FdT)
df_IC_FdT    <-data.frame(confint(Proporcion_FdT))

df_proporcion_FdT <- cbind(df_ratio_FdT, df_SE_FdT,df_IC_FdT)

df_proporcion_FdT$CV <- 100*CV_FdT[1,1]
df_proporcion_FdT$deff <- survey::deff(Proporcion_FdT)
colnames(df_proporcion_FdT) <- c("Proporción", "Error Estándar (SE)", "IC
Inferior", "IC Superior", "Coeficiente de Variación (CV)", "Diseño Efectivo
(Deff)")

df_proporcion_FdT
```

```
Proporción Error Estándar (SE) IC Inferior IC Superior
FdT 0.3046485      0.008046654  0.2888774  0.3204197
Coeficiente de Variación (CV) Diseño Efectivo (Deff)
FdT                2.641291          0.8043235
```

```
# FIT
Proporcion_FIT <- survey :: svyratio(~FIT , ~Total, diseno, deff=TRUE )
Proporcion_FIT
```

```
Ratio estimator: svyratio.survey.design2(~FIT, ~Total, diseno, deff = TRUE)
Ratios=
      Total
FIT 0.05318414
SEs=
      Total
FIT 0.002530923
```

```
df_ratio_FIT <- data.frame(Proporcion_FIT[[1]])
df_SE_FIT    <- data.frame( sqrt(Proporcion_FIT[[2]]))
CV_FIT       <- survey::cv(Proporcion_FIT)
df_IC_FIT    <-data.frame(confint(Proporcion_FIT))

df_proporcion_FIT <- cbind(df_ratio_FIT, df_SE_FIT,df_IC_FIT)

df_proporcion_FIT$CV <- 100*CV_FIT[1,1]
df_proporcion_FIT$deff <- survey::deff(Proporcion_FIT)
```

```
colnames(df_proporcion_FIT) <- c("Proporción", "Error Estándar (SE)", "IC Inferior", "IC Superior", "Coeficiente de Variación (CV)", "Diseño Efectivo (Deff)")
```

```
df_proporcion_FIT
```

	Proporción	Error Estándar (SE)	IC Inferior	IC Superior
FIT	0.05318414	0.002530923	0.04822362	0.05814466
	Coeficiente de Variación (CV)	Diseño Efectivo (Deff)		
FIT	4.758792		0.8981481	

```
# Juntos
```

```
Proporcion_Juntos <- survey :: svyratio(~Juntos , ~Total, diseno, deff=TRUE )
Proporcion_Juntos
```

```
Ratio estimator: svyratio.survey.design2(~Juntos, ~Total, diseno, deff = TRUE)
Ratios=
```

	Total
Juntos	0.3404646

SEs=

	Total
Juntos	0.00946172

```
df_ratio_Juntos <- data.frame(Proporcion_Juntos[[1]])
df_SE_Juntos <- data.frame( sqrt(Proporcion_Juntos[[2]]))
CV_Juntos <- survey::cv(Proporcion_Juntos)
df_IC_Juntos <- data.frame(confint(Proporcion_Juntos))

df_proporcion_Juntos <- cbind(df_ratio_Juntos, df_SE_Juntos, df_IC_Juntos)

df_proporcion_Juntos$CV <- 100*CV_Juntos[1,1]
df_proporcion_Juntos$deff <- survey::deff(Proporcion_Juntos)
colnames(df_proporcion_Juntos) <- c("Proporción", "Error Estándar (SE)", "IC Inferior", "IC Superior", "Coeficiente de Variación (CV)", "Diseño Efectivo (Deff)")

df_proporcion_Juntos
```

	Proporción	Error Estándar (SE)	IC Inferior	IC Superior
Juntos	0.3404646	0.00946172	0.32192	0.3590093
	Coeficiente de Variación (CV)	Diseño Efectivo (Deff)		
Juntos	2.779061		0.6081707	

ESTRATEGIA 2: Madow en cada estrato, con probabilidad de selección proporcional a la cantidad de mesas electorales del colegio, ordenando los estratos por jurisdicción, Sección y IdCircuito.

```
#Datos para generar probabilidad de selección
N_mesas<-      tabla_votos_mesa      %>%      group_by(estrato,Distrito,
IdSeccion,IdCircuito,Establecimiento) %>%
  summarise(N_mesas = n())

tabla_votos_establecimiento<-tabla_votos_establecimiento %>%
  left_join(N_mesas, by = c("estrato", "Distrito","IdSeccion","IdCircuito",
"Establecimiento"))

tabla_votos_establecimiento<-merge(tabla_votos_establecimiento,
tabla_estratos_final[, c("estrato", "n_prop", "TotalMesas")],by="estrato")

# Genero probabilidad de seleccion de cada establecimiento
tabla_votos_establecimiento$pi_i <- tabla_votos_establecimiento$n_prop*
  tabla_votos_establecimiento$N_mesas/tabla_votos_establecimiento$TotalMesas

# Controlo
sum(tabla_votos_establecimiento$pi_i)
```

```
[1] 401
```

```
# Ordenamos el marco de muestreo

tabla_votos_establecimiento      <-
tabla_votos_establecimiento[order(tabla_votos_establecimiento$estrato,

tabla_votos_establecimiento$Distrito,

tabla_votos_establecimiento$IdSeccion,

tabla_votos_establecimiento$IdCircuito),]

# Selecciono la muestra
smpMadow = sampling::strata(tabla_votos_establecimiento, stratanames =
c("estrato") ,
                           size=tabla_estratos_final$n_prop , description=TRUE,
                           method = "systematic",
pik=tabla_votos_establecimiento$pi_i)
```

