Trabajo final Muestreo II

Fiorella Lúngaro, Emanuelle Marsella y Maximiliano Saldaña

Diciembre 2021

Parte 1

Se calculan las estimaciones puntuales de la tasa de desempleo, la proporción de personas pobres y del ingreso promedio, haciendo uso de los ponderadores originales w_0, es decir, sin ajustar por no respuesta. Esta estrategia de cómputo resulta correcta si el esquema de no respuesta que se considera es *Missing Completely at Random* (MCAR), bajo el cual la probabilidad de responder no depende de las variables de interés ni auxiliares y todas las unidad del marco tienen la misma probabilidad de responder (Ferreira y Zoppolo, 2017).

```
# Diseño usando los ponderadores originales, MCAR.
design1 <- muestra %>%
  filter(R==1) %>%
  as_survey_design(ids = id_hogar, strata = estrato, weights = w0)
```

```
## Tasa de desempleo (desempleados/activos)

#Se piensa como un problema de estimación en dominios
#Nos interesan los desempleados considerando el grupo de los activos.
design1 %>%
  filter(activo == 1) %>%
  group_by(desocupado) %>%
  summarise(tasa_desempleo = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   desocupado tasa_desempleo tasa_desempleo_se tasa_desempleo_cv tasa_desempleo_~
##   <fct>               <dbl>             <dbl>             <dbl>            <dbl>
## 1 0                 0.918           0.00330           0.00360             1.07
## 2 1                 0.0824          0.00330           0.0400              1.07
```

La estimación puntual de la proporción de desempleados es 0,0824 (1 representa a los desocupados y 0 a los ocupados); mientras que el error estándar (la medida que empleamos para medir la variación del estimador entre muestra y muestra) es 0,0033. Otra medida de la calidad de un estimador $\hat{\theta}$ es su coeficiente de variación, que mide su dispersión relativa. Se define como (Ferreira y Zoppollo, 2017):

$$CV(\hat{\theta}) = \frac{\sqrt{\hat{V}(\hat{\theta})}}{\left|E(\hat{\theta})\right|}$$

Y en el caso del estimador de la proporción de desempleados su estimación es 0,04.

El efecto diseño es una medida que permite comparar la eficiencia en términos de variabilidad del estimador para el diseño utilizado, respecto al diseño aleatorio simple sin reposición que. Siendo $p(s)$ el diseño medible considerado, se define como:

$$Deff(p(s), \hat{\theta}) = \frac{V_{p(s)}(\hat{\theta})}{V_{SI}(\hat{\theta})}$$

En el caso del estimador de la proporción de desempleados su valor es 1,07; lo que indica que en este caso el diseño SI es un 7 % más eficiente que el empleado.

```
## Proporción de personas pobres
design1 %>%
  group_by(pobreza) %>%
  summarise(prop_pobres = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   pobreza prop_pobres prop_pobres_se prop_pobres_cv prop_pobres_deff
##   <fct>         <dbl>          <dbl>          <dbl>            <dbl>
## 1 0             0.919        0.00377        0.00410             2.84
## 2 1             0.0811       0.00377        0.0465              2.84
```

En cuanto a la proporción de personas pobres, la estimación puntual es de 0,0811 (1 representa a las personas pobres y 0 los que no lo son). El error estándar se estima que es 0,004 aproximadamente, mientras que el coeficiente de variación se estima que es 0,05 aproximadamente. La estimación del efecto diseño es 2,84; un elevado valor que indica que el diseño empleado es altamente ineficiente en comparación con el SI, en particular casi tres veces más.

```
## Ingreso promedio
design1 %>%
  summarise(ingreso_prom = survey_mean(ingreso, deff = TRUE, vartype = c('se','cv')))
```

```
##   ingreso_prom ingreso_prom_se ingreso_prom_cv ingreso_prom_deff
## 1     21798.64        239.5809      0.01099063         0.9354982
```

La estimación puntual del ingreso promedio es 21799, siendo la estimación de su error estándar 240. Por otro lado, el coeficiente de variación toma el valor 0,011. La estimación del efecto diseño es 0,94 aproximadamente, por lo que en este caso el diseño empelado resulta más eficiente que el SI, un 6 % más.

```
muestra %>%
  summarise(
    # tasa de no respuesta no ponderada
    nr_np = 1 - mean(R),
    # tasa de no respuesta ponderada
    nr_p = 1 - weighted.mean(R, w0))
```

```
## # A tibble: 1 x 2
##   nr_np  nr_p
##   <dbl> <dbl>
## 1 0.474 0.476
```

```
summary(muestra$w0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   104.4   110.8   124.8   125.6   140.3   162.0
```

```
#considerando por estratos

muestra %>%
  group_by(estrato) %>%
  summarise(
    # tasa de no respuesta no ponderada
    nr_np = mean(R),
    # tasa de no respuesta ponderada
    nr_p = weighted.mean(R, w0))
```

```
## # A tibble: 12 x 3
##    estrato nr_np  nr_p
##    <fct>   <dbl> <dbl>
##  1 1       0.455 0.455
##  2 2       0.530 0.530
##  3 3       0.533 0.533
##  4 4       0.547 0.547
##  5 5       0.538 0.538
##  6 6       0.487 0.487
##  7 7       0.543 0.543
##  8 8       0.543 0.543
##  9 9       0.526 0.526
## 10 10      0.508 0.508
## 11 11      0.552 0.552
## 12 12      0.564 0.564
```

```
#considerando por departamento

muestra %>%
  group_by(dpto) %>%
  summarise(
   # tasa de no respuesta no ponderada
    nr_np = mean(R),
    # tasa de no respuesta ponderada
    nr_p = weighted.mean(R, w0))
```

```
## # A tibble: 19 x 3
##    dpto  nr_np  nr_p
##    <fct> <dbl> <dbl>
##  1 1     0.523 0.521
##  2 2     0.545 0.545
##  3 3     0.515 0.512
##  4 4     0.542 0.542
##  5 5     0.511 0.511
##  6 6     0.570 0.570
##  7 7     0.552 0.552
##  8 8     0.576 0.576
##  9 9     0.556 0.556
## 10 10    0.534 0.534
## 11 11    0.517 0.517
## 12 12    0.498 0.498
## 13 13    0.535 0.535
## 14 14    0.533 0.533
## 15 15    0.545 0.545
## 16 16    0.498 0.494
## 17 17    0.499 0.499
## 18 18    0.540 0.540
## 19 19    0.556 0.556
```

```
#considerando por edad

muestra %>%
  group_by(edad) %>%
```

```
  summarise(
   # tasa de no respuesta no ponderada
    nr_np = mean(R),
    # tasa de no respuesta ponderada
    nr_p = weighted.mean(R, w0))
```

```
## # A tibble: 8 x 3
##   edad     nr_np  nr_p
##   <fct>    <dbl> <dbl>
## 1 [0,14)   0.499 0.486
## 2 [14,20)  0.576 0.566
## 3 [20,25)  0.574 0.567
## 4 [25,30)  0.549 0.549
## 5 [30,40)  0.541 0.541
## 6 [40,50)  0.542 0.547
## 7 [50,60)  0.516 0.521
## 8 [60,Inf) 0.499 0.500
```

La tasa de no respuesta no ponderada es del 47,4 % mientras que la ponderada es del 47,6 %. El hecho de que ambas tasas de no respuesta sean similares puede deberse a que los pesos w_0 no son muy disímiles entre sí, siendo su mínimo 104,4; su media 125.6 y su máximo 162. Al considerar la proporción de no respondentes por estrato se puede apreciar que ocurre lo mismo. En este caso se puede observar que la tasa de no respuesta varía según el estrato considerado, siendo el primero (Montevideo bajo) el que cuenta con la mayor tasa, del 56 %, y el doceavo el que cuenta con la menor, del 46 %. Estas diferencias se ven reflejadas también al considerar la tasa de no respondentes por departamento. Resultan bastante diferentes las tasas de no respuesta considerando los distintos segmentos de edad, el de 0 a 14 años y el de 60 en adelante son los que presentan menor tasa de no respuesta, siendo las de los primeros 50 % (no ponderada) y 49 % (ponderada) y 50 % (no ponderada y ponderada) la de los segundos.

