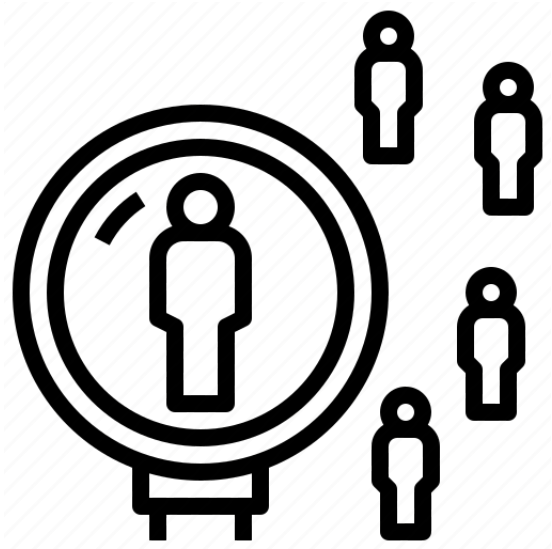
Universidad de la República
Facultad de Ciencias Economicas y de Administración
Licenciatura en Estadística

Muestreo II



Proyecto final

Ignacio Acosta - Valentina Caldiroli - Mauro Loprete

Parte 1 : Estimaciones con ponderadores originales

Se calculan las estimaciones con los ponderadores originales, estimaciones de la tasa de desempleo, la proporción de personas pobres e ingreso promedio.

Dada la existencia de no respuesta en la muestra y el tratamiento realizado, estamos frente a **una postura determinística de la no respuesta**.

A continuación se muestra el código utilizado para realizar las diferentes estimaciones :

```
muestra %>%
    as_survey_design(
        ids = id_hogar,
        weight = w0,
        strata = estrato
    ) %T>%
    assign(
        "diseño",
        .,
        envir = .GlobalEnv
    ) %>%
    filter(
        R > 0
    ) %>%
    summarize(
        td = survey_ratio(
            desocupado,
            activo,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        pobre = survey_mean(
            pobreza,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        yprom = survey_mean(
            ingreso,
            deff = TRUE,
            vartype = c("se","cv")
        )
    ) %>%
    assign(
      "est_originales",
      .,
      envir = .GlobalEnv
    )
```

Los resultados se encuentran en el siguiente cuadro:

Cuadro 1: Estimaciones poblacionales usando ponderadores originales

Variable	Estimación puntual	Error estandar	CV	deff
pobre	0.085	0.004	0.047	2.976
td	0.082	0.003	0.041	1.079
yprom	22037.709	257.069	0.012	0.937

Con base en el cuadro, se puede ver que los errores estándar son relativamente chicos.
Analizando el incremento de varianza respecto a un diseño simple, haciendo uso del efecto $deff$, puede verse como las mismas son altas debido al diseño en varias etapas de esta encuesta.

Tasa de no respuesta

Un enfoque determinista de la tasa de no respuesta plantea a la misma como el cociente entre la cantidad de personas que sí respondieron a la pregunta de interés y el total de personas de la muestra.
Es decir, es posible particionar la muestra en los respondentes r_u y no respondentes $s - r_u$.

$$p_{r_u} = \frac{n_{r_u}}{n_s}$$

Para nuestra muestra particular, esta medida viene dada por :

```
muestra %>%
    summarize.(
        tr = mean(R)
    ) %>%
    mutate.(
        tnr = 1 - tr
    ) %>%
    assign(
        "tasaRespuesta",
        .,
        envir = .GlobalEnv
    )
```

Cuadro 2: Tasa de Respuesta

Tasa de Respuesta	Tasa de No Respuesta
0.54	0.46

En base a este indicador, podemos ver que poco más de la mitad de las personas seleccionadas en la muestra se pudo recabar información.

Por último, podemos ver la tasa de no respuesta poblacional, definida como :

$$\hat{p}_{r_u} = \frac{\sum_{r_u} w_0}{\sum_{r_s} w_0} = \frac{\hat{N}_{r_u}}{\hat{N}_s}$$

```
muestra %>%
    summarize.(
        tr = sum(R*w0) / sum(w0)
    ) %>%
    assign(
        "tasaRespuestapob",
        .,
        envir = .GlobalEnv
    )
```

Cuadro 3: Tasa de Respuesta poblacional

Tasa de respuesta poblacional	Tasa de no respuesta poblacional
0.54	0.46

Considerando 2 cifras significativas en el análisis, la tasa calculada en esta sección coincide con la anterior.

Esta estimación tiene la siguiente interpretación : *%54 es el porcentaje de la población que estoy cubriendo una vez expandida la muestra*, que para este caso particular, **es sumamente bajo**.

Parte 2

Parte a

A continuación se calcuara la tasa de respuesta asumiendo un patrón del tipo MAR. En el mismo se supone que la tasa de respuesta puede ser expresada como una función de un set de covariables, es decir $\Phi_i = \Phi_i(\vec{X})$.

Este tipo de estrategia se basa en crear grupos de individuos con comportamiento similar en la no respuesta, a todos los ponderadores considerados dentro del mismo grupo se le aplicara el mismo ajuste Φ_i.

Los diferentes grupos de no respuesta se construirán con base en el departamento y estrato al que pertenecen, para aprovechar al máximo las variables consideradas en el marco.

A exepción de Montevideo, Canelones y San José que presentan comportamientos disímiles entre algunos subsectores, a cada departamento se le imputara su tasa de respuesta.

```
muestra %>%
    summarize.(
        tr_w_estrato_dpto = weighted.mean(
            R
        ),
        .by = c(
            "estrato",
            "dpto"
        )
    ) %>%
    assign(
        "tr_estrato_dpto",
        .,
        envir = .GlobalEnv
    )

muestra %<>%
    left_join.(
        tr_estrato_dpto,
        by = c(
            "estrato" = "estrato",
            "dpto" = "dpto"
        )
    ) %>%
    mutate.(
        w_nr_post = w0 / tr_w_estrato_dpto
    )
```

La gráfica ilustra los diferentes comportamientos de la tasa de respuesta en los diferentes departamentos, aunque la mayor diferencia se nota en San José, con una tasa de respuesta mayor en el Litoral Sur respecto al de la zona metropolitana.