```
Stratum 1
```

```
Population total and number of selected units: 1021 30  
Stratum 2
```

```
Population total and number of selected units: 3270 89  
Stratum 3
```

```
Population total and number of selected units: 3832 88  
Stratum 4
```

```
Population total and number of selected units: 3757 84  
Stratum 5
```

```
Population total and number of selected units: 2840 59  
Stratum 6
```

```
Population total and number of selected units: 2222 51  
Number of strata 6  
Total number of selected units 401
```

```
# Recupero datos del marco de muestreo  
muestra_2 <- sampling :: getdata(tabla_votos_establecimiento, smplMadow)
```

```
# Calculo el factor de expansion utilizando probabilidad de seleccion calculada  
muestra_2$pondera <- 1/muestra_2$Prob
```

```
# Le indico a survey el diseno de muestra  
# pero supongo muestreo con reposicion (omito fpc)  
diseno2 <- survey :: svydesign(id=~1, strata=~estrato,  
                             weights=~pondera, data=muestra_2)  
diseno2
```

```
Stratified Independent Sampling design (with replacement)  
survey::svydesign(id = ~1, strata = ~estrato, weights = ~pondera,  
  data = muestra_2)
```

```
#Estimaciones de totales  
# FdT  
EstimTotalFdT2 <- survey :: svytotal(~FdT, diseno2, deff=TRUE, cv=TRUE )  
EstimTotalFdT2
```

```
      total      SE  DEff  
FdT 7345129 171971 0.4128
```

```
cv(EstimTotalFdT2)
```

```
      FdT  
FdT 0.02341289
```

```
deff(EstimTotalFdT2)
```

```
      FdT  
0.4127541
```

```
confint(EstimTotalFdT2)
```

```
      2.5 % 97.5 %  
FdT 7008072 7682185
```

```
# FIT  
EstimTotalFIT2 <- survey :: svytotal(~FIT, diseno2, deff=TRUE, cv=TRUE )  
EstimTotalFIT2
```

```
      total      SE  DEff  
FIT 1214217  43833 0.4973
```

```
cv(EstimTotalFIT2)
```

```
      FIT  
FIT 0.03609946
```

```
deff(EstimTotalFIT2)
```

```
      FIT  
0.497337
```

```
confint(EstimTotalFIT2)
```

```
      2.5 % 97.5 %  
FIT 1128307 1300127
```

```
# Juntos
EstimTotalJuntos2 <- survey :: svytotal(~Juntos, diseno2, deff=TRUE, cv=TRUE )
EstimTotalJuntos2
```

```
      total      SE  DEff
Juntos 8107366 198647 0.359
```

```
cv(EstimTotalJuntos2)
```

```
      Juntos
Juntos 0.02450203
```

```
deff(EstimTotalJuntos2)
```

```
      Juntos
0.3590009
```

```
confint(EstimTotalJuntos2)
```

```
      2.5 % 97.5 %
Juntos 7718025 8496707
```

```
#Dataframe de totales
#FdT
# Extraer el total y el error estándar
total <- coef(EstimTotalFdT2) # Estimación del total
se <- SE(EstimTotalFdT2)      # Error estándar

# Extraer coeficiente de variación y diseño efectivo
cv_val <- cv(EstimTotalFdT2)  # Coeficiente de variación
deff_val <- deff(EstimTotalFdT2) # Diseño efectivo

# Extraer el intervalo de confianza
conf <- confint(EstimTotalFdT2) # Intervalo de confianza

# Crear un data.frame con los resultados
resultados_totalesFdT2 <- data.frame(
  Total = total,
  SE = se,
  CV = cv_val,
  Deff = deff_val,
```

```

    IC_Lower = conf[, 1], # Límite inferior del intervalo de confianza
    IC_Upper = conf[, 2]  # Límite superior del intervalo de confianza
  )

#FIT
# Extraer el total y el error estándar
total <- coef(EstimTotalFIT2) # Estimación del total
se <- SE(EstimTotalFIT2)      # Error estándar

# Extraer coeficiente de variación y diseño efectivo
cv_val <- cv(EstimTotalFIT2)  # Coeficiente de variación
deff_val <- deff(EstimTotalFIT2) # Diseño efectivo

# Extraer el intervalo de confianza
conf <- confint(EstimTotalFIT2) # Intervalo de confianza

# Crear un data.frame con los resultados
resultados_totalesFIT2 <- data.frame(
  Total = total,
  SE = se,
  CV = cv_val,
  Deff = deff_val,
  IC_Lower = conf[, 1], # Límite inferior del intervalo de confianza
  IC_Upper = conf[, 2]  # Límite superior del intervalo de confianza
)

#Juntos
# Extraer el total y el error estándar
total <- coef(EstimTotalJuntos2) # Estimación del total
se <- SE(EstimTotalJuntos2)      # Error estándar

# Extraer coeficiente de variación y diseño efectivo
cv_val <- cv(EstimTotalJuntos2)  # Coeficiente de variación
deff_val <- deff(EstimTotalJuntos2) # Diseño efectivo

# Extraer el intervalo de confianza
conf <- confint(EstimTotalJuntos2) # Intervalo de confianza

# Crear un data.frame con los resultados
resultados_totalesJuntos2 <- data.frame(
  Total = total,
  SE = se,
  CV = cv_val,
  Deff = deff_val,
  IC_Lower = conf[, 1], # Límite inferior del intervalo de confianza
  IC_Upper = conf[, 2]  # Límite superior del intervalo de confianza
)

```



