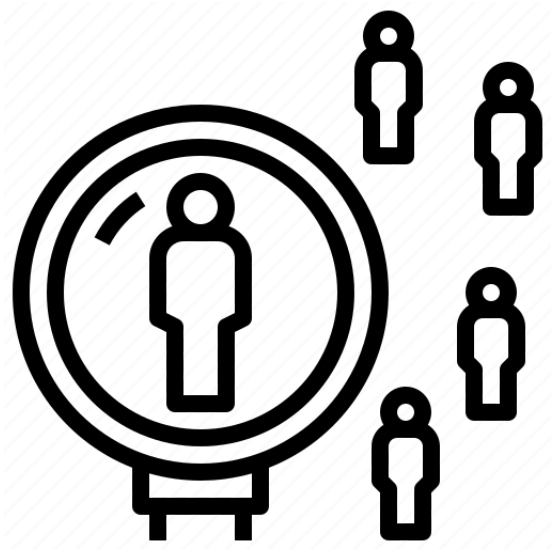
Universidad de la República
Facultad de Ciencias Economicas y de Administración
Licenciatura en Estadística

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 deterministica 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

En base al cuadro, podemos ver que los errores estandar son relativamente chicos, de manera análoga podemos tomar el incremento de varianza respecto a un diseño simple, en base al $deff$, estos son altos debido al diseño en varias etapas de esta encuesta.

Tasa de no respuesta

Desde un enfoque determinístico de la no respuesta, podemos dividir a la muestra en aquellos individuos que respondierón y en aquellos que no lo hicierón.
Es decir, podemos particionar la muestra en aquellos respondentes r_u y no respondentes $s - r_u$.

Una medida de interés, es ver la proporción de respuestas en nuestra muestra, definida como :

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

Para nuestra muestra particular, este dato viene dado 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

De manera análoga, la tasa de respuesta es identica a la anterior, si consideramos dos cifras significativas. Esta estimación puede tener la siguiente interpretación : *el porcentaje de la población que estoy cubriendo una vez que expanda la muestra*, que para este caso particular, **es sumamente bajo**.

Parte 2

Parte a

A continuación se calcuara la tasa de respuesta asumiento un patrón del tipo MAR, es decir, la no respuesta depende de covariables no utilizadas para la estimación.