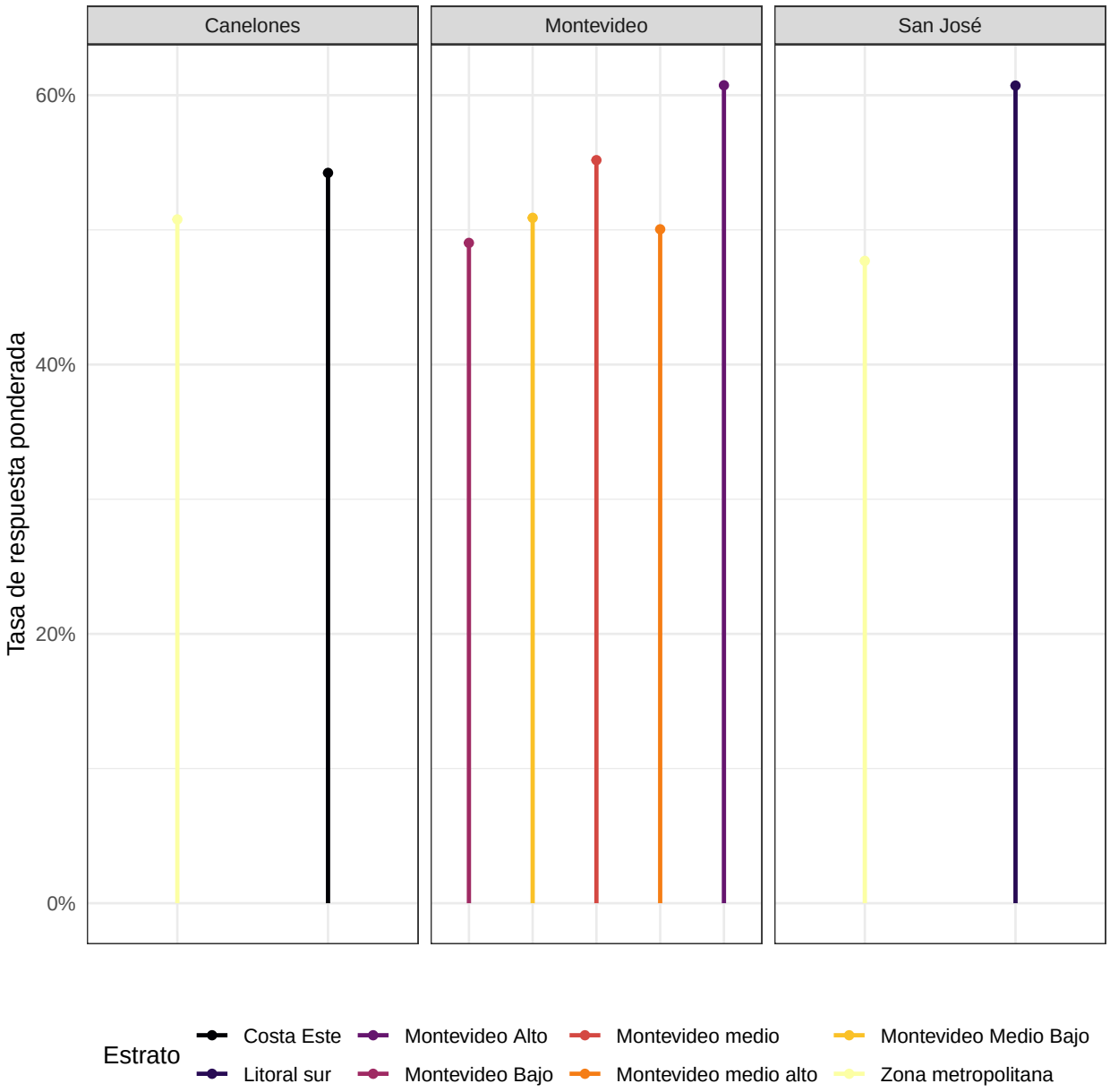


Figura 1: Tasa de respuesta ponderada por departamento y estrato

En lo que refiere a Montevideo,se puede ver una relación creciente (no monótona debido al estrato medio alto) al estrato referido al contexto económico [1] para la tasa de respuesta, mientras que para Canelones no se notan grandes diferencias.

Una vez hecho esto, se continuará con el ajuste por no respuesta para los ponderadores originales:

```
muestra %>%
    as_survey_design(
        ids = id_hogar,
        weight = w_nr_post,
        strata = estrato
    ) %T>%
    assign(
        "diseño",
        .,
        envir = .GlobalEnv
    ) %>%
    filter(
        R > 0
    ) %>%
    summarize(
        td = survey_ratio(
            desocupado,
            activo,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        pobre = survey_mean(
            pobreza,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        yprom = survey_mean(
            ingreso,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        deffK = deff(
          w_nr_post,
          type = "kish"
        )
    ) %>%
    assign(
      "est_ponderados_nr",
      .,
      envir = .GlobalEnv
    )
```

Resultando así, las estimaciones del punto anterior y considerando además el *Efecto diseño de Kish*.

[1] Asumiendo que los estratos son los mismos que el de la ECH

Cuadro 4: Estimaciones poblacionales usando ponderadores por no respuesta

Variable	Estimación puntual	Error estandar	CV	deff	Efecto diseño de Kish
pobre	0.088	0.004	0.047	3.068	1.03
td	0.082	0.003	0.042	1.097	1.03
yprom	21914.940	257.318	0.012	0.952	1.03

Una vez realizado el ajuste, estimaciones de las variables, cálculo de error estándar, coeficiente de variación y efecto diseño, los mismos difirieron un poco con los calculados en el enfoque determinístico.

Algo que puede llamar la atención es como bajó el ingreso promedio, así como todas las estimaciones referidas a su estadístico.
A lo que refiere al efecto diseño de Kish, podemos ver que se encuentra en un nivel de 1.03 un aumento poco considerable en la variabilidad de los ponderadores, respecto a un diseño autoponderado.