Parte 2

Ajuste por no respuesta por medio de post-estratos de no respuesta

Bajo el enfoque de no respuesta considerado, el MAR (*Missing at Random*), se trabaja bajo el supuesto de que la no respuesta no depende de las variables de interés, pero sí es completamente explicada por variables auxiliares. Lo que se puede hacer en este caso es construir un modelo de respuesta basado en la información auxiliar (Ferreira y Zoppolo, 2017).

Siguiendo este enfoque, una manera de realizar el ajuste es mediante clases de no respuesta, creadas en base a información de las unidades presente en el marco muestral. Se crean g clases y se asume que todas las unidades dentro de cada una de ellas tiene la misma probabilidad de responder, cuya fórmula es:

$$\hat{\phi}_{i,g} = TR_w = \frac{\sum_{i \in R} w_i}{\sum_{i \in s} w_i}, \quad i \in g$$

Luego, los ponderadores por no respuesta son:

$$w_i^{nr} = \frac{1}{\pi_i \times \hat{\phi}_{i,g}}$$

(π_i son las probabilidad de inclusión originales)

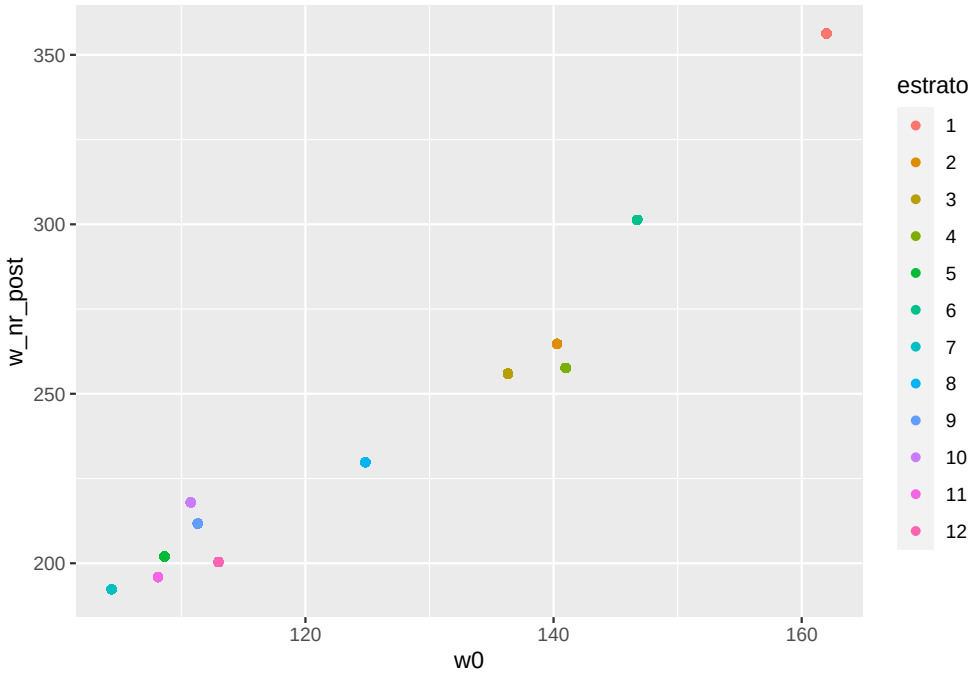
En nuestro caso, la información que podríamos emplear que se encuentra en el marco son los estratos.

```
ajuste_nr_estrato <-  muestra %>%
  group_by(estrato) %>%
  summarise(
    tr = mean(R),
    tr_w = weighted.mean(R, w0)
    )

muestra <- left_join(muestra , select(ajuste_nr_estrato, estrato, tr_w)) %>%
               mutate(w_nr_post = w0/tr_w)
```

```
## Joining, by = "estrato"
```

```
ggplot(muestra) +
 geom_point(aes(w0, w_nr_post, color = estrato))
```




```
# los ponderadores originales se ven bastante alterados
```

Una forma de medir la variabilidad global de los ponderadores (que en caso de ser alta puede resultar en una variabilidad alta de los estimadores) es el efecto diseño de Kish, que representa el incremento en la variabilidad de los estimadores causada por usar ponderadores distintos para las unidades de la muestra con respecto a usar el mismo ponderador. Su fórmula es:

$$deff_w = 1 + \frac{1}{n}\frac{\sum_s (w_k - \bar{w})^2}{\bar{w}^2}$$

donde $\bar{w} = n^{-1}\sum_s w_k$ es el promedio de los ponderadores.

La práctica usual es calcularlo luego de realizar cada ajuste a los estimadores (no respuesta, calibración, en el caso de este trabajo). Se usa la regla empírica que $deff_w > 1,5$ indican que hay valores extremos de los ponderadores que repercuten en los finales.

```
# Efecto diseño de Kish

(deffK_w_nr_post <- deffK(muestra %>%
        filter(R ==1) %>%
        select(w_nr_post) %>%
        pull()
      ))
```

```
## [1] 1.031808
```

En nuestro caso, luego de realizar el ajuste por no respuesta $deff_w \simeq 1,031 < 1,5$, por lo que parecería que no resultó en valores extremos de los ponderadores que repercutirían en la variabilidad de los estimadores.

```r
#estimaciones empleando ajuste por no respuesta mediante clases de no respuesta

design2 <- muestra %>%
  filter(R==1) %>%
  as_survey_design(ids = id_hogar, strata = estrato, weights = w_nr_post)

# Tasa de desempleo
design2 %>%
  filter(activo == 1) %>%
  group_by(desocupado) %>%
  summarise(tasa_desempleo = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   desocupado tasa_desempleo tasa_desempleo_se tasa_desempleo_cv tasa_desempleo_~
##   <fct>               <dbl>             <dbl>             <dbl>            <dbl>
## 1 0                 0.917           0.00336           0.00366             1.09
## 2 1                 0.0831          0.00336           0.0404              1.09
```

Una vez se realiza el ajuste por no respuesta mediante clases de no respuesta, la estimación puntual de la tasa de desempleo cambia, pasa de 8,24 % a 8,31 %. El desvío disminuye, pasando de 0,0033 a 0,00336; lo mismo ocurre con el coeficiente de variación, pasando de 0,04 a 0,0404. El efecto diseño pasa de 1,07 a 1,09.

```r
## Proporción de personas pobres
design2 %>%
  group_by(pobreza) %>%
  summarise(prop_pobres = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   pobreza prop_pobres prop_pobres_se prop_pobres_cv prop_pobres_deff
##   <fct>         <dbl>          <dbl>          <dbl>            <dbl>
## 1 0             0.916        0.00389        0.00424             2.92
## 2 1             0.0837       0.00389        0.0465              2.92
```

En cuanto a la estimación de la proporción de personas pobres, ahora aumenta de 0,0811 a 0,0837. La estimación del error estándar aumenta a 0,00389, mientras que el coeficiente de vairación no presenta cambio al corregir por no respuesta. El efecto diseño presenta un aumento, de 2,84 a 2,92.

```r
## Ingreso promedio
design2 %>%
  summarise(ingreso_prom = survey_mean(ingreso, deff = TRUE, vartype = c('se','cv')))
```

```
##   ingreso_prom ingreso_prom_se ingreso_prom_cv ingreso_prom_deff
## 1     21686.43        238.2187      0.01098469         0.9313662
```

La estimación puntual del ingreso promedio disminuye, de 21799 a 21686, el error estándar presenta una disminución en una unidad. El coeficiente de variación con y sin ajuste son iguales hasta 4 lugares después de la coma, mientras que el efecto diseño presenta una leve disminución; pasa de 0,9355 a 0,9314.