```
#Renombrar columnas

colnames(resultados_totalesFdT2) <- c(
  "Total",
  "Error Estándar (SE)",
  "Coeficiente de Variación (CV)",
  "Diseño Efectivo (Deff)",
  "IC Inferior",
  "IC Superior"
)
# Ver los resultados
print(resultados_totalesFdT2)
```

	Total	Error Estándar (SE)	Coeficiente de Variación (CV)
FdT	7345129	171970.7	0.02341289

	Diseño Efectivo (Deff)	IC Inferior	IC Superior
FdT	0.4127541	7008072	7682185

```
colnames(resultados_totalesFIT2) <- c(
  "Total",
  "Error Estándar (SE)",
  "Coeficiente de Variación (CV)",
  "Diseño Efectivo (Deff)",
  "IC Inferior",
  "IC Superior"
)
# Ver los resultados
print(resultados_totalesFIT2)
```

	Total	Error Estándar (SE)	Coeficiente de Variación (CV)
FIT	1214217	43832.57	0.03609946

	Diseño Efectivo (Deff)	IC Inferior	IC Superior
FIT	0.497337	1128307	1300127

```
colnames(resultados_totalesJuntos2) <- c(
  "Total",
  "Error Estándar (SE)",
  "Coeficiente de Variación (CV)",
  "Diseño Efectivo (Deff)",
  "IC Inferior",
  "IC Superior"
)
```

```
# Ver los resultados
kable(resultados_totalesJuntos2)
```

	To- tal	Error Están- dar (SE)	Coficiente de Varia- ción (CV)	Diseño Efectivo (Deff)	IC Infe- rior	IC Su- perior
Juntos	107366	198646.9	0.024502	0.3590009	7718025	8496707

```
#Estimaciones de proporciones
# FdT
Proporcion_FdT2 <- survey :: svyratio(~FdT , ~Total, diseno2, deff=TRUE )
Proporcion_FdT2
```

```
Ratio estimator: svyratio.survey.design2(~FdT, ~Total, diseno2, deff = TRUE)
Ratios=
      Total
FdT 0.3136641
SEs=
      Total
FdT 0.007246237
```

```
df_ratio_FdT2 <- data.frame(Proporcion_FdT2[[1]])
df_SE_FdT2 <- data.frame( sqrt(Proporcion_FdT2[[2]]))
CV_FdT2 <- survey::cv(Proporcion_FdT2)
df_IC_FdT2 <- data.frame(confint(Proporcion_FdT2))

df_proporcion_FdT2 <- cbind(df_ratio_FdT2, df_SE_FdT2, df_IC_FdT2)

df_proporcion_FdT2$CV_FdT2 <- 100*CV_FdT2[1,1]
df_proporcion_FdT2$deff <- survey::deff(Proporcion_FdT2)
colnames(df_proporcion_FdT2) <- c("Proporción", "Error Estándar (SE)", "IC Inferior", "IC Superior", "Coficiente de Variación (CV)", "Diseño Efectivo (Deff)")

print(df_proporcion_FdT2)
```

```
Proporción Error Estándar (SE) IC Inferior IC Superior
FdT 0.3136641 0.007246237 0.2994618 0.3278665
Coficiente de Variación (CV) Diseño Efectivo (Deff)
FdT 2.31019 0.6960076
```

```
# FIT
Proporcion_FIT2 <- survey :: svyratio(~FIT , ~Total, diseno2, deff=TRUE )
Proporcion_FIT2
```

```
Ratio estimator: svyratio.survey.design2(~FIT, ~Total, diseno2, deff = TRUE)
Ratios=
      Total
FIT 0.05185155
SEs=
      Total
FIT 0.001847191
```

```
df_ratio_FIT2 <- data.frame(Proporcion_FIT2[[1]])
df_SE_FIT2 <- data.frame( sqrt(Proporcion_FIT2[[2]]))
CV_FIT2 <- survey::cv(Proporcion_FIT2)
df_IC_FIT2 <- data.frame(confint(Proporcion_FIT2))

df_proporcion_FIT2 <- cbind(df_ratio_FIT2, df_SE_FIT2, df_IC_FIT2)

df_proporcion_FIT2$CV <- 100*CV_FIT2[1,1]
df_proporcion_FIT2$deff <- survey::deff(Proporcion_FIT2)
colnames(df_proporcion_FIT2) <- c("Proporción", "Error Estándar (SE)", "IC Inferior", "IC Superior", "Coeficiente de Variación (CV)", "Diseño Efectivo (Deff)")

print(df_proporcion_FIT2)
```

```
Proporción Error Estándar (SE) IC Inferior IC Superior
FIT 0.05185155      0.001847191 0.04823112 0.05547197
Coeficiente de Variación (CV) Diseño Efectivo (Deff)
FIT              3.56246              0.6707851
```

```
# Juntos
Proporcion_Juntos2 <- survey :: svyratio(~Juntos , ~Total, diseno2, deff=TRUE )
Proporcion_Juntos2
```

```
Ratio estimator: svyratio.survey.design2(~Juntos, ~Total, diseno2, deff = TRUE)
Ratios=
      Total
Juntos 0.3462145
SEs=
      Total
Juntos 0.008121637
```

```

df_ratio_Juntos2 <- data.frame(Proporcion_Juntos2[[1]])
df_SE_Juntos2    <- data.frame( sqrt(Proporcion_Juntos2[[2]]))
CV_Juntos2       <- survey::cv(Proporcion_Juntos2)
df_IC_Juntos2    <-data.frame(confint(Proporcion_Juntos2))

df_proporcion_Juntos2 <- cbind(df_ratio_Juntos2, df_SE_Juntos2,df_IC_Juntos2)

df_proporcion_Juntos2$CV <- 100*CV_Juntos2[1,1]
df_proporcion_Juntos2$deff <- survey::deff(Proporcion_Juntos2)
colnames(df_proporcion_Juntos2) <- c("Proporción", "Error Estándar (SE)", "IC Inferior", "IC Superior", "Coeficiente de Variación (CV)", "Diseño Efectivo (Deff)")

print(df_proporcion_Juntos2)

```

	Proporción	Error Estándar (SE)	IC Inferior	IC Superior
Juntos	0.3462145	0.008121637	0.3302964	0.3621326
	Coeficiente de Variación (CV)	Diseño Efectivo (Deff)		
Juntos	2.34584		0.5035463	

9. Presentar en un cuadro y gráfico los resultados