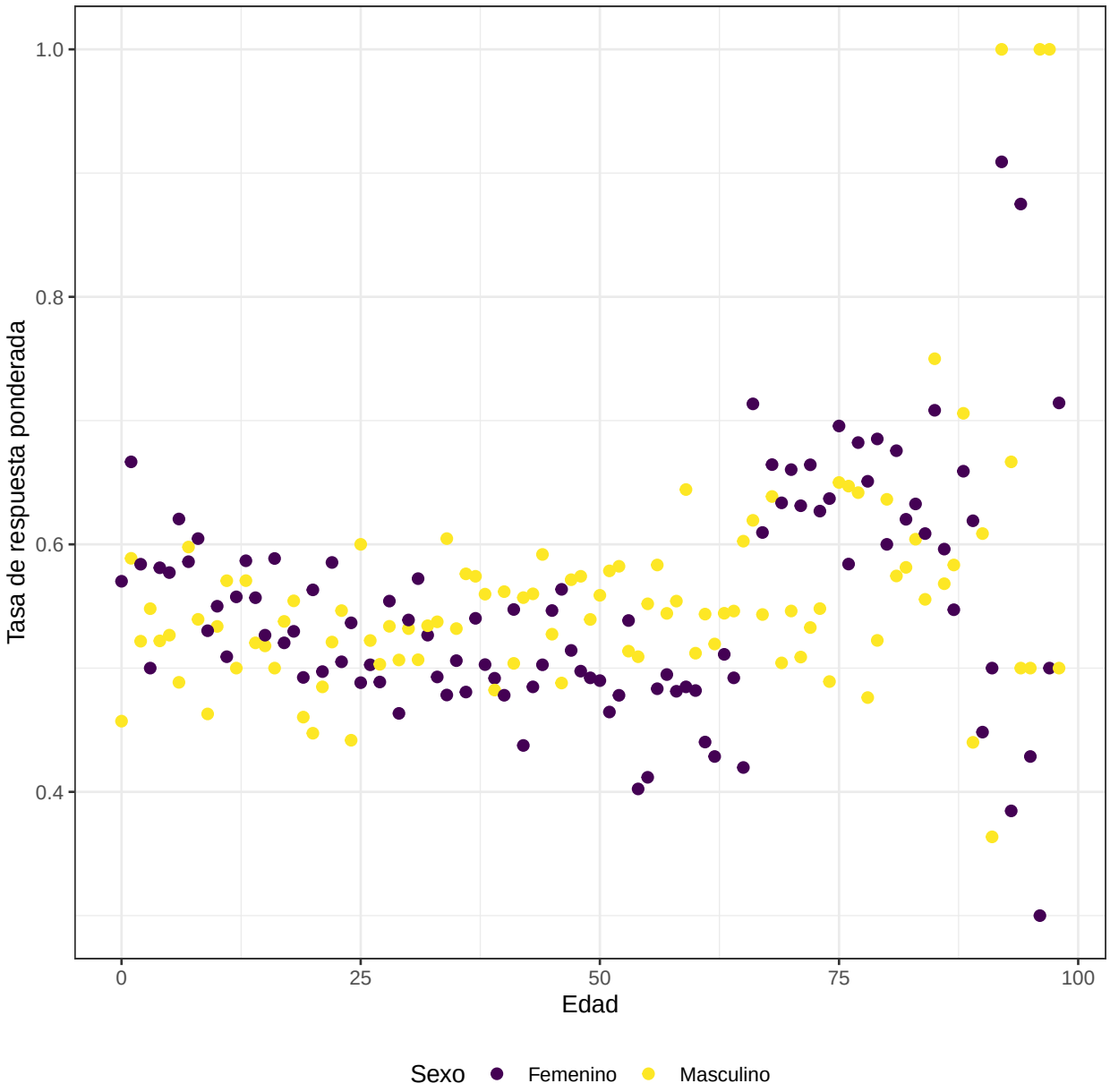
Parte b

Mediante el uso de modelos de Machine Learning de aprendizaje supervizado, estimaremos el *propensity score*. Se hizo uso del método de **Gradient boosting**.

Este tipo de modelos es similar al Random Forest, generando diferentes arboles de decisión y promediando sus resultados.
La diferencia con este último es que la secuencia de árboles es dependiente de la realización anterior, ya que en cada paso se minimiza una función de pérdida (predicciones contra valores observados).

Este tipo de modelos supera a los del tipo de RF, ya que cada nuevo árbol de clasificación se genera tomando en cuenta los errores del paso anterior y no simplemente por azar.

Dado que este tipo de modelos tiende a tener problemas de sobreajuste, debemos de elegir una cantidad de árboles moderada, en nuestro caso 200.



Salvo exepciones, puede verse que en edades jovenes y adultas la tasa de respuesta no es diferente según sexo. En edades ancianas, la tasa de respuesta en mujeres tiende a aumentar respecto a la de los hombres, es por esto que sería bueno incluir estas dos variables en el modelo de clasificación.

```
boost_tree(
    trees = 300
) %>%
set_engine(
    "xgboost"
) %>%
set_mode(
    "classification"
) %>%
fit(
    as.factor(R) ~ estrato + sexo + edad + dpto, data = muestra
```

```
) %>%
assign(
    "modelo_boost",
    .,
    envir = .GlobalEnv
)

## [23:19:19] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the def
```

Análisis de ajuste

```
## Warning in vec2table(truth = truth, estimate = estimate, dnn = dnn, ...): `truth`
was converted to a factor
```

Cuadro 5: Matriz de confusión

Predicción	Observados	
	0	1
0	7597	3947
1	4763	10562

A lo que refiere a la sensibilidad del modelo podemos ver que es de 70 %, mientras que la especificidad es de un casi 62 % y un error total de 31 % .

Se calculan las propensiones individuales y ajustamos los ponderadores por no respuesta:

```
muestra %<>%
    bind_cols.(
        pred_boost
    ) %>%
    mutate.(
        w_nr_boost = (w0*R)/(pred_boost)
    )
```

```
muestra %>%
    as_survey_design(
        ids = id_hogar,
        weight = w_nr_boost,
        strata = estrato
    ) %T>%
    assign(
        "diseño_boost",
        .,
        envir = .GlobalEnv
    ) %>%
    filter(
        R > 0
    ) %>%
    summarize(
        td = survey_ratio(
            desocupado,
            activo,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        pobre = survey_mean(
            pobreza,
```

```
            deff = TRUE,
            vartype = c("se","cv")
        ),
        yprom = survey_mean(
            ingreso,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        deffK = deff(
          w_nr_boost,
          type = "kish"
        )
    ) %>%
    assign(
      "est_ponderados_nr_boost",
      .,
      envir = .GlobalEnv
    )
```

Cuadro 6: Estimaciones poblacionales usando ponderadores por no respuesta utilizando Boosting

Variable	Estimación puntual	Error estandar	CV	deff	Efecto diseño de Kish
pobre	0.089	0.004	0.047	3.171	1.153
td	0.083	0.004	0.044	1.257	1.153
yprom	21878.361	268.478	0.012	1.039	1.153

Una vez realizadas las estimaciones, se puede ver que las mismas difieren poco. Se nota a su vez un leve aumento en los errores estándar y coeficientes de variación, sin embargo se puede notar un aumento en el efecto diseño, así como también en el efecto diseño de Kish.

Este último, si bien es mas alto que el anterior basándonos en la regla empírica, no es algo para preocuparse por el momento.
Por último, veremos un gráfico de dispersión de los ponderadores originales, respecto a los recién ajustados.