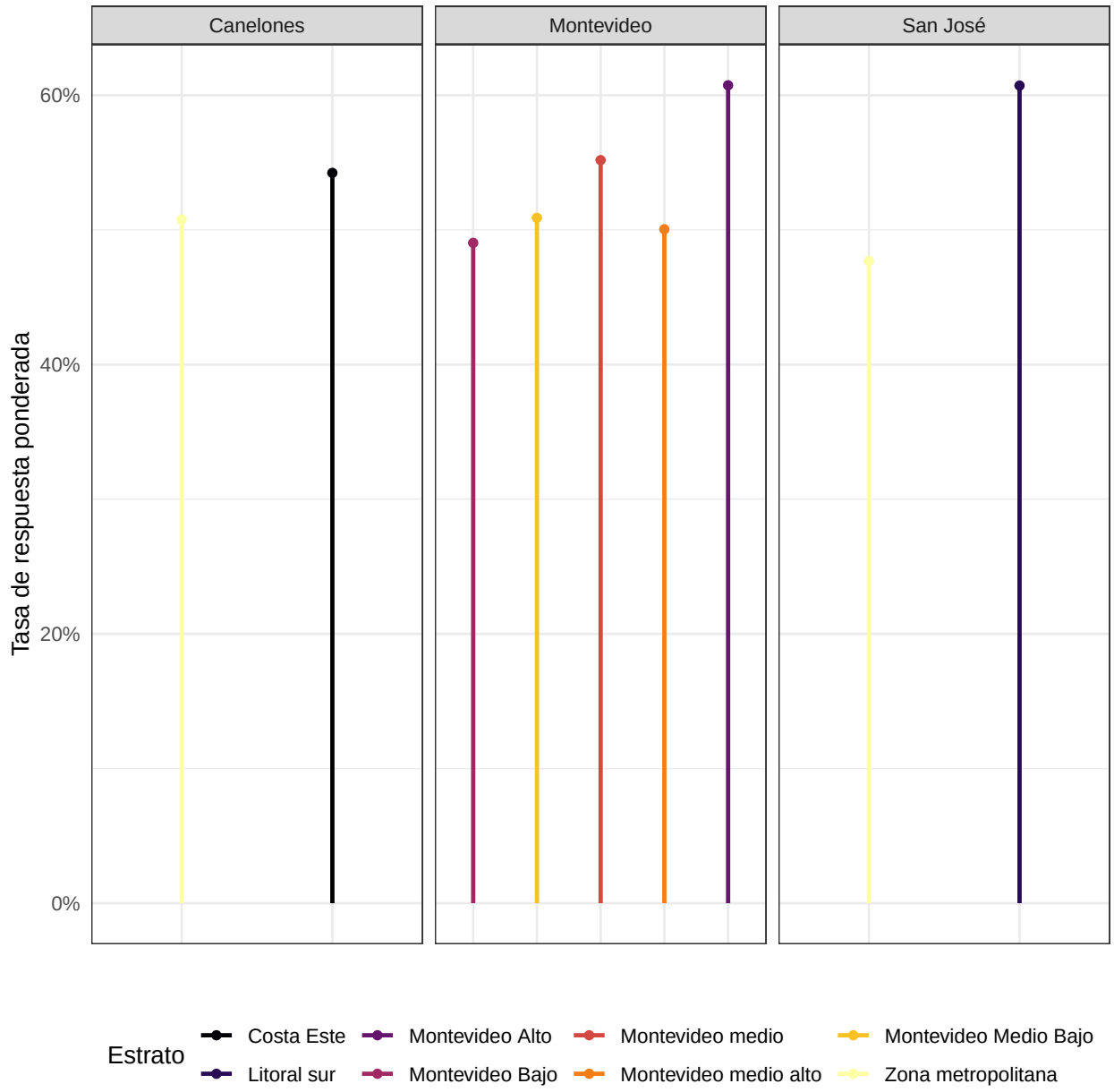
Este tipo de estrategia se basa en crear grupos de individuos con comportamiento similar en la no respuesta, a todos los ponderadores considerados dentro del mismo grupo se le aplicara el mismo ajuste Φ_i a los ponderadores del i-ésimo grupo.