```

#Unifico df de estimación de totales 1ra estrategia
resultados_totales1 <- rbind(resultados_totalesFdT, resultados_totalesFIT,
resultados_totalesJuntos)

#Unifico df de estimación de totales 2da estrategia
resultados_totales2 <- rbind(resultados_totalesFdT2, resultados_totalesFIT2,
resultados_totalesJuntos)

#Unifico ambos
# Agregar una columna de identificación
resultados_totales1$estrategia <- "MAS"
resultados_totales2$estrategia <- "MADOW"
resultados_totales <- rbind(resultados_totales1, resultados_totales2)
library(tibble)
resultados_totales <- rownames_to_column(resultados_totales, var = "Partido")

#Unifico df de estimación de proporción 1ra estrategia
df_proporcion1 <- rbind(df_proporcion_FdT, df_proporcion_FIT,
df_proporcion_Juntos)

#Unifico df de estimación de proporción 2da estrategia
df_proporcion2 <- rbind(df_proporcion_FdT2, df_proporcion_FIT2,
df_proporcion_Juntos2)

```

```
#Unifico ambos
df_proporcion1$estrategia <- "MAS"
df_proporcion2$estrategia <- "MADOW"
df_proporcion <- rbind(df_proporcion1, df_proporcion2)
df_proporcion <- rownames_to_column(df_proporcion, var = "Partido")

print(resultados_totales)
```

	Partido	Total	Error Estándar (SE)	Coeficiente de Variación (CV)
1	FdT	6752484	235058.95	0.03481074
2	FIT	1178818	65433.81	0.05550800
3	Juntos	7546343	291350.55	0.03860818
4	FdT1	7345129	171970.67	0.02341289
5	FIT1	1214217	43832.57	0.03609946
6	Juntos1	7546343	291350.55	0.03860818

	Diseño Efectivo (Deff)	IC Inferior	IC Superior	estrategia
1	0.8215972	6291777	7213192	MAS
2	0.8870591	1050570	1307066	MAS
3	0.7226383	6975306	8117379	MAS
4	0.4127541	7008072	7682185	MADOW
5	0.4973370	1128307	1300127	MADOW
6	0.7226383	6975306	8117379	MADOW

```
print(df_proporcion)
```

	Partido	Proporción	Error Estándar (SE)	IC Inferior	IC Superior
1	FdT	0.30464853	0.008046654	0.28887738	0.32041968
2	FIT	0.05318414	0.002530923	0.04822362	0.05814466
3	Juntos	0.34046464	0.009461720	0.32192001	0.35900927
4	FdT1	0.31366415	0.007246237	0.29946178	0.32786651
5	FIT1	0.05185155	0.001847191	0.04823112	0.05547197
6	Juntos1	0.34621450	0.008121637	0.33029638	0.36213261

	Coeficiente de Variación (CV)	Diseño Efectivo (Deff)	estrategia
1	2.641291	0.8043235	MAS
2	4.758792	0.8981481	MAS
3	2.779061	0.6081707	MAS
4	2.310190	0.6960076	MADOW
5	3.562460	0.6707851	MADOW
6	2.345840	0.5035463	MADOW

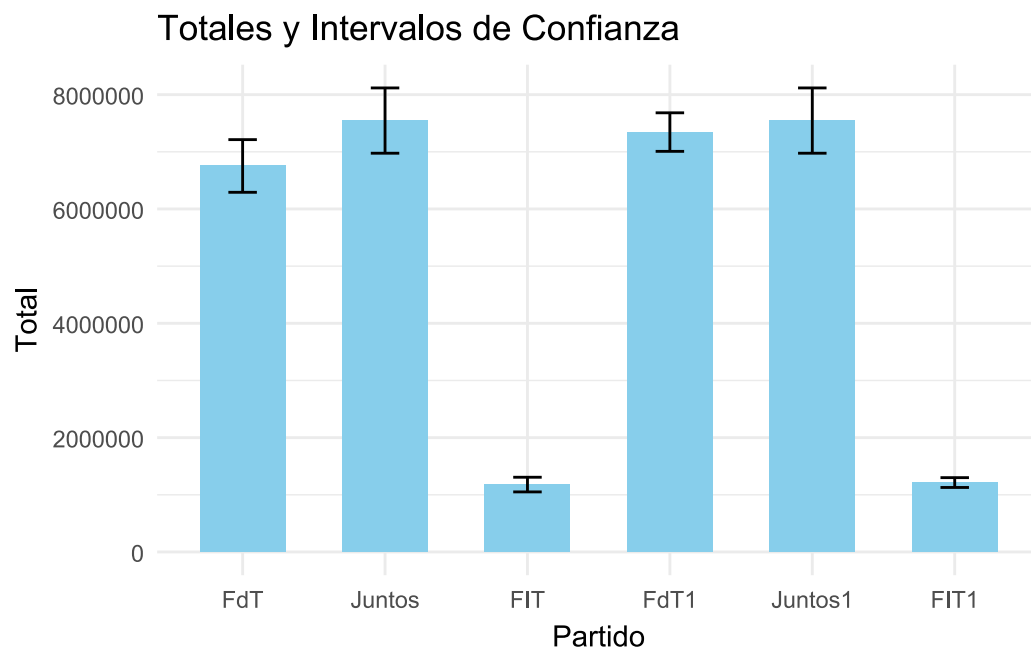
```
# Crear el gráfico
# Ordenar manualmente los partidos

ggplot(resultados_totales, aes(x = Partido, y = Total)) +
```

```

geom_bar(stat = "identity", fill = "skyblue", width = 0.6) + # Barras
geom_errorbar(aes(ymin = `IC Inferior`, ymax = `IC Superior`), width = 0.2)
+
scale_x_discrete(limits = c("FdT", "Juntos", "FIT", "FdT1", "Juntos1", "FIT1"))
+
labs(x = "Partido", y = "Total", title = "Totales y Intervalos de Confianza")
+
theme_minimal()