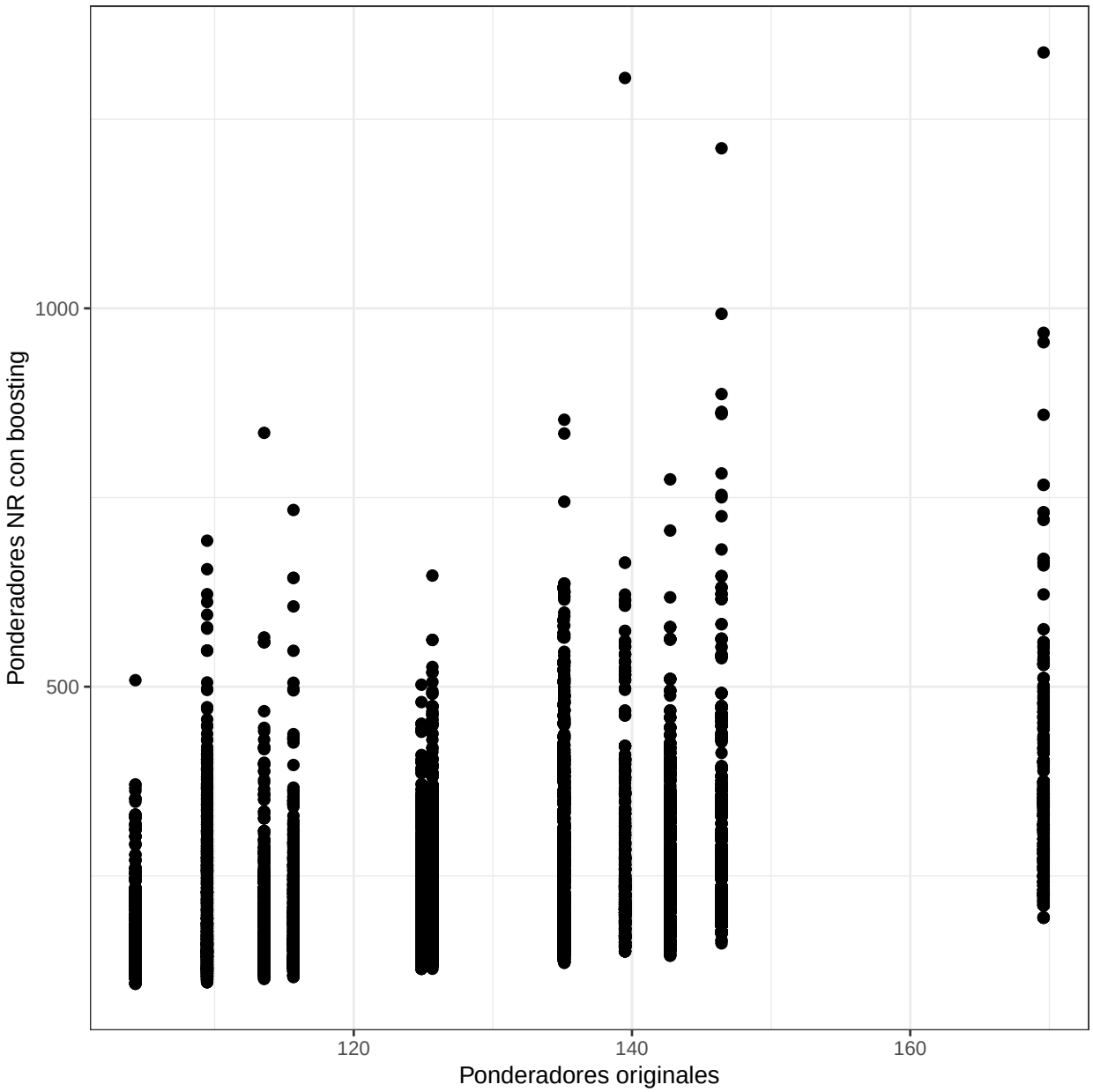


Figura 2: Gráfica de dispersión de ponderadores originales vs propensiones con Boosting

En este gráfico puede verse como se aumentaron los ponderadores, algunos de ellos de forma considerable. Si bien esto puede sonar alarmante los encuestados respondentes tienen que brindar información sobre los que no respondieron. Al tener tasas de respuesta bajas, la variación de los ponderadores, tiene que reflejar esa situación.

Algo a considerar, es que la estimación de la población obtenida con este nuevo sistema de ponderadores es de 3.415.329 siendo la original de 3.518.412.
La estimaciones son similares entre sí, algo que no ocurría utilizando la estrategia de la parte 2, esta crecia a casí el doble 6.582.193.

Parte c

Con el mismo modelo de la parte anterior se ajustarán las tasas de no respuesta mediante propensiones estratificadas.

Primero realizaremos un histograma de estas propensiones :
Podemos ver una distribución simétrica, centrada en valores poco mas grandes que 1/2 aproximándose al valor de la tasa de respuesta.

A continuación calcularemos las propensiones estratificadas utilizando los quintiles de la distribución para agrupar y el estadístico utilizado para resumir el score será la mediana:

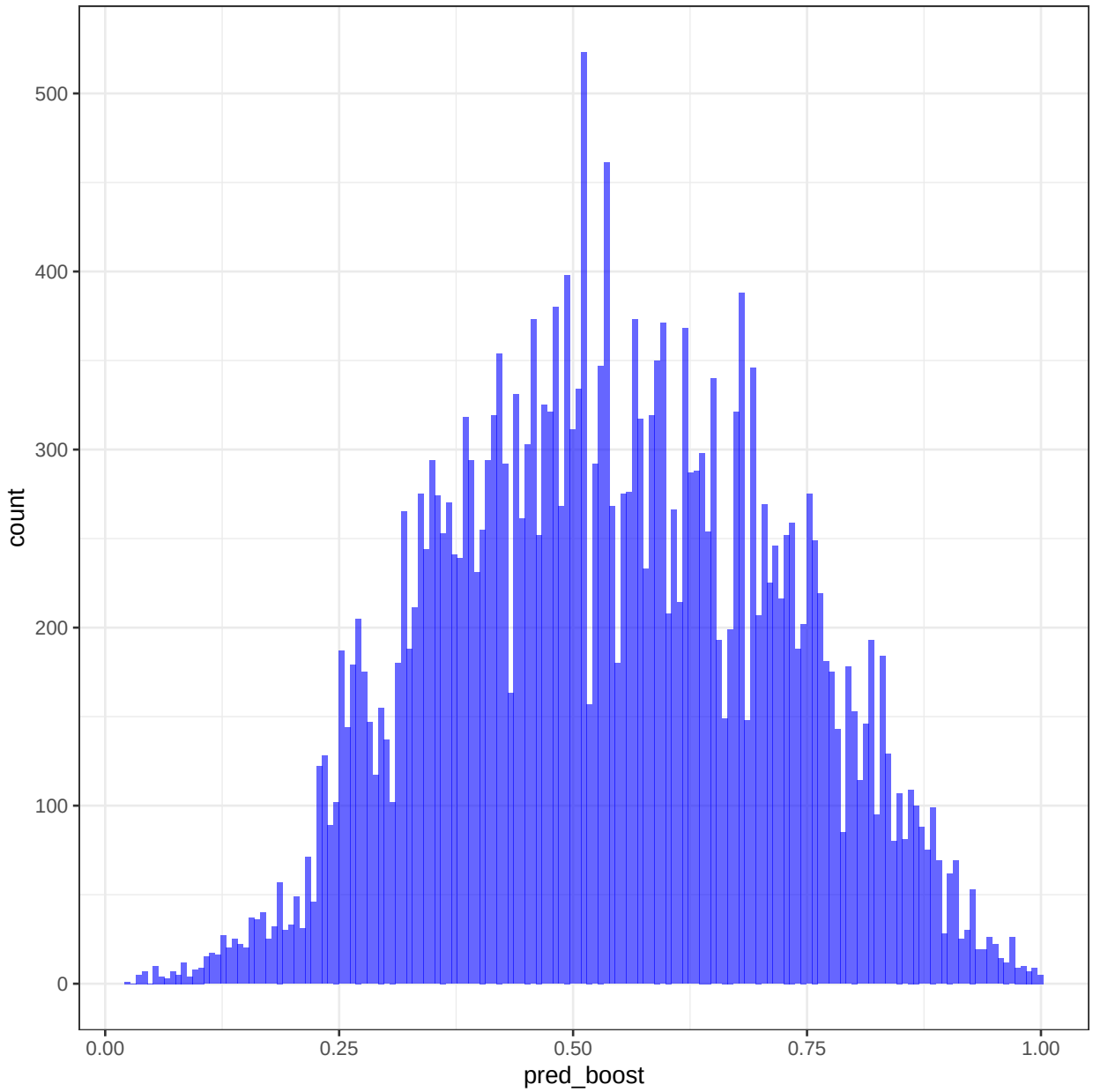


Figura 3: Histograma de propensiones del método Boosting

```
muestra %<>%
    mutate.(
        boost_class = cut(
            pred_boost,
            breaks = quantile(
                pred_boost,
                probs = seq(0,1,1/5)
            ),
            include.lowest = TRUE
        )
    ) %>%
    mutate.(
        ajuste_boost_clases = 1/median(pred_boost),
        .by = boost_class
    ) %>%
    mutate.(
        w_nr_boost_clases = R * w0 * ajuste_boost_clases
    )
```

```
muestra %>%
    as_survey_design(
        ids = id_hogar,
        weight = w_nr_boost_clases,
        strata = estrato
    ) %T>%
    assign(
        "diseño_nr_boost_clases",
        .,
        envir = .GlobalEnv
    ) %>%
    filter(
        R > 0
    ) %>%
    summarize(
        td = survey_ratio(
            desocupado,
            activo,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        pobre = survey_mean(
            pobreza,
            deff = TRUE,
            vartype = c("se","cv")
        ),
        yprom = survey_mean(
            ingreso,
            deff = TRUE,
            vartype = c("se","cv")
        ),
```

```
      deffK = deff(
        w_nr_boost_clases,
        type = "kish"
      )
  ) %>%
  assign(
    "est_ponderados_nr_clases",
    .,
    envir = .GlobalEnv
  )
```

Cuadro 7: Estimaciones poblacionales usando ponderadores por no respuesta utilizando las propensiones del punto anterior por clases

Variable	Estimación puntual	Error estandar	CV	deff	Efecto diseño de Kish
pobre	0.089	0.004	0.047	3.171	1.153
td	0.083	0.004	0.044	1.257	1.153
yprom	21878.361	268.478	0.012	1.039	1.153

En base al cuadro se puede ver que el efecto diseño de Kish sigue controlado, mientras que el error estandar y el efecto diseño aumentarón levemente.

Parte 3

Una vez realizado el ajuste por no respuesta, vamos a calibrar estos ponderadores en base conteos poblacionales por departamento, sexo y edad. Primero construiremos los totales marginales:

```
read_excel(
    here(
        "data",
        "dpto.xlsx"
    )
) %>%
rename.(
    personas_dpto = personas
) %>%
pull.(
    "personas_dpto"
) %>%
unique() %>%
assign(
    "total_dpto",
    .,
    envir = .GlobalEnv
)

edad_sexo <- read_excel(
    here(
        "data",
        "sexo_edad.xlsx"
    )
)

edad_sexo  %>%
    mutate.(
        total = hombres + mujeres,
        .keep = "unused"
    ) %>%
    mutate.(
        edad_tramo = cut(
            edad,
            breaks = c(0,14,20,25,30,40,50,60,Inf),
            right=FALSE
        )
    ) %>%
    summarize.(
        total = sum(total),
        .by = "edad_tramo"
    ) %>%
    rename.(
        total_edad = total
    ) %>%
    pull.(
```

```
        "total_edad"
    ) %>%
    unique() %>%
    assign(
        "total_edad",
        .,
        envir = .GlobalEnv
    )

edad_sexo  %>%
    summarize.(
        hombres = sum(hombres),
        mujeres = sum(mujeres)
    ) %>%
    pivot_longer.(
        names_to = "sexo",
        values_to = "valor"
    ) %>%
    rename.(
        total_sexo = valor
    ) %>%
    mutate.(
        sexo = ifelse.(
            sexo == "hombres",
            1,
            2
        )
    ) %>%
    pull.(
        "total_sexo"
    ) %>%
    unique() %>%
    assign(
        "total_sexo",
         .,
        envir = .GlobalEnv
    )



conteos <- c(
    sum(muestra$w0),
    total_dpto[-1],
    total_edad[-1],
    total_sexo[-1]
)
```

Ahora construiremos los nuevos ponderadores calibrados usando el método de Raking o post-estratificación incompleta, recortamos los mismos y aumentamos el tamaño de iteraciones por defecto.

Con los limites fijados los ponderadores pueden variar hasta un 30 % de su valor obtenido con las propensiones estratificadas del boosting,es decir, al argumento bounds le fijamos -1.3 y 1.3

```
survey::calibrate(
    design = diseño_nr_boost_clases,
    formula = ~ as.factor(dpto) + edad_tramo + as.factor(sexo),
    population = conteos,
    calfun = "raking",
    bounds = c(
      -1.3,
      1.3
    ),
    maxit = 100,
    bounds.const = FALSE
) %>%
assign(
    "w_nr_calibrados",
    .,
    envir = .GlobalEnv
)

muestra %<>%
    mutate.(
        w_nr_boost_clases_calibrados = weights(
            w_nr_calibrados
        )
    )
```

Si realizamos un diagrama de dispersión de los ponderadores calibrados y los obtenidos en el punto anterior, podemos ver su cambio. Si bien al utilizar trimming en el raking los totales poblacionales de variables auxiliares no se cumplen de manera exacta para cada grupo, aunque sigue estimando sin error el total poblacional fijado con los ponderadores originales.
Antes de culminar esta etapa, podemos ver que el efecto diseño de Kish aumento levemente a 1.166.