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

A exepción de Montevideo, Canelones y San José, a cada departamento se le imputara la tasa de respuesta de su mismo departamento. Esto puede llegar a hacer diferencias ya que si resumimos esta variable considerando la interacción de estos dos grupos, se pueden encontrar diferentes comportamientos en la 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
    )
```



En base a la gráfica podemos ver diferentes comportamientos 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.

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

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

Una vez hecho esto, continuaremos 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*. Una vez realizado el ajuste, las estimaciones de las variables, además de las referidas a las del error estandar, el coeficiente de variación y efecto diseño, difierón poco respecto a un enfoque deterministico.

Algo que puede llamar la atención es como bajó el ingreso promedio, así como todas las estimaciones referidas a su estadístico.

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

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.

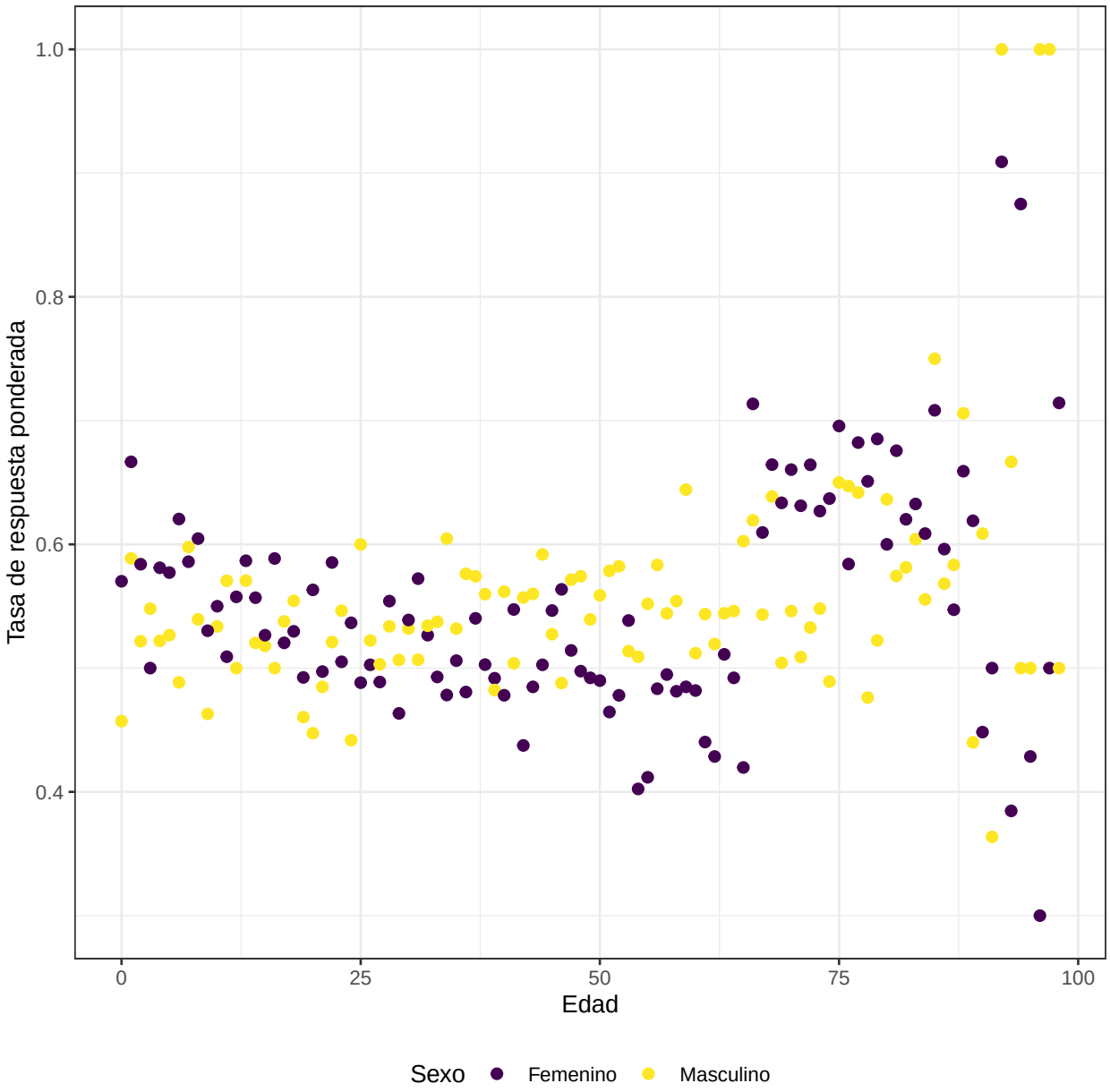
Parte b

Mediante el uso de modelos de Machine Learning de aprendizaje supervizado, estimaremos el *propensity score* conocidos como **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 (predicciónes contra valores observados).

Este tipo de modelos supera a los del tipo de RF, ya que cada nuevo árbol de clasficación 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 se encuentra sin diferencias por sexo, aunque 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
)
```

```
## [18:39:41] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the def
```

```
## 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 % , calculamos 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")
        ),
```