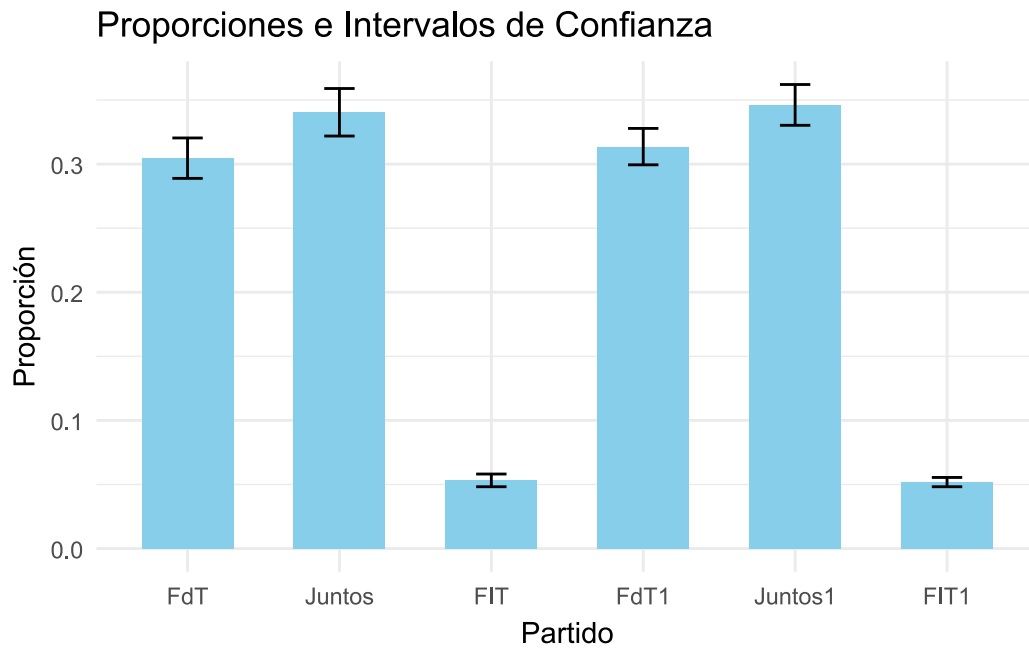
```



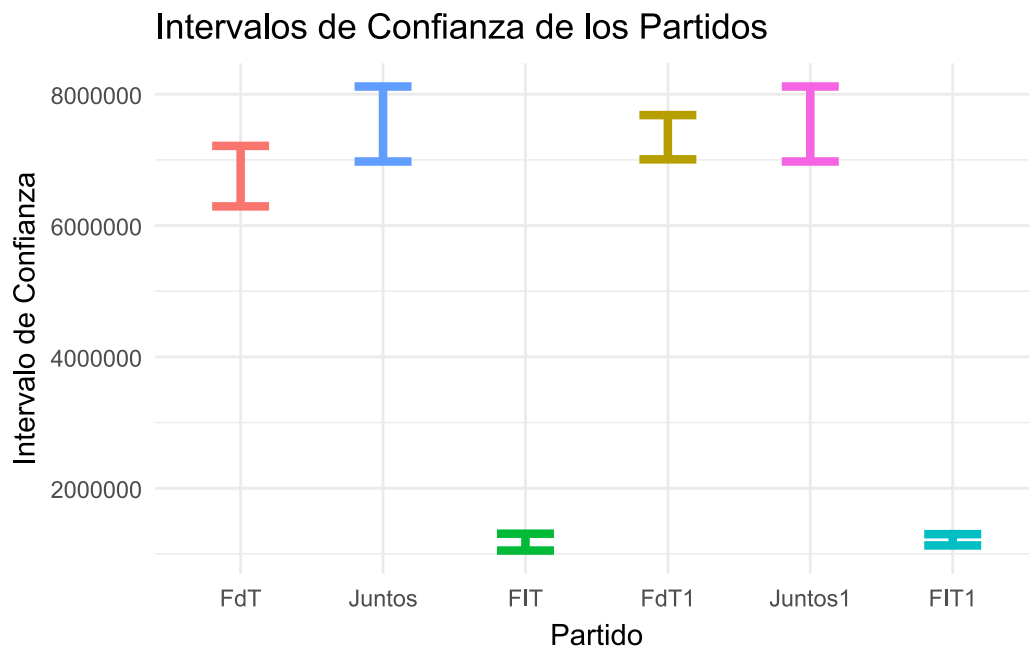
```

ggplot(df_proporcion, aes(x = Partido, y = Proporción)) +
geom_bar(stat = "identity", fill = "skyblue", width = 0.6) + # Barras
geom_errorbar(aes(ymin = `IC Inferior`, ymax = `IC Superior`), width = 0.2) +
scale_x_discrete(limits = c("FdT", "Juntos", "FIT", "FdT1", "Juntos1",
"FIT1"))+
labs(x = "Partido", y = "Proporción", title = "Proporciones e Intervalos de
Confianza") +
theme_minimal()