Estimación de propensiones simples de responder utilizando el algoritmo random forest.

El ajuste por propensiones simples consiste en:

$$w_i^{nr} = \frac{1}{\pi_i \times \hat{\phi}_i}$$

donde $\hat{\phi}_i$ es la propensión a responder de la unidad i, la cual se estima a partir de un modelo o un algoritmo, haciendo uso de variables auxiliares conocidas tanto para respondentes como no respondentes.

El algoritmo elegido es *Random Forest* (RF), un método no paramétrico mediante el cual se hace uso de múltiples árboles de decisión para obtener una estimación de la propensión a responder de cada individuo a partir de alguna medida de resumen de la clasificación que hacen los árboles (por ejemplo el modo o la media).

```
# modelamos la no respuesta con random forest
modelo_rf <-  rand_forest(trees = 100) %>%
  set_engine("ranger") %>%
  set_mode("classification") %>%
  fit(as.factor(R) ~ estrato + sexo + edad + dpto, data = muestra)
```

```
# Para ver que tan bien predice el algoritmo
pred_rf <- tibble(predict(modelo_rf, muestra, type= "prob"), predict(modelo_rf, muestra) )

conf_mat(data = bind_cols(select(muestra, R), select(pred_rf, .pred_class)),
         truth = R,
         estimate = .pred_class)
```

```
##           Truth
## Prediction    0    1
##          0 7658 4955
##          1 5603 9788
```

El modelo predice correctamente al 57 % de los no respondentes y al 67 % de los respondentes.

```
# Agregamos las propensiones estimadas con random forest a la muestra
pred_rf <-  pred_rf %>% rename(prop_rf = .pred_1)

muestra <-  muestra %>% bind_cols(select(pred_rf, prop_rf))

# Calculamos los ponderadores ajustados por no respuesta usando las propensiones de arriba
muestra <-  muestra %>%
  mutate(w_nr_rf = w0/prop_rf)
```

```
# Calculamos el efecto diseño de Kish

(deffK_w_nr_rf <- deffK(muestra %>%
        filter(R ==1) %>%
        select(w_nr_rf) %>%
        pull()
      ))
```

```
## [1] 1.073195
```

Una vez ajustados los ponderadores por no respuesta por propensiones simples empleando las estimaciones obtenidas mediante *Random Forest*, el efecto diseño de Kish es de 1,077, que si bien es menor a 1,5, es mayor al valor obtenido luego del ajuste anterior.

```
# Estimaciones empleando ajuste por no respuesta mediante propensiones simples estimadas por random for

design3 <- muestra %>%
  filter(R==1) %>%
  as_survey_design(ids = id_hogar, strata = estrato, weights = w_nr_rf)

## Tasa de desempleo
design3 %>%
  filter(activo == 1) %>%
  group_by(desocupado) %>%
  summarise(tasa_desempleo = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   desocupado tasa_desempleo tasa_desempleo_se tasa_desempleo_cv tasa_desempleo_~
##   <fct>               <dbl>             <dbl>             <dbl>            <dbl>
## 1 0                  0.919           0.00335           0.00365             1.11
## 2 1                  0.0814          0.00335           0.0412              1.11
```

La estimación de la tasa de desempleo al ajustar los ponderadores mediante propensiones simples estimadas por RF es del 8,2 % aproximadamente, disminuyendo en comparación a la estimación realizada con los ponderadores ajustados mediante clases de no respuesta. Se destaca además un aumento del efecto diseño, que ahora es de 1,12 en comparación al de 1,09 del ajuste anterior.

```
## Proporción de personas pobres
design3 %>%
  group_by(pobreza) %>%
  summarise(prop_pobres = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   pobreza prop_pobres prop_pobres_se prop_pobres_cv prop_pobres_deff
##   <fct>         <dbl>          <dbl>          <dbl>            <dbl>
## 1 0             0.917        0.00392        0.00428             2.98
## 2 1             0.0833       0.00392        0.0471              2.98
```

Considerando ambos ajuste por no respuesta realizados hasta el momento, la estimación de la proporción de personas pobres resulta muy similar, siendo la primera realizada 0,0837 y 0,0834 con el último ajuste considerado. En este caso el efecto diseño cuando se ajusta con propensiones simples y estimadas por RF se eleva a 2,99, en comparación al valor del ajuste anterior; 2,92.

```
## Ingreso promedio
design3 %>%
  summarise(ingreso_prom = survey_mean(ingreso, deff = TRUE, vartype = c('se','cv')))
```

```
##   ingreso_prom ingreso_prom_se ingreso_prom_cv ingreso_prom_deff
## 1     21523.31        237.1656      0.01101901         0.9482575
```

La estimación puntual del ingreso promedio considerando el último ajuste realizado es 21491, en comparación a 21686 de la estimación bajo el ajuste por clases de no respuesta. El efecto diseño sigue siendo menor a 1, pero aumenta a 0,949 en el ajuste mediante propensiones simples estimadas por RF. Por otro lado, el desvío estimado disminuye en el caso del último ajuste, siendo de 236,5 en comparación a 238,2.

Ajuste por no respuesta creando clases de no respuesta, utilizando las propensiones estimadas en el punto anterior.

Este tipo de ajuste tiene la intención de hacer más estables las estimaciones finales de la respuesta. Para realizarlo se forman grupos de unidades de la muestra en base a las propensiones estimadas, en nuestro caso creamos clases en base a los quintiles de las propensiones. Una vez se cuenta con los grupos, se resumen los valores de las propensiones dentro de los mismo para contar con un valor representante dentro de la clase, con la mediana o la media (se opta por esta última).

```
quintiles_phi <- quantile(muestra$prop_rf, c(0.2, 0.4, 0.6 , 0.8, 1))

muestra <- muestra %>%
  mutate(
    clase_nr_rf = case_when(
      prop_rf <=  quintiles_phi[1] ~ 1,
      prop_rf > quintiles_phi[1] & prop_rf <= quintiles_phi[2] ~ 2,
      prop_rf > quintiles_phi[2] & prop_rf <= quintiles_phi[3] ~ 3,
      prop_rf > quintiles_phi[3] & prop_rf <= quintiles_phi[4] ~ 4,
      prop_rf > quintiles_phi[4] ~ 5,
    ) %>%  as.factor()
  )

post_estratos_rf <- muestra %>%
  group_by(clase_nr_rf) %>%
  summarise(prop_clase_rf = mean(prop_rf))

muestra <- muestra %>%
  mutate(
    prop_clase_rf =  case_when(
      clase_nr_rf == 1 ~ post_estratos_rf$prop_clase_rf[1],
      clase_nr_rf == 2 ~ post_estratos_rf$prop_clase_rf[2],
      clase_nr_rf == 3 ~ post_estratos_rf$prop_clase_rf[3],
      clase_nr_rf == 4 ~ post_estratos_rf$prop_clase_rf[4],
      clase_nr_rf == 5 ~ post_estratos_rf$prop_clase_rf[5]
    ),
  w_post_nr_rf = w0/prop_clase_rf
  )
```

```
# Efecto diseño de Kish
(deffK_post_nr_rf <-  deffK(muestra %>%
       filter(R ==1) %>%
       select(w_post_nr_rf) %>%
       pull()
      ))
```

```
## [1] 1.067093
```