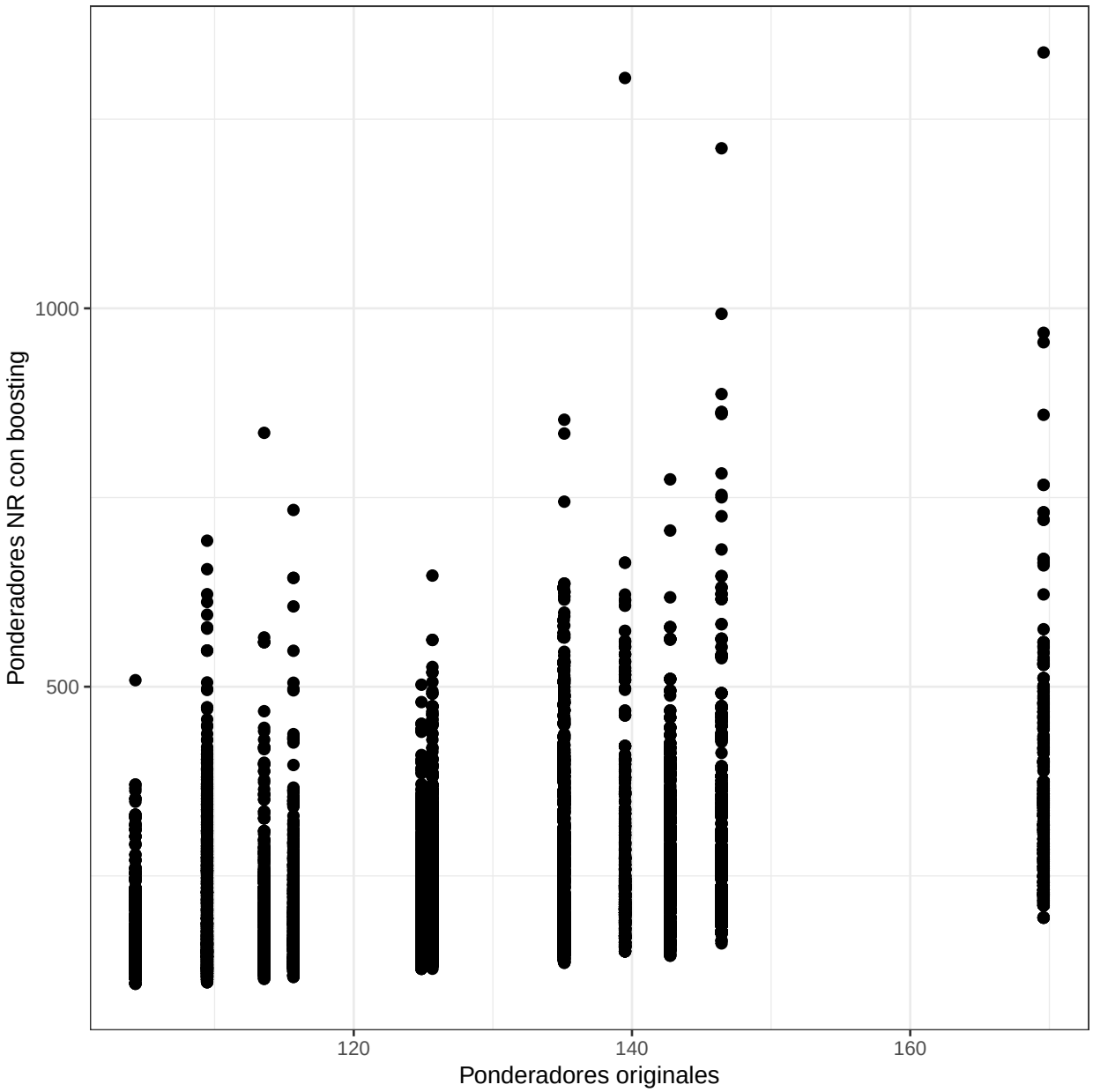
```
    yprom = survey_mean(
        ingreso,
        deff = TRUE,
        vartype = c("se","cv")
    ),
    deffK = deff(
      w_nr_boost,
      type = "kish"
    )
) %>%
assign(
  "est_ponderados_nr_boost",
  .,
  envir = .GlobalEnv
)
```

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 estimaciones difieren poco, además de un leve aumento en los errores estandar 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 basandonos en la regla empírica, al momento, no debemos de preocuparnos. Por último, veremos un gráfico de dispersión de los ponderadores originales, respecto a los recién ajustados.