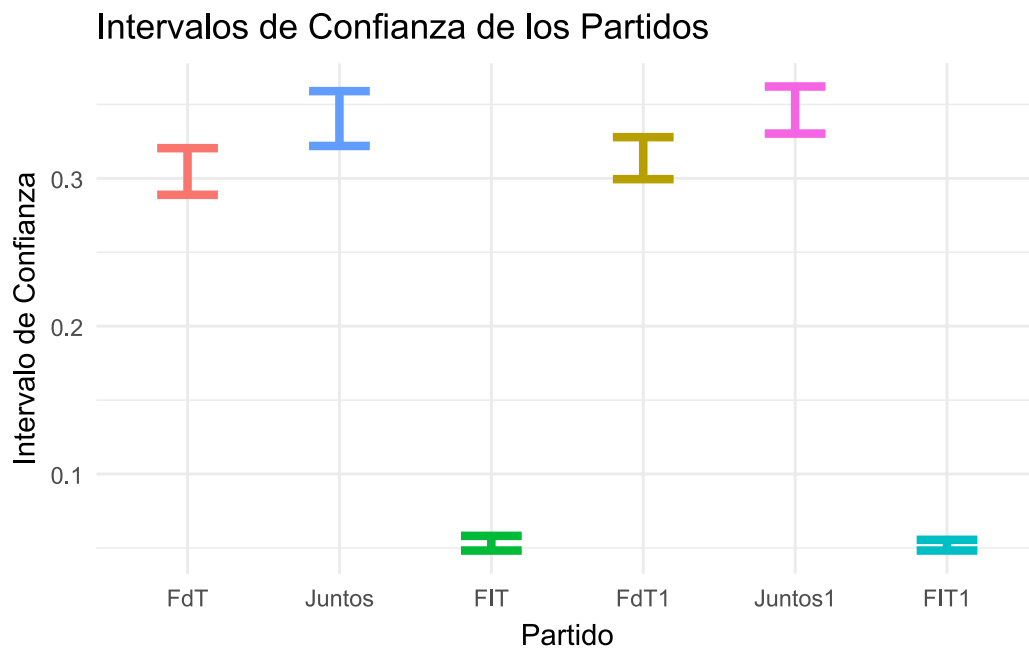
```



```
ggplot(resultados_totales, aes(x = Partido, ymin = `IC Inferior`, ymax = `IC Superior`, color = Partido)) +  
  geom_errorbar(linewidth = 1.5, width = 0.4) + # Líneas de error más gruesas  
  labs(x = "Partido", y = "Intervalo de Confianza", title = "Intervalos de Confianza de los Partidos") +  
  scale_x_discrete(limits = c("FdT", "Juntos", "FIT", "FdT1", "Juntos1", "FIT1")) +  
  theme_minimal() +  
  theme( #axis.text.x = element_text(angle = 45, hjust = 1),  
        legend.position = "none") # Rotar las etiquetas del eje X si es necesario
```



```
ggplot(df_proporcion, aes(x = Partido, ymin = `IC Inferior`, ymax = `IC Superior`,
  color = Partido)) +
  geom_errorbar(linewidth = 1.5, width = 0.4) + # Líneas de error más gruesas
  labs(x = "Partido", y = "Intervalo de Confianza", title = "Intervalos de
  Confianza de los Partidos") +
  scale_x_discrete(limits = c("FdT", "Juntos", "FIT", "FdT1", "Juntos1", "FIT1"))
+
  theme_minimal() +
  theme( #axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "none") # Rotar las etiquetas del eje X si es necesario
```

10. Con alguna de las dos estrategias se puede determinar con un 95% de confianza quien sacó más votos?

Comparación de intervalos:

En la primera estrategia si bien las proporciones que sacaron los candidatos son diferentes los intervalos de confianza se superponen, es decir que el límite superior del frente de todos está dentro del intervalo de Juntos por el cambio, por lo que no puede afirmarse que un partido sacó más votos que el otro. Por el contrario con la segunda estrategia de muestreo los IC no se solapan, el límite inferior de Juntos es mayor que el límite superior del frente de todos, de forma que podría afirmarse con un 95% de confianza que Juntos por el cambio sacó más votos en la elección.

De igual manera el valor del DEFF en la segunda estrategia es menor demostrando que es una mejor opción para la selección de los casos al reducir la varianza de la estimación.

Ejercicio 2

```
#Creación de dataframe
zona<-c("A", "B", "C")
hogares_marco<-c(25000, 65000, 20000)
hogares_encuestados<-c(200, 150, 250)
hogares_pobres_muestra<-c(70, 80, 22)
poblacion_pobre_encuestada<-c(260, 400, 60)
poblacion_encuestada<-c(820, 700, 600)

tabla_datos<-as.data.frame(cbind(zona, hogares_marco, hogares_encuestados, hogares_pobres_muestra, poblacion_pobre_encuestada, poblacion_encuestada))
```

```
#str(tabla_datos)

tabla_datos <- tabla_datos %>%
  mutate(across(-1, as.numeric))

print(tabla_datos)
```

	zona	hogares_marco	hogares_encuestados	hogares_pobres_muestra
1	A	25000	200	70
2	B	65000	150	80
3	C	20000	250	22

	poblacion_pobre_encuestada	poblacion_encuestada
1	260	820
2	400	700
3	60	600

1. Presentar en una tabla o gráfico la proporción de hogares pobres por zona