El efecto diseño en este caso es de 1,07, con lo que al ser menor a 1,5 esto indicaría que el ajuste realizado parece no haber generado valores extremos de los ponderadores. Este valor es menor al obtenido en el ajuste por propensiones simples, pero mayor que el del ajuste por clases de no respuesta.

```
# Estimaciones empleando clases de no respuesta en base a propensiones estimadas por RF

design4 <- muestra %>%
  filter(R==1) %>%
  as_survey_design(ids = id_hogar, strata = estrato, weights = w_post_nr_rf)

## Tasa de desempleo
design4 %>%
  filter(activo == 1) %>%
  group_by(desocupado) %>%
  summarise(tasa_desempleo = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   desocupado tasa_desempleo tasa_desempleo_se tasa_desempleo_cv tasa_desempleo_~
##   <fct>               <dbl>             <dbl>             <dbl>            <dbl>
## 1 0                  0.919            0.00333           0.00362             1.10
## 2 1                  0.0811           0.00333           0.0410              1.10
```

Luego de realizar el ajuste se aprecia que la estimación puntual de la tasa de desempleo es la menor de las tres estimadas, siendo del 8,12 %. El efecto diseño es 1,1; el valor se ubican entre los dos obtenidos con los otros ajustes. El coeficiente de variación se encuentra entre los otros dos estimados para esta variable de interés.

```
## Proporción de personas pobres
design4 %>%
  group_by(pobreza) %>%
  summarise(prop_pobres = survey_mean(deff = TRUE, vartype = c('se','cv')))
```

```
## # A tibble: 2 x 5
##   pobreza prop_pobres prop_pobres_se prop_pobres_cv prop_pobres_deff
##   <fct>         <dbl>          <dbl>          <dbl>            <dbl>
## 1 0             0.917         0.00390        0.00426             2.96
## 2 1             0.0830        0.00390        0.0470              2.96
```

La proporción de personas pobres también es la menor entre las obtenidas luego de los distintos ajustes. El efecto diseño y coeficiente de variación en este caso se encuentra entre los de los estimados luego de los otros ajustes.

```
## Ingreso promedio
design4 %>%
  summarise(ingreso_prom = survey_mean(ingreso, deff = TRUE, vartype = c('se','cv')))
```

```
##   ingreso_prom ingreso_prom_se ingreso_prom_cv ingreso_prom_deff
## 1     21575.45        238.2837      0.01104421         0.9540805
```

La estimación puntual del ingreso en este caso está entre las otras dos anteriores, lo mismo que el efecto diseño y el coeficiente de variación.

Parte 3

Calibración de los ponderadores

Para seleccionar cuál de los ponderadores ajustados por no respuesta utilizaremos para la calibración, tomamos como criterio seleccionar el que compute el menor efecto diseño de Kish. Bajo este criterio se elijen los ponderadores ajustados por clases de no respuesta.

```
data<- data.frame(deffK_w_nr_post, deffK_w_nr_rf, deffK_post_nr_rf)
colnames(data)[1]<- "postest-NR"
colnames(data)[2]<- "NR-RF"
colnames(data)[3]<- "postest-NR-RF"
data
```

```
##   postest-NR    NR-RF postest-NR-RF
## 1   1.031808 1.073195      1.067093
```

Conteos poblacionales de las tres variables auxiliares

Para los estimadores calibrados utilizaremos el método de post-estratificación incompleta (raking). El objetivo es calibrar los ponderadores utilizando las variables auxiliares edad, sexo y departamento. El raking se realiza post-estratificando una variable por turno, obteniendo los factores de ajustes g_i como el resultado de un ajuste iterativo de los totales estimados de las marginales, hasta alcanzar un error prefijado entre las estimaciones y los verdaderos totales poblacionales.

Previo a la calibración post-estratificada incompleta (raking), calculamos el verdadero conteo poblacional de las tres variables a emplear para luego contrastar con el tamaño estimado.

```
pop_count_dpto
```

```
## # A tibble: 19 x 2
##    dpto     Freq
##    <fct>   <int>
##  1 1     1383135
##  2 2       74075
##  3 3      603750
##  4 4       89630
##  5 5      131297
##  6 6       58975
##  7 7       26485
##  8 8       69324
##  9 9       58699
## 10 10     195005
## 11 11     119882
## 12 12      58308
## 13 13     109039
## 14 14      74238
## 15 15     133707
## 16 16     118270
## 17 17      83714
## 18 18      92894
## 19 19      50485
```

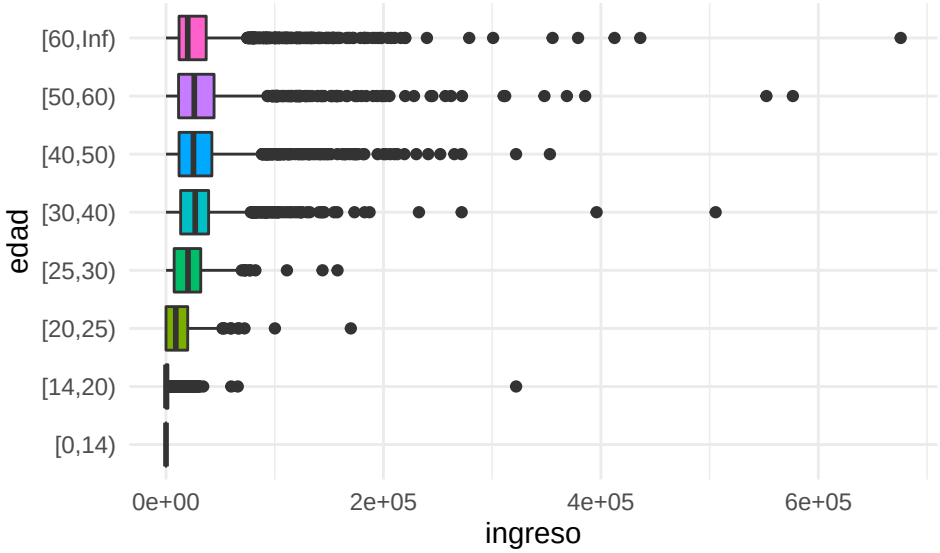
```
pop_count_edad
```

```
## # A tibble: 8 x 2
##   edad       Freq
##   <fct>     <dbl>
## 1 [0,14)   646382
## 2 [14,20)  301873
## 3 [20,25)  269085
## 4 [25,30)  261182
## 5 [30,40)  485015
## 6 [40,50)  464651
## 7 [50,60)  403596
## 8 [60,Inf) 699123
```

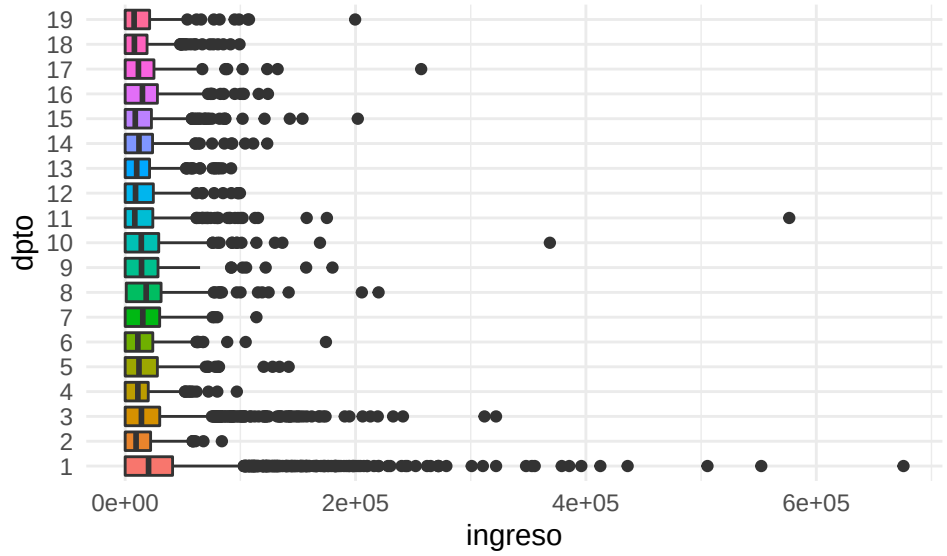
```
pop_count_sexo
```

```
## # A tibble: 2 x 2
##   sexo     Freq
##   <fct>   <dbl>
## 1 1     1711487
## 2 2     1819420
```