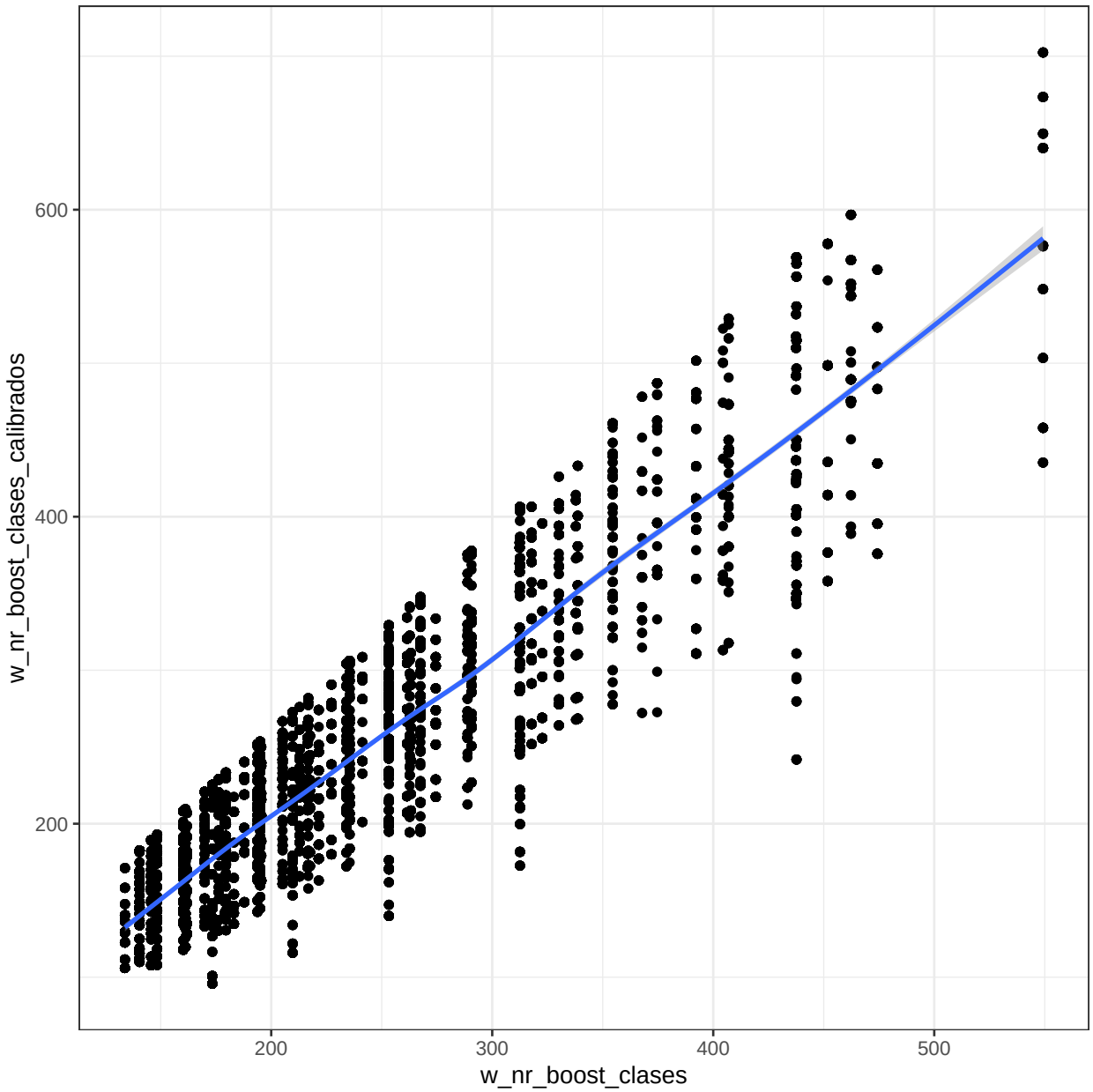


Figura 4: Gráfico de dispersión de ponderadores por clases vs calibrados

Parte 4

Una vez realizado el ajuste por no respuesta y calibrando con las variables poblacionales podemos realizar las estimaciones referidas a la tasa de desempleo, la proporción de personas pobres y el ingreso promedio, para personas empleadas mayores a 25 años.

```
library(survey)

muestra %<>%
  mutate.(
    desocupado = replace_na.(
      desocupado,
      0
    ),
    activo = replace_na.(
      activo,
      0
    )
  )


disenio_final <- as_survey_design(
  muestra,
  ids = id_hogar,
  weight = w_nr_boost_clases_calibrados,
  strata = estrato
)

disenio_final_rep <- survey::as.svrepdesign(
  disenio_final,
  type = "subbootstrap",
  replicates = 500
)
```

Método del útlimo conglomerado

```
svyby(
  formula = ~ desocupado,
  by = ~ dpto,
  FUN = svyratio,
  design = disenio_final,
  denominator = ~ activo,
  vartype = c("ci","se","cv")
) %>%
tibble() %>%
set_names(
  c(
    "Dpto",
    "Est.",
    "SE",
```

```
    "Inter.Inf",
    "Inter.Sup",
    "CV"
  )
) %>%
mutate.(
  across.(
    -Dpto,
    .fns = ~ round(.x,3)
  )
) %>%
kbl(
  booktabs = TRUE,
  caption = "Estimación de la tasa de desempleo usando el último conglomerado por Departamento
) %>%
kable_styling(
    latex_options = c(
        "striped",
        "hold_position"
    )
)
```

Cuadro 8: Estimación de la tasa de desempleo usando el último conglomerado por Departamento

Dpto	Est.	SE	Inter.Inf	Inter.Sup	CV
1	0.082	0.006	0.070	0.093	0.071
2	0.108	0.028	0.054	0.163	0.255
3	0.088	0.010	0.069	0.107	0.111
4	0.029	0.013	0.004	0.054	0.446
5	0.061	0.015	0.031	0.091	0.250
6	0.175	0.030	0.116	0.235	0.173
7	0.038	0.021	-0.004	0.080	0.560
8	0.065	0.023	0.020	0.109	0.352
9	0.086	0.025	0.037	0.135	0.290
10	0.084	0.019	0.046	0.122	0.231
11	0.128	0.029	0.071	0.184	0.225
12	0.060	0.020	0.020	0.099	0.338
13	0.113	0.023	0.069	0.158	0.201
14	0.085	0.022	0.041	0.128	0.263
15	0.126	0.022	0.083	0.168	0.172
16	0.043	0.012	0.019	0.068	0.286
17	0.117	0.026	0.066	0.167	0.222
18	0.087	0.024	0.040	0.133	0.272
19	0.051	0.025	0.001	0.100	0.502

```
svyby(
  formula = ~ pobreza,
  by = ~ dpto,
```

```
  FUN = svymean,
  na.rm = TRUE,
  design = disenio_final,
  vartype = c("ci","se","cv")
) %>%
tibble() %>%
set_names(
  c(
    "Dpto",
    "Est.",
    "SE",
    "Inter.Inf",
    "Inter.Sup",
    "CV"
  )
) %>%
mutate.(
  across.(
    -Dpto,
    .fns = ~ round(.x,3)
  )
) %>%
kbl(
  booktabs = TRUE,
  caption = "Estimación de la tasa de pobreza usando el último conglomerado por Departamento"
) %>%
kable_styling(
    latex_options = c(
        "striped",
        "hold_position"
    )
)
```

```
muestra %>%
  filter.(
    edad >= 25,
    ocupado == 1
  ) %>%
as_survey_design(
  .,
  ids = id_hogar,
  weight = w_nr_boost_clases_calibrados,
  strata = estrato
) %>%
  assign(
    "disenio_final_aux",
    .,
    envir = .GlobalEnv
  )
```

Cuadro 9: Estimación de la tasa de pobreza usando el último conglomerado por Departamento

Dpto	Est.	SE	Inter.Inf	Inter.Sup	CV
1	0.135	0.008	0.118	0.151	0.062
2	0.132	0.032	0.069	0.196	0.244
3	0.057	0.010	0.038	0.076	0.170
4	0.086	0.024	0.039	0.133	0.277
5	0.022	0.010	0.002	0.041	0.453
6	0.105	0.034	0.039	0.171	0.321
7	0.000	0.000	0.000	0.000	NaN
8	0.046	0.021	0.005	0.087	0.459
9	0.066	0.031	0.005	0.126	0.469
10	0.017	0.009	-0.001	0.034	0.527
11	0.076	0.022	0.032	0.119	0.293
12	0.026	0.016	-0.005	0.058	0.617
13	0.183	0.029	0.126	0.241	0.160
14	0.051	0.018	0.015	0.087	0.356
15	0.081	0.022	0.038	0.125	0.274
16	0.022	0.011	0.000	0.044	0.509
17	0.046	0.020	0.007	0.085	0.433
18	0.113	0.029	0.056	0.169	0.257
19	0.164	0.043	0.081	0.248	0.259

```
svyby(
  formula = ~ ingreso,
  by = ~ dpto,
  FUN = svymean,
  na.rm = TRUE,
  design = disenio_final_aux,
  vartype = c("ci","se","cv")
) %>%
tibble() %>%
set_names(
  c(
    "Dpto",
    "Est.",
    "SE",
    "Inter.Inf",
    "Inter.Sup",
    "CV"
  )
) %>%
mutate.(
  across.(
    -Dpto,
    .fns = ~ round(.x,3)
  )
) %>%
```

```
kbl(
  booktabs = TRUE,
  caption = "Estimación del ingreso promedio usando método del último conglomerado por Departa
) %>%
kable_styling(
    latex_options = c(
        "striped",
        "hold_position"
    )
)
```