```
tabla_datos <- tabla_datos %>%
  mutate(ph_hogares=hogares_pobres_muestra/hogares_encuestados,
         ph_poblacion=poblacion_pobre_encuestada/poblacion_encuestada)

print(tabla_datos[, c("zona", "ph_hogares", "ph_poblacion")])
```

	zona	ph_hogares	ph_poblacion
1	A	0.3500000	0.3170732
2	B	0.5333333	0.5714286
3	C	0.0880000	0.1000000

2. Hallar el factor de expansión de cada hogar de la muestra y de cada persona encuestada

```
tabla_datos <- tabla_datos %>%
  mutate(F_hogar=hogares_marco/hogares_encuestados,
         F_persona=hogares_marco/hogares_encuestados)
```

Como todas las personas del hogar son encuestadas, la probabilidad de selección de una persona es igual a la del hogar y por tanto el factor de expansión, que es la inversa de la probabilidad de selección, también lo será.

3. Las tres zonas presentan un perfil diferencial en términos de pobreza?

Si se observan las estimaciones muestrales puede observarse que la zona B presenta una proporción de hogares pobres mucho mayor que los demás estratos, lo mismo sucede con la proporción

de las personas pobres. Sin embargo, para afirmar que existen diferencias entre los estratos hay que analizar los intervalos de confianza para cada estimación, si los mismos no se solapan puede afirmarse, con el nivel de confianza definido, que las estimaciones de pobreza difieren entre los estratos, pudiendo determinarse el estrato con mayor y menor pobreza. Dicho procedimiento se realiza en el punto n°4.

4. Estimar, con la muestra seleccionada, el total de hogares pobres y la proporción de hogares pobres, el CV y deff correspondientes. Dar un IC(90%) para cada estimación.

```
#Estimación

n=sum(tabla_datos$hogares_encuestados) # en la consigna dice 700, no son 600 hogares?
N=sum(tabla_datos$hogares_marco)

#peso en cada estrato

tabla_datos<-tabla_datos %>%
  mutate(wh=hogares_marco/N)

#peso al cuadrado en cada estrato

tabla_datos<-tabla_datos %>%
  mutate(wh2=(hogares_marco/N)^2)

#s2 en cada estrato
tabla_datos <- tabla_datos %>%
  mutate(sh2= ph_hogares * (1 - ph_hogares))

#varianza en estratos
tabla_datos <- tabla_datos %>%
  mutate(varianza= wh2*sh2/hogares_encuestados)

#varianza del estimador
varianza_estimador=sum(tabla_datos$varianza) * sum(tabla_datos$hogares_marco)^2

#desvío
ds=sqrt(varianza_estimador)

#probabilidad en estrato
tabla_datos <- tabla_datos %>%
  mutate(ph_wh2= ph_hogares * wh)

#estimación del total de hogares pobres
estimacion_total_hogares <- sum(tabla_datos$ph_wh2 * N)
```

```
print(paste("Estimación total hogares pobres: ",  
round(estimacion_total_hogares)))
```

```
[1] "Estimación total hogares pobres: 45177"
```

```
#Intervalos de confianza 90%  
IC_infT <- estimacion_total_hogares - 1.64 * ds  
IC_supT <- estimacion_total_hogares + 1.64 * ds  
print(paste("IC Inferior Total hogares pobres: ", IC_infT))
```

```
[1] "IC Inferior Total hogares pobres: 40581.8217173449"
```

```
print(paste("IC Superior Total hogares pobres: ", IC_supT))
```

```
[1] "IC Superior Total hogares pobres: 49771.5116159885"
```

```
#estimación de la proporción de hogares pobres  
estimacion_proporcion_hogares <- sum(tabla_datos$ph_wh2 * 100)  
print(paste("Estimación proporción hogares pobres: ",  
estimacion_proporcion_hogares))
```

```
[1] "Estimación proporción hogares pobres: 41.069696969697"
```

```
IC_infP <- estimacion_proporcion_hogares/100 - 1.64 * ds/sum(hogares_marco)  
IC_supP <- estimacion_proporcion_hogares/100 + 1.64 * ds/sum(hogares_marco)  
print(paste("IC Inferior Proporción hogares pobres: ", IC_infP * 100))
```

```
[1] "IC Inferior Proporción hogares pobres: 36.8925651975862"
```

```
print(paste("IC Superior Proporción hogares pobres: ", IC_supP * 100))
```

```
[1] "IC Superior Proporción hogares pobres: 45.2468287418077"
```

```
#CV  
CV_total <- ds / estimacion_total_hogares * 100  
print(paste("CV del total: ", CV_total))
```

```
[1] "CV del total: 6.20172963593941"
```

```
CV_prop <- (ds/sum(hogares_marco)) / sum(tabla_datos$ph_wh2) * 100  
print(paste("CV de la proporción: ", CV_prop))
```

```
[1] "CV de la proporción: 6.20172963593941"
```

```
#varianza en MAS (1-n/N)*s2/n (NO ME QUEDA CLARO SI DIVIDO POR n)  
  
p_total_hogares=sum(tabla_datos$hogares_pobres_muestra)/  
sum(tabla_datos$hogares_encuestados)  
s2_total= p_total_hogares * (1 - p_total_hogares)  
var_MAS=(1-n/N)*s2_total/n  
  
#DEFF  
deff= (ds/sum(hogares_marco))^2/var_MAS  
print(paste("DEFF: ", deff))
```

```
[1] "DEFF: 1.91392781399597"
```

5. Por qué, siendo que las tres zonas son diferentes respecto a la variable bajo estudio (proporción de hogares pobres), el deff es claramente mayor a 1 en la estimación del total y proporción de hogares pobres?

Porque la asignación de la cantidad de hogares seleccionado por estrato no se hizo en función de un criterio adecuado, no fue uniforme, ni ponderada ni óptima. De hecho en el estrato donde hay mas hogares en el universo se seleccionan menos hogares en la muestra y viceversa.

6. Estimar el total de personas pobres y la proporción de personas pobres (recordar que población pobre es la que habita en hogares pobres). Se puede con los datos disponibles estimar el CV y deff de estas estimaciones?