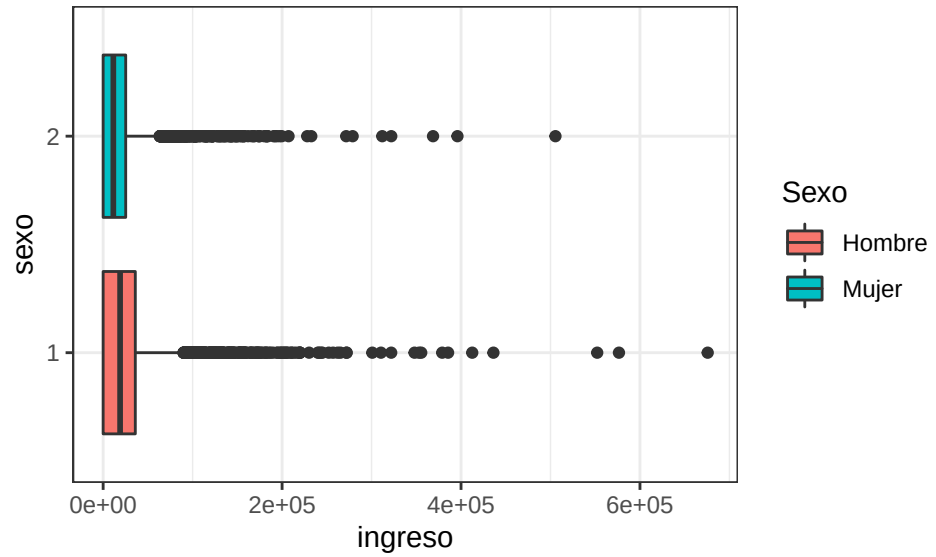
Visualización de los post-estratos con las variables auxiliares para ver si explican a la variable de interés.



Al hacer la visualización de los datos con la variable edad, vemos que esta explica en cierta medida la variabilidad de los datos, notándose que a mayor edad hasta los 60 años la distribución de los ingresos tiende a acumularse a mayores valores, luego de los 60 hay una caída en el ingreso mediano. Por lo tanto, la variable auxiliar edad puede resultar útil para mejorar las estimaciones.



En la visualización de la variable ingreso por departamento podemos notar que no hay diferencias considerables a nivel ingresos entre los departamentos, con la excepción de Montevideo (Departamento 1), el cual cuenta con una mayor cantidad de atípicos.



Por último, el sexo resulta útil para explicar la ingreso de la persona. Observándose que la mediana de la distribución de los ingresos para los hombres resulta mayor a la de las mujeres.

Diseño muestral

Para la muestra seleccionada, usaremos como ponderadores los ajustados por clases de no respuesta.

```
diseno <- muestra %>%
  filter(R==1) %>%
  as_survey_design(ids = id_hogar, strata = estrato, weights = w_nr_post)
```

Post estratificación incompleta raking con las variables auxiliares edad, sexo y departamento de la forma tradicional

```
rake1 <- rake(design=diseno,
           sample.margins=list(~sexo,~edad,~dpto),
           population.margins=list(pop_count_sexo,
                                   pop_count_edad,
                                   pop_count_dpto)
           )
```

```
svytotal(~sexo,rake1)
```

```
##         total     SE
## sexo1 1711490 0.0026
## sexo2 1819422 0.0026
```

```
svytotal(~edad,rake1)
```

```
##                total    SE
## edad[0,14)    646383 7e-04
## edad[14,20)   301873 4e-04
## edad[20,25)   269085 4e-04
## edad[25,30)   261182 4e-04
## edad[30,40)   485016 5e-04
## edad[40,50)   464652 5e-04
## edad[50,60)   403597 4e-04
## edad[60,Inf)  699124 7e-04
```

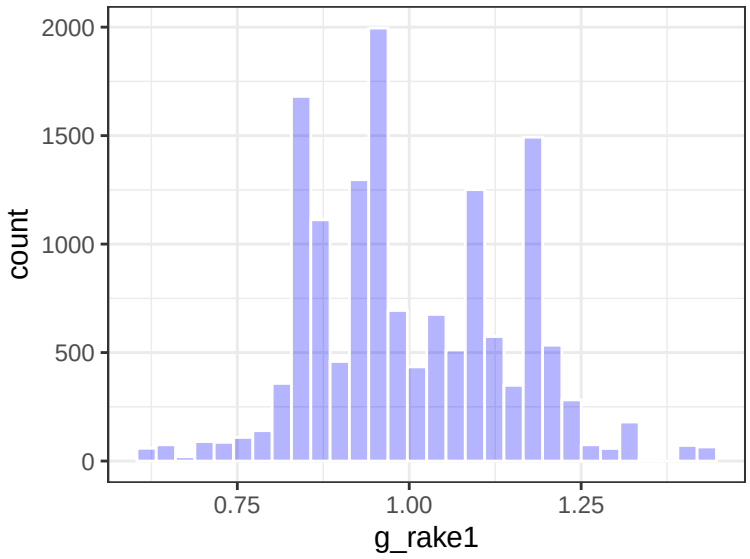
```
svytotal(~dpto,rake1)
```

```
##          total SE
## dpto1  1383135  0
## dpto2    74075  0
## dpto3   603750  0
## dpto4    89630  0
## dpto5   131297  0
## dpto6    58975  0
## dpto7    26485  0
## dpto8    69324  0
## dpto9    58699  0
## dpto10  195005  0
## dpto11  119882  0
## dpto12   58308  0
## dpto13  109039  0
## dpto14   74238  0
## dpto15  133707  0
## dpto16  118270  0
## dpto17   83714  0
## dpto18   92894  0
## dpto19   50485  0
```

```
mean(muestra$g_rake1)
```

```
## [1] 1.003053
```

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



Al verificar si los ponderadores calibrados cumple con la ecuación de calibración, podemos notar que sí lo hacen. Con la variable departamento lo hacen al valor exacto sin error, mientra que a la variable edad y sexo lo hacen con un error muy pequeño cercano a cero.

Post estratificación incompleta raking con las variables auxiliares edad, sexo y departamento usando la función calibrate

```
counts <- c(sum(pop_count_sexo$Freq),
         pop_count_dpto$Freq[-1],
         pop_count_sexo$Freq[-1],
         pop_count_edad$Freq[-1])
```

```
r2 <- calibrate(design=diseno,
            formula=~dpto+sexo+edad,
            population=counts,
            calfun="raking")
```

```
svytotal(~sexo,r2)
```

```
##         total SE
## sexo1 1711487  0
## sexo2 1819420  0
```

```
svytotal(~edad,r2)
```