Cuadro 10: Estimación del ingreso promedio usando método del último conglomerado por Departamento

Dpto	Est.	SE	Inter.Inf	Inter.Sup	CV
1	45048.45	933.709	43218.41	46878.48	0.021
2	29229.10	1796.948	25707.15	32751.06	0.061
3	36416.20	1173.340	34116.50	38715.90	0.032
4	27104.19	1596.957	23974.21	30234.17	0.059
5	32463.64	1226.895	30058.97	34868.31	0.038
6	35559.32	5077.885	25606.85	45511.79	0.143
7	32762.61	2241.106	28370.12	37155.10	0.068
8	33653.30	1916.168	29897.68	37408.92	0.057
9	31133.03	2408.254	26412.94	35853.12	0.077
10	32916.07	1677.659	29627.92	36204.22	0.051
11	31034.92	1822.738	27462.42	34607.42	0.059
12	33695.73	2693.451	28416.66	38974.79	0.080
13	23339.26	1291.590	20807.78	25870.72	0.055
14	31186.69	2038.975	27190.37	35183.00	0.065
15	28152.51	2030.995	24171.83	32133.19	0.072
16	31017.27	1095.901	28869.35	33165.20	0.035
17	36663.34	2872.247	31033.84	42292.84	0.078
18	24517.75	1202.479	22160.94	26874.57	0.049
19	26972.37	2758.660	21565.49	32379.24	0.102

```
svymean(
  ~ pobreza,
  disenio_final,
  vartype = c("ci","se","cv"),
  na.rm = TRUE
)

##              mean     SE
## pobreza 0.092709 0.0044

svyratio(
  ~ desocupado,
  ~ activo,
  disenio_final,
```

```r
  vartype = c("ci","se","cv")
)

## Ratio estimator: svyratio.survey.design2(~desocupado, ~activo, disenio_final,
##     vartype = c("ci", "se", "cv"))
## Ratios=
##                activo
## desocupado 0.08548824
## SEs=
##                activo
## desocupado 0.00375566

svymean(
  ~ ingreso,
  disenio_final_aux,
  vartype = c("ci","se","cv"),
  na.rm = TRUE
)

##          mean     SE
## ingreso 37661 485.62
```

Usando boostrap Rao-Wu

```
svyby(
  formula = ~ pobreza,
  by = ~ dpto,
  FUN = svymean,
  na.rm = TRUE,
  design = disenio_final_rep,
  vartype = c("ci","se","cv")
) %>%
tibble() %>%
set_names(
  c(
    "Dpto",
    "Est.",
    "SE",
    "Inter.Inf",
    "Inter.Sup",
    "CV"
  )
) %>%
mutate.(
  across.(
    -Dpto,
    .fns = ~ round(.x,3)
  )
) %>%
kbl(
  booktabs = TRUE,
  caption = "Estimación de la tasa de pobreza usando Boostrap Rao-Wu por Departamento"
) %>%
kable_styling(
    latex_options = c(
        "striped",
        "hold_position"
    )
)
```

```
muestra %>%
  filter.(
    edad >= 25,
    ocupado == 1
  ) %>%
as_survey_design(
  .,
  ids = id_hogar,
  weight = w_nr_boost_clases_calibrados,
  strata = estrato
) %>%
  assign(
```

Cuadro 11: Estimación de la tasa de pobreza usando Boostrap Rao-Wu por Departamento

Dpto	Est.	SE	Inter.Inf	Inter.Sup	CV
1	0.135	0.009	0.117	0.152	0.065
2	0.132	0.031	0.072	0.193	0.234
3	0.057	0.010	0.037	0.077	0.178
4	0.086	0.024	0.040	0.132	0.276
5	0.022	0.010	0.003	0.041	0.446
6	0.105	0.034	0.038	0.171	0.324
7	0.000	0.000	0.000	0.000	NaN
8	0.046	0.021	0.005	0.087	0.460
9	0.066	0.030	0.006	0.125	0.461
10	0.017	0.009	-0.001	0.034	0.541
11	0.076	0.022	0.032	0.119	0.291
12	0.026	0.017	-0.006	0.059	0.636
13	0.183	0.030	0.125	0.241	0.161
14	0.051	0.018	0.016	0.086	0.349
15	0.081	0.023	0.037	0.126	0.277
16	0.022	0.011	0.001	0.043	0.484
17	0.046	0.020	0.008	0.085	0.424
18	0.113	0.029	0.056	0.170	0.258
19	0.164	0.042	0.083	0.246	0.254

```
    "disenio_final_aux",
    .,
    envir = .GlobalEnv
  )

disenio_final_rep_aux <- survey::as.svrepdesign(
  disenio_final_aux,
  type = "subbootstrap",
  replicates = 500
)


svyby(
  formula = ~ ingreso,
  by = ~ dpto,
  FUN = svymean,
  na.rm = TRUE,
  design = disenio_final_rep_aux,
  vartype = c("ci","se","cv")
) %>%
tibble() %>%
set_names(
  c(
    "Dpto",
    "Est.",
    "SE",
```

```
    "Inter.Inf",
    "Inter.Sup",
    "CV"
  )
) %>%
mutate.(
  across.(
    -Dpto,
    .fns = ~ round(.x,3)
  )
) %>%
kbl(
  booktabs = TRUE,
  caption = "Estimación del ingreso promedio usando Boostrap Rao-Wu por Departamento"
) %>%
kable_styling(
    latex_options = c(
        "striped",
        "hold_position"
    )
)
```

Cuadro 12: Estimación del ingreso promedio usando Boostrap Rao-Wu por Departamento

Dpto	Est.	SE	Inter.Inf	Inter.Sup	CV
1	45048.45	922.935	43239.53	46857.37	0.020
2	29229.10	1841.496	25619.84	32838.37	0.063
3	36416.20	1188.668	34086.46	38745.95	0.033
4	27104.19	1547.395	24071.35	30137.03	0.057
5	32463.64	1260.825	29992.47	34934.82	0.039
6	35559.32	4889.391	25976.29	45142.35	0.137
7	32762.61	2420.853	28017.83	37507.39	0.074
8	33653.30	1883.552	29961.61	37345.00	0.056
9	31133.03	2400.230	26428.67	35837.40	0.077
10	32916.07	1744.285	29497.33	36334.81	0.053
11	31034.92	1785.003	27536.38	34533.46	0.058
12	33695.73	2625.893	28549.07	38842.38	0.078
13	23339.26	1327.943	20736.53	25941.98	0.057
14	31186.69	2065.237	27138.90	35234.48	0.066
15	28152.51	1982.077	24267.71	32037.31	0.070
16	31017.27	1084.403	28891.88	33142.67	0.035
17	36663.34	2930.219	30920.22	42406.47	0.080
18	24517.75	1120.523	22321.57	26713.94	0.046
19	26972.37	2786.421	21511.08	32433.65	0.103

```
svyratio(
  ~ desocupado,
  ~ activo,
```

```
  disenio_final_rep,
  vartype = c("ci","se","cv")
)

## Ratio estimator: svyratio.svyrep.design(~desocupado, ~activo, disenio_final_rep,
##     vartype = c("ci", "se", "cv"))
## Ratios=
##                 activo
## desocupado 0.08548824
## SEs=
##            [,1]
## [1,] 0.0037423

svymean(
  ~ ingreso,
  disenio_final_rep_aux,
  vartype = c("ci","se","cv"),
  na.rm = TRUE
)

##          mean     SE
## ingreso 37661 489.44

svymean(
  ~ pobreza,
  disenio_final_rep,
  vartype = c("ci","se","cv"),
  na.rm = TRUE
)

##             mean     SE
## pobreza 0.092709 0.0045
```