```
#probabilidad en estrato  
tabla_datos <- tabla_datos %>%  
  mutate(ph_poblacion_wh= ph_poblacion * wh)  
  
#estimación del total de hogares pobres  
estimacion_total_pobacion <- sum(tabla_datos$ph_poblacion_wh * N)  
print(paste("Estimación total población pobre: ", estimacion_total_pobacion))
```

```
[1] "Estimación total población pobre: 47069.6864111498"
```

```
#Intervalos de confianza 90%
IC_infT <- estimacion_total_pobacion - 1.64 * ds
IC_supT <- estimacion_total_pobacion + 1.64 * ds
print(paste("IC Inferior Total hogares pobres: ", IC_infT))
```

```
[1] "IC Inferior Total hogares pobres: 42474.841461828"
```

```
print(paste("IC Superior Total hogares pobres: ", IC_supT))
```

```
[1] "IC Superior Total hogares pobres: 51664.5313604716"
```

```
#estimación de la proporción de hogares pobres
estimacion_proporcion_poblacion <- sum(tabla_datos$ph_poblacion_wh * 100)
print(paste("Estimación proporción población pobre: ",
estimacion_proporcion_poblacion))
```

```
[1] "Estimación proporción población pobre: 42.7906240101362"
```

```
IC_infP <- estimacion_proporcion_poblacion - 1.64 * ds
IC_supP <- estimacion_proporcion_poblacion + 1.64 * ds
print(paste("IC Inferior proporción población pobre: ", IC_infP))
```

```
[1] "IC Inferior proporción población pobre: -4552.05432531167"
```

```
print(paste("IC Superior proporción población pobre: ", IC_supP))
```

```
[1] "IC Superior proporción población pobre: 4637.63557333195"
```

No se puede estimar el cv y el DEFF ya que los datos disponibles en el marco muestral refieren a la distribución de hogares por estrato, no conocemos la distribución real de la población pobre entre los estratos para calcular el peso de la población pobre (wh) en el marco muestral, y de esa manera no podemos calcular la varianza del estimador y con ello el CV y el deff que toman este insumo para su cálculo

7. En base a los resultados de la encuesta, cómo debería haber sido la distribución de la muestra por zona si el objetivo era estimar el total de hogares pobres en la localidad? (obtener la asignación de Neyman)

```
#creación de Sh

tabla_datos <- tabla_datos %>%
  mutate(Sh= sqrt(sh2))

#creación de Nh*Sh
tabla_datos <- tabla_datos %>%
  mutate(Nh_Sh= hogares_marco*Sh)

#creación de hogares seleccionados
tabla_datos <- tabla_datos %>%
  mutate(nh_nuevo= round(n*Nh_Sh/sum(Nh_Sh)))

print(tabla_datos[, c("zona", "hogares_marco", "nh_nuevo")])
```

	zona	hogares_marco	nh_nuevo
1	A	25000	143
2	B	65000	389
3	C	20000	68

Ejercicio 3

Seleccionar con sampling una muestra ese diseño y estimar luego con survey la media de Y y el correspondiente CV

```
data <- data.frame(
  Estrato = c(1, 2, 2, 3, 3, 3, 3, 3), # Estratos
  Unidad = 1:8, # Unidades
  Y = c(18, 9, 10, 5, 6, 2, 4, 6) # Valores de la variable Y
)
```

Paso 1: Agrego fpc pues selecciono MAS en cada estrato

```
data <- data %>%
  group_by(Estrato) %>%
  mutate(fpc=n())
```

Paso 2: Selección de muestra

```
# Tamaños de muestra por estrato
n <- c(1, 1, 3) # Estrato 1: 1 unidad, Estrato 2: 1 unidad, Estrato 3: 3 unidades

# Selección de la muestra estratificada
set.seed(123) # Para reproducibilidad
```

```
#options(survey.lonely.psu="fail")

muestra <- sampling::strata(data = data, stratanames = "Estrato", size = n,
method = "srswor") # Sin reemplazo

# Extraer las unidades seleccionadas
muestra_seleccionada <- getdata(data, muestra)
```

Paso 3: Diseño de encuesta

La opción **adjust** utiliza la media total para el cálculo de varianza en el estrato con solo una observación

```
options(survey.lonely.psu="adjust")

# Crear un diseño estratificado con los pesos correctos

design <- svydesign(
  ids = ~1, # Las unidades seleccionadas no tienen subgrupos
  strata = ~Estrato, # Estratos
  data = muestra_seleccionada,
  weights = ~1 / Prob, # Pesos: inverso de la probabilidad de selección
  fpc = ~ fpc
)
```

Paso 4: Cálculo de estimaciones

```
# Estimar la media de Y
media_Y <- svymean(~Y, design)
print("Media estimada de Y:")
```

```
[1] "Media estimada de Y:"
```

```
print(media_Y)
```

```
      mean      SE
Y 9.3636 0.4616
```

```
# Calcular el coeficiente de variación (CV) de la media
cv_Y <- cv(media_Y)
print("Coeficiente de variación (CV) de la media:")
```

```
[1] "Coeficiente de variación (CV) de la media:"
```



```
print(cv_Y)
```

```
      Y  
Y 0.04929865
```

Ejercicio 4

```
Deptos2022 <- read_excel("data/tabla_deptos_2022.xlsx")
```