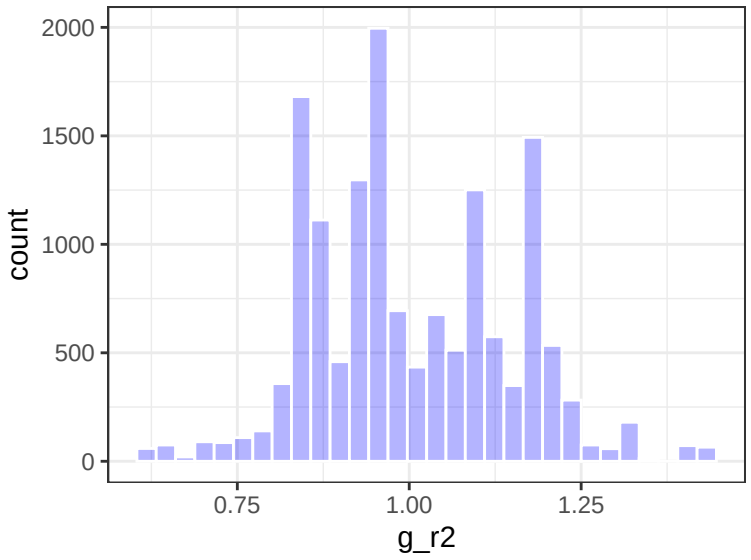
```
##                total SE
## edad[0,14)    646382  0
## edad[14,20)   301873  0
## edad[20,25)   269085  0
## edad[25,30)   261182  0
## edad[30,40)   485015  0
## edad[40,50)   464651  0
## edad[50,60)   403596  0
## edad[60,Inf) 699123  0
```

```
svytotal(~dpto,r2)
```

```
##           total SE
## dpto1  1383130  0
## dpto2    74075  0
## dpto3   603750  0
## dpto4    89630  0
## dpto5   131297  0
## dpto6    58975  0
## dpto7    26485  0
## dpto8    69324  0
## dpto9    58699  0
## dpto10  195005  0
## dpto11  119882  0
## dpto12   58308  0
## dpto13  109039  0
## dpto14   74238  0
## dpto15  133707  0
## dpto16  118270  0
## dpto17   83714  0
## dpto18   92894  0
## dpto19   50485  0
```

```
muestra <-  muestra  %>%
  filter(R==1) %>%
  mutate(g_r2 = weights(r2)/w_nr_post)
```

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




```
mean(muestra$g_r2)
```

```
## [1] 1.003052
```

Al computar los ponderadores con la función raking, podemos notar que estima los valores poblacionales sin margen de error en las tres variables.

Parte 4

```
#Calculo los ponderadores finales ajustados por no respuesta y calibrados
muestra <- muestra %>%
  mutate(w_nr_calibrados = w_nr_post*g_r2)

design_final <- muestra %>%
  filter(R==1) %>%
  as_survey_design(ids = id_hogar, strata = estrato, weights = w_nr_calibrados)


set.seed(1)
```

Estimaciones realizadas con los ponderadores finales ajustados por no respuesta y calibración

```
#Tasa de desempleo
#Al igual que en la parte 1, nos interesan los desempleados considerando el grupo de los activos.

design_final %>%
filter(activo == 1) %>%
group_by(desocupado) %>%
summarise(tasa_desempleo = survey_mean(deff = FALSE, vartype=NULL))
```

```
## # A tibble: 2 x 2
##   desocupado tasa_desempleo
##   <fct>               <dbl>
## 1 0                   0.916
## 2 1                   0.0843
```

Tenemos que la estimación puntual de la tasa de desempleo utilizando los ponderadores ajustados por no respuesta y calibrados es 0,0843, respecto a la tasa de 0,0824 que habíamos obtenido en la parte 1.

```
#Proporción de personas pobres
design_final %>%
group_by(pobreza) %>%
summarise(prop_pobres = survey_mean(deff = FALSE, vartype=NULL))
```

```
## # A tibble: 2 x 2
##   pobreza prop_pobres
##   <fct>         <dbl>
## 1 0             0.912
## 2 1             0.0880
```

La proporción de personas pobres estimada con los ponderadores finales es de 0,088, mientras que la estimación realizada en la parte 1 era de 0,0811.

```r
## Ingreso promedio
design_final %>%
summarise(ingreso_prom = survey_mean(ingreso, vartype=NULL))
```

```
##   ingreso_prom
## 1     20863.83
```

La estimación puntual del ingreso promedio para el total país realizada con los ponderadores finales es de 20864, inferior al 21799 estimado en la parte 1.

Estimaciones realizadas a nivel departamental

```r
#Tasa de desempleo
#Al igual que en la parte 1, nos interesan los desempleados considerando el grupo de los activos.

(tasa_desempleo_survey_dptos <- svyby(
  ~desocupado,
  by =~dpto,
  design_final %>% filter(activo == 1),
  FUN= svymean,
  vartype=c("se","cv","ci")) %>%
  transmute(dpto,
            tasa_desempleo = desocupado1,
            tasa_desempleo_se = se.desocupado1,
            CI_lower = ci_l.desocupado1,
            CI_upper= ci_u.desocupado1,
            CV = cv.desocupado1))
```

```
##    dpto tasa_desempleo tasa_desempleo_se   CI_lower   CI_upper         CV
## 1     1     0.08063068       0.005368948 0.07010773 0.09115362 0.06658692
## 2     2     0.09222236       0.021004629 0.05105405 0.13339068 0.22776069
## 3     3     0.08942103       0.008854581 0.07206637 0.10677569 0.09902124
## 4     4     0.04476621       0.015030487 0.01530700 0.07422542 0.33575518
## 5     5     0.07681823       0.016546797 0.04438710 0.10924935 0.21540196
## 6     6     0.16126584       0.027601728 0.10716745 0.21536423 0.17115669
## 7     7     0.12498209       0.033908217 0.05852321 0.19144098 0.27130460
## 8     8     0.05612753       0.017166214 0.02248237 0.08977269 0.30584305
## 9     9     0.05319328       0.018150362 0.01761923 0.08876734 0.34121531
## 10   10     0.08208560       0.016094767 0.05054044 0.11363076 0.19607296
## 11   11     0.08958731       0.018989571 0.05236843 0.12680619 0.21196720
## 12   12     0.08330363       0.026705804 0.03096122 0.13564605 0.32058391
## 13   13     0.05740650       0.014932499 0.02813934 0.08667366 0.26011860
## 14   14     0.08752165       0.022408743 0.04360132 0.13144198 0.25603656
## 15   15     0.09002738       0.017808235 0.05512388 0.12493088 0.19780911
## 16   16     0.06879262       0.016230946 0.03698055 0.10060468 0.23594024
## 17   17     0.15286770       0.031383587 0.09135700 0.21437840 0.20529901
## 18   18     0.09797292       0.020969316 0.05687382 0.13907203 0.21403175
## 19   19     0.10432281       0.028931310 0.04761848 0.16102714 0.27732487
```

Realizamos la estimación de la tasa de desempleo, con su correspondiente error estándar, coeficiente de variación e intervalo de confianza al 95 % para los distintos departamentos. Podemos ver que el departamento

con una mayor tasa de desempleo es el número 6 con 0,16. El que tiene un mayor error estándar estimado es el número 17, con un valor de 0,031.

```
#Proporción de personas pobres

(prop_pobres_survey_dptos <- svyby(
  ~pobreza,
  by =~dpto, design_final ,
  FUN= svymean,
  vartype=c("se","cv","ci")) %>%
  transmute(
    dpto,
    prop_pobres = pobreza1,
    prop_pobres_se = se.pobreza1,
    CI_lower = ci_l.pobreza1,
    CI_upper = ci_u.pobreza1,
    CV = cv.pobreza1))
```

```
##    dpto prop_pobres prop_pobres_se     CI_lower   CI_upper         CV
## 1     1  0.12851493    0.008035721  0.112765210 0.14426466 0.06252753
## 2     2  0.06064603    0.019901786  0.021639245 0.09965281 0.32816306
## 3     3  0.05218321    0.007693528  0.037104169 0.06726224 0.14743302
## 4     4  0.08201289    0.023439513  0.036072291 0.12795349 0.28580278
## 5     5  0.04896699    0.016051802  0.017506041 0.08042795 0.32780860
## 6     6  0.05874842    0.022401403  0.014842477 0.10265436 0.38131073
## 7     7  0.03019073    0.023713264 -0.016286416 0.07666787 0.78544858
## 8     8  0.04748346    0.019084946  0.010077654 0.08488927 0.40192829
## 9     9  0.07974982    0.022944467  0.034779491 0.12472015 0.28770556
## 10   10  0.03332764    0.013175208  0.007504706 0.05915057 0.39532377
## 11   11  0.11819313    0.025465381  0.068281902 0.16810436 0.21545567
## 12   12  0.04512861    0.020446061  0.005055064 0.08520215 0.45306209
## 13   13  0.10807866    0.024600042  0.059863466 0.15629386 0.22761238
## 14   14  0.06634370    0.022432105  0.022377584 0.11030982 0.33811959
## 15   15  0.04475807    0.013831883  0.017648077 0.07186806 0.30903664
## 16   16  0.02979550    0.013578873  0.003181398 0.05640960 0.45573570
## 17   17  0.07596791    0.024030057  0.028869863 0.12306596 0.31631852
## 18   18  0.10723749    0.025008101  0.058222512 0.15625247 0.23320297
## 19   19  0.10459096    0.033589851  0.038756065 0.17042586 0.32115443
```