Tras realizar ambas estimaciones por ambos métodos, se ve claramente que las estimaciones puntuales coinciden en su totalidad.

Por otra parte, cuando se analiza la variabilidad de las estimaciones haciendo uso del desvío estándar, puede verse como ambos métodos mantienen (en muy pocos conglomerados) diferencias mínimas, de 0.001 como máximo.

Lo que conlleva a deducír que el comportamiento de los métodos es muy similar.

Lo mismo sucede con los intervalos de confianza, las diferencias son muy pequeñas (aunque sí se puede notar que las mismas son mayores que en SE y la estimación puntual).

Referencias

Libros consultados

[26] Hadley Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

Paquetes de R

[1] Stefan Milton Bache y Hadley Wickham. magrittr: A Forward-Pipe Operator for R. R package version 2.0.1. 2020. URL: https://CRAN.R-project.org/package=magrittr.

[2] Andrew Bray y col. infer: Tidy Statistical Inference. R package version 1.0.0. 2021. URL: https://CRAN.R-project.org/package=infer.

[3] Tianqi Chen y col. xgboost: Extreme Gradient Boosting. R package version 1.5.0.2. 2021. URL: https://github.com/dmlc/xgboost.

[4] Mark Fairbanks. tidytable: Tidy Interface to data.table. R package version 0.6.5. 2021. URL: https://github.com/markfairbanks/tidytable.

[5] Greg Freedman Ellis y Ben Schneider. srvyr: dplyr-Like Syntax for Summary Statistics of Survey Data. R package version 1.1.0. 2021. URL: https://CRAN.R-project.org/package=srvyr.

[6] Lionel Henry y Hadley Wickham. purrr: Functional Programming Tools. R package version 0.3.4. 2020. URL: https://CRAN.R-project.org/package=purrr.

[7] Max Kuhn. modeldata: Data Sets Used Useful for Modeling Packages. R package version 0.1.1. 2021. URL: https://CRAN.R-project.org/package=modeldata.

[8] Max Kuhn. tune: Tidy Tuning Tools. R package version 0.1.6. 2021. URL: https://CRAN.R-project.org/package=tune.

[9] Max Kuhn. workflowsets: Create a Collection of tidymodels Workflows. R package version 0.1.0. 2021. URL: https://CRAN.R-project.org/package=workflowsets.

[10] Max Kuhn y Hannah Frick. dials: Tools for Creating Tuning Parameter Values. R package version 0.0.10. 2021. URL: https://CRAN.R-project.org/package=dials.

[11] Max Kuhn y Davis Vaughan. parsnip: A Common API to Modeling and Analysis Functions. R package version 0.1.7. 2021. URL: https://CRAN.R-project.org/package=parsnip.

[12] Max Kuhn y Davis Vaughan. yardstick: Tidy Characterizations of Model Performance. R package version 0.0.8. 2021. URL: https://CRAN.R-project.org/package=yardstick.

[13] Max Kuhn y Hadley Wickham. recipes: Preprocessing and Feature Engineering Steps for Modeling. R package version 0.1.17. 2021. URL: https://CRAN.R-project.org/package=recipes.

[14] Max Kuhn y Hadley Wickham. Tidymodels: a collection of packages for modeling and machine learning usi 2020. URL: https://www.tidymodels.org.

[15] Max Kuhn y Hadley Wickham. tidymodels: Easily Install and Load the Tidymodels Packages. R package version 0.1.4. 2021. URL: https://CRAN.R-project.org/package=tidymodels.

[16] Kirill Müller. here: A Simpler Way to Find Your Files. R package version 1.0.1. 2020. URL: https://CRAN.R-project.org/package=here.

[17] Kirill Müller y Hadley Wickham. tibble: Simple Data Frames. R package version 3.1.5. 2021. URL: https://CRAN.R-project.org/package=tibble.

[18] R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria, 2021. URL: https://www.R-project.org/.

[19] Tyler Rinker y Dason Kurkiewicz. pacman: Package Management Tool. R package version 0.5.1. 2019. URL: https://github.com/trinker/pacman.

[20] Tyler W. Rinker y Dason Kurkiewicz. pacman: Package Management for R. version 0.5.0. Buffalo, New York, 2018. URL: http://github.com/trinker/pacman.

[21] David Robinson, Alex Hayes y Simon Couch. broom: Convert Statistical Objects into Tidy Tibbles. R package version 0.7.9. 2021. URL: https://CRAN.R-project.org/package=broom.

[22] Julia Silge y col. rsample: General Resampling Infrastructure. R package version 0.1.0. 2021. URL: https://CRAN.R-project.org/package=rsample.

[23] Richard Valliant, Jill A. Dever y Frauke Kreuter. PracTools: Tools for Designing and Weighting Survey Sam R package version 1.2.2. 2020. URL: https://CRAN.R-project.org/package=PracTools.

[24] Davis Vaughan. workflows: Modeling Workflows. R package version 0.2.4. 2021. URL: https://CRAN.R-project.org/package=workflows.

[25] Hadley Wickham. forcats: Tools for Working with Categorical Variables (Factors). R package version 0.5.1. 2021. URL: https://CRAN.R-project.org/package=forcats.

[27] Hadley Wickham. tidyr: Tidy Messy Data. R package version 1.1.4. 2021. URL: https://CRAN.R-project.org/package=tidyr.

[28] Hadley Wickham y Jennifer Bryan. readxl: Read Excel Files. R package version 1.3.1. 2019. URL: https://CRAN.R-project.org/package=readxl.

[29] Hadley Wickham y Dana Seidel. scales: Scale Functions for Visualization. R package version 1.1.1. 2020. URL: https://CRAN.R-project.org/package=scales.

[30] Hadley Wickham y col. dplyr: A Grammar of Data Manipulation. R package version 1.0.7. 2021. URL: https://CRAN.R-project.org/package=dplyr.

[31] Hadley Wickham y col. ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. R package version 3.3.5. 2021. URL: https://CRAN.R-project.org/package=ggplot2.

[32] Hao Zhu. kableExtra: Construct Complex Table with kable and Pipe Syntax. R package version 1.3.4. 2021. URL: https://CRAN.R-project.org/package=kableExtra.