El departamento con mayor proporción de pobreza estimada es Montevideo con 0,13 aproximadamente. El intervalo de confianza de la proporción de pobreza para este departamento es $(0,113; 0,144)$. El que tiene mayor desvío es el departamente número 19, con un desvío de 0,034. El departamento que le sigue a Montevideo en proporción de pobres es el número 11.

```
## Ingreso promedio
design_final %>%
  group_by(dpto) %>%
summarise(ingreso_prom = survey_mean(ingreso, vartype=c("se", "cv", "ci"), level=0.95)) %>%
  arrange(desc(ingreso_prom)) %>%
  rename(CI_low = "ingreso_prom_low", CI_upp = "ingreso_prom_upp")
```

```
## # A tibble: 19 x 6
##    dpto  ingreso_prom ingreso_prom_se ingreso_prom_cv CI_low CI_upp
```

```
##    <fct>        <dbl>          <dbl>          <dbl>  <dbl>  <dbl>
##  1 1           26514.          446.         0.0168 25640. 27387.
##  2 8           22412.         1544.         0.0689 19384. 25439.
##  3 3           21212.          615.         0.0290 20006. 22418.
##  4 7           19211.         1760.         0.0916 15761. 22660.
##  5 10          18480.          946.         0.0512 16626. 20334.
##  6 9           18155.         1326.         0.0730 15556. 20754.
##  7 16          18097.          835.         0.0461 16461. 19732.
##  8 5           16429.          735.         0.0447 14988. 17869.
##  9 11          15632.         1231.         0.0787 13220. 18044.
## 10 17          15490.         1176.         0.0759 13185. 17794.
## 11 14          14858.          863.         0.0581 13167. 16550.
## 12 6           14833.         1087.         0.0733 12701. 16964.
## 13 12          14685.         1255.         0.0855 12224. 17146.
## 14 15          14013.          736.         0.0525 12571. 15455.
## 15 19          14007.         1317.         0.0941 11424. 16590.
## 16 13          12974.          706.         0.0544 11590. 14357.
## 17 4           12615.          636.         0.0505 11368. 13863.
## 18 2           12465.          644.         0.0517 11202. 13727.
## 19 18          11910.          707.         0.0594 10524. 13295.
```

Para la estimación del ingreso promedio a nivel de departamentos, tenemos nuevamente que el departamento con mayor ingreso promedio es Montevideo con 26514, si bien tiene el menor error estándar de 535.

Cálculo del error estándar a partir del método del último conglomerado y Bootstrap Rao Wu

A continuación compararemos los errores estándar calculados anteriormente a nivel departamento utilizando el método del último conglomerado que utiliza por defecto el paquete survey, con los calculados utilizando Bootstrap Rao Wu.

El método del último conglomerado se utiliza para la estimación de los errores estándar en diseños por conglomerados, donde los conglomerados son seleccionados con probabilidades proporcionales al tamaño. Por su parte, el Bootstrap es un método de remuestreo que genera replicas de la población original mediante muestras aleatorias con reposición de la muestra, las cuales se utilizan para calcular varias réplicas del estadístico de interés y a partir de esas réplicas estimar propiedades del estadístico como su desviación estándar. En particular utilizamos Bootstrap Rao Wu el cual se aplica a diseños aleatorios, estratificados y en varias etapas de selección.

```r
#Al igual que en la parte 1, nos interesan los desempleados considerando el grupo de los activos.

#Tasa de desempleo

reps <- 500

tasa_desempleo_survey <- design_final %>%
filter(activo == 1) %>%
group_by(desocupado) %>%
summarise(tasa_desempleo = survey_mean(deff = FALSE)) %>%
  filter(desocupado==1) %>%
  mutate(dpto="pais") %>%
  select(-"desocupado")
```

```r
#Tasa de desempleo con SE estimado por bootstrap rao wu

tasa_desempleo_boot <-
design_final %>%
  filter(activo==1) %>%
  as.svrepdesign(design=., type="subbootstrap", replicates=reps) %>%
  svymean(~desocupado, .) %>%
  as.data.frame() %>%
  mutate(dpto="pais") %>%
  subset(., rownames(.) %in% "desocupado1") %>%
  `rownames<-`( NULL ) %>%
  rename(tasa_desempleo_boot="mean", tasa_desempleo_se_boot="SE")


design_bs_desocup <- as.svrepdesign(
  design = design_final %>% filter(activo == 1),
  type="subbootstrap",
  replicates=reps)

tasa_desempleo_boot_dptos <- svyby(
  ~desocupado,
  by = ~dpto,
  design_bs_desocup,
  svymean) %>%
  transmute(
    dpto,
    tasa_desempleo_boot = desocupado1,
    tasa_desempleo_se_boot = se2)
```

```r
rbind(
  left_join(tasa_desempleo_survey, tasa_desempleo_boot, by="dpto") %>%
    relocate(dpto, tasa_desempleo, tasa_desempleo_boot, tasa_desempleo_se, tasa_desempleo_se_boot),
  left_join(tasa_desempleo_survey_dptos, tasa_desempleo_boot_dptos, by="dpto") %>%
    relocate(dpto, tasa_desempleo, tasa_desempleo_boot, tasa_desempleo_se, tasa_desempleo_se_boot)
)
```

```
## # A tibble: 20 x 5
##    dpto  tasa_desempleo tasa_desempleo_boot tasa_desempleo_se tasa_desempleo_se~
##    <chr>          <dbl>               <dbl>             <dbl>              <dbl>
##  1 pais          0.0843              0.0843           0.00342            0.00311
##  2 1             0.0806              0.0806           0.00537            0.00534
##  3 2             0.0922              0.0922           0.0210             0.0212
##  4 3             0.0894              0.0894           0.00885            0.00881
##  5 4             0.0448              0.0448           0.0150             0.0149
##  6 5             0.0768              0.0768           0.0165             0.0173
##  7 6             0.161               0.161            0.0276             0.0278
##  8 7             0.125               0.125            0.0339             0.0343
##  9 8             0.0561              0.0561           0.0172             0.0173
## 10 9             0.0532              0.0532           0.0182             0.0183
## 11 10            0.0821              0.0821           0.0161             0.0152
## 12 11            0.0896              0.0896           0.0190             0.0191
## 13 12            0.0833              0.0833           0.0267             0.0267
```

```
## 14 13           0.0574           0.0574           0.0149           0.0145
## 15 14           0.0875           0.0875           0.0224           0.0239
## 16 15           0.0900           0.0900           0.0178           0.0185
## 17 16           0.0688           0.0688           0.0162           0.0156
## 18 17           0.153            0.153            0.0314           0.0296
## 19 18           0.0980           0.0980           0.0210           0.0205
## 20 19           0.104            0.104            0.0289           0.0288
```

Con la estimación del error estándar realizada por el método del último conglomerado, el SE estimado a nivel país es 0.00342, mientras que el SE estimado a través de Bootstrap Rao Wu es 0,00339.

```
#Proporción de personas pobres
prop_pobres_survey <-
design_final %>%
group_by(pobreza) %>%
summarise(prop_pobres = survey_mean(deff = FALSE)) %>%
  filter(pobreza==1) %>%
  mutate(dpto="pais") %>%
  select(-"pobreza")


prop_pobres_boot <-
  design_final %>%
  as.svrepdesign(design=., type="subbootstrap", replicates=reps) %>%
  svymean(~pobreza, .) %>%
  as.data.frame() %>%
  mutate(dpto="pais") %>%
  subset(., rownames(.) %in% "pobreza1") %>%
  `rownames<-`( NULL ) %>%
  rename(prop_pobreza_boot="mean", prop_pobreza_se_boot="SE")


# Proporción de pobres por departamento

prop_pobres_survey_dptos <- svyby(~pobreza, by =~dpto, design_final , FUN= svymean) %>%
  transmute(dpto, prop_pobres = pobreza1, prop_pobres_se = se.pobreza1)


design_bs <- as.svrepdesign(design = design_final, type = "subbootstrap", replicates=reps)


prop_pobres_boot_dptos <- svyby(~pobreza, by =~dpto, design_bs , FUN= svymean) %>%
  transmute(dpto, prop_pobreza_boot = pobreza1, prop_pobreza_se_boot = se2)
```

```
rbind(
  left_join(prop_pobres_survey, prop_pobres_boot, by="dpto") %>%
    relocate(dpto, prop_pobres, prop_pobres_se, prop_pobreza_boot , prop_pobreza_se_boot),
  left_join(prop_pobres_survey_dptos, prop_pobres_boot_dptos, by="dpto") %>%
    relocate(dpto, prop_pobres, prop_pobres_se, prop_pobreza_boot , prop_pobreza_se_boot)
)
```

```
## # A tibble: 20 x 5
##    dpto  prop_pobres prop_pobres_se prop_pobreza_boot prop_pobreza_se_boot
```

```
##    <chr>       <dbl>          <dbl>             <dbl>                <dbl>
##  1 pais       0.0880        0.00411           0.0880              0.00404
##  2 1          0.129         0.00804           0.129               0.00849
##  3 2          0.0606        0.0199            0.0606              0.0199
##  4 3          0.0522        0.00769           0.0522              0.00738
##  5 4          0.0820        0.0234            0.0820              0.0241
##  6 5          0.0490        0.0161            0.0490              0.0158
##  7 6          0.0587        0.0224            0.0587              0.0236
##  8 7          0.0302        0.0237            0.0302              0.0243
##  9 8          0.0475        0.0191            0.0475              0.0188
## 10 9          0.0797        0.0229            0.0797              0.0236
## 11 10         0.0333        0.0132            0.0333              0.0127
## 12 11         0.118         0.0255            0.118               0.0257
## 13 12         0.0451        0.0204            0.0451              0.0208
## 14 13         0.108         0.0246            0.108               0.0242
## 15 14         0.0663        0.0224            0.0663              0.0224
## 16 15         0.0448        0.0138            0.0448              0.0136
## 17 16         0.0298        0.0136            0.0298              0.0134
## 18 17         0.0760        0.0240            0.0760              0.0237
## 19 18         0.107         0.0250            0.107               0.0239
## 20 19         0.105         0.0336            0.105               0.0324
```

La estimación del SE a nivel país realizada con el método del último conglomerado es 0.00411, mientras que la realizada con Bootstrap Rao Wu es 0.00410. El departamento con el mayor SE bajo los dos métodos es el número 19.

```
## Ingreso promedio
ingreso_promedio_survey <-
  design_final %>%
summarise(ingreso_prom = survey_mean(ingreso)) %>%
  mutate(dpto="pais")

ingreso_promedio_boot <-
  design_final %>%
  as.svrepdesign(design=., type="subbootstrap", replicates=reps) %>%
  svymean(~ingreso, design=.)  %>%
  as.data.frame() %>%
  mutate(dpto="pais") %>%
  as.data.frame %>%
  `rownames<-`( NULL ) %>%
  rename(ingreso_promedio_boot="mean", ingreso_promedio_se_boot="SE")

# Ingreso promedio por departamento

ingreso_promedio_survey_dptos <-
  design_final %>%
  group_by(dpto) %>%
  summarise(ingreso_prom = survey_mean(ingreso))


ingreso_promedio_boot_dptos <-
  design_final %>%
  as.svrepdesign(design=., type="subbootstrap", replicates=reps) %>%
  svyby(~ingreso, ~dpto, design=., svymean) %>%
```

```
  as.data.frame() %>%
  `rownames<-`( NULL ) %>%
  rename(ingreso_promedio_boot="ingreso", ingreso_promedio_se_boot="se")
```

```
rbind(
  left_join(ingreso_promedio_survey, ingreso_promedio_boot, by="dpto") %>%
    relocate(dpto, ingreso_prom, ingreso_promedio_boot, ingreso_prom_se, ingreso_promedio_se_boot),
  left_join(ingreso_promedio_survey_dptos, ingreso_promedio_boot_dptos, by="dpto") %>%
    relocate(dpto, ingreso_prom, ingreso_promedio_boot, ingreso_prom_se, ingreso_promedio_se_boot)
)
```

```
##    dpto ingreso_prom ingreso_promedio_boot ingreso_prom_se
## 1  pais     20863.83              20863.83        230.9334
## 2     1     26513.58              26513.58        445.7250
## 3     2     12464.68              12464.68        644.1569
## 4     3     21212.07              21212.07        615.2384
## 5     4     12615.31              12615.31        636.4583
## 6     5     16428.58              16428.58        734.7708
## 7     6     14832.68              14832.68       1087.3498
## 8     7     19210.67              19210.67       1759.7127
## 9     8     22411.51              22411.51       1544.2999
## 10    9     18154.81              18154.81       1325.8827
## 11   10     18480.18              18480.18        945.6630
## 12   11     15631.89              15631.89       1230.6176
## 13   12     14684.60              14684.60       1255.4309
## 14   13     12973.77              12973.77        705.7192
## 15   14     14858.49              14858.49        863.0155
## 16   15     14013.18              14013.18        735.7158
## 17   16     18096.51              18096.51        834.5504
## 18   17     15489.88              15489.88       1175.6603
## 19   18     11909.65              11909.65        706.8827
## 20   19     14007.08              14007.08       1317.4882
##    ingreso_promedio_se_boot
## 1                  231.2209
## 2                  452.0792
## 3                  620.0294
## 4                  636.2249
## 5                  654.5645
## 6                  715.2308
## 7                 1132.5180
## 8                 1796.9564
## 9                 1574.9530
## 10                1272.7349
## 11                 983.1317
## 12                1226.2369
## 13                1324.2168
## 14                 727.7311
## 15                 886.8933
## 16                 688.8454
## 17                 837.7706
## 18                1244.0756
## 19                 724.6336
## 20                1384.9326
```

A partir del método del último conglomerado la estimación del SE para la estimación del ingreso a nivel país es 230,93, mientras que la estimación realizada con Bootstrap Rao Wu es de 231,03. El departamento con una mayor estimación del SE es el número 7, con un valor 1759,7.

Bibliografía

Greg Freedman Ellis and Ben Schneider (2021). *srvyr: 'dplyr'-Like Syntax for Summary Statistics of Survey Data*. R package version 1.0.1. https://CRAN.R-project.org/package=srvyr

Ferreira, J. P. y Zoppolo, G. (2017). *Métodos de ponderación en encuestas complejas: recomendaciones para una buena práctica.*

T. Lumley (2020) *"survey: analysis of complex survey samples"*. R package version 4.0.

R Core Team (2021). R: *A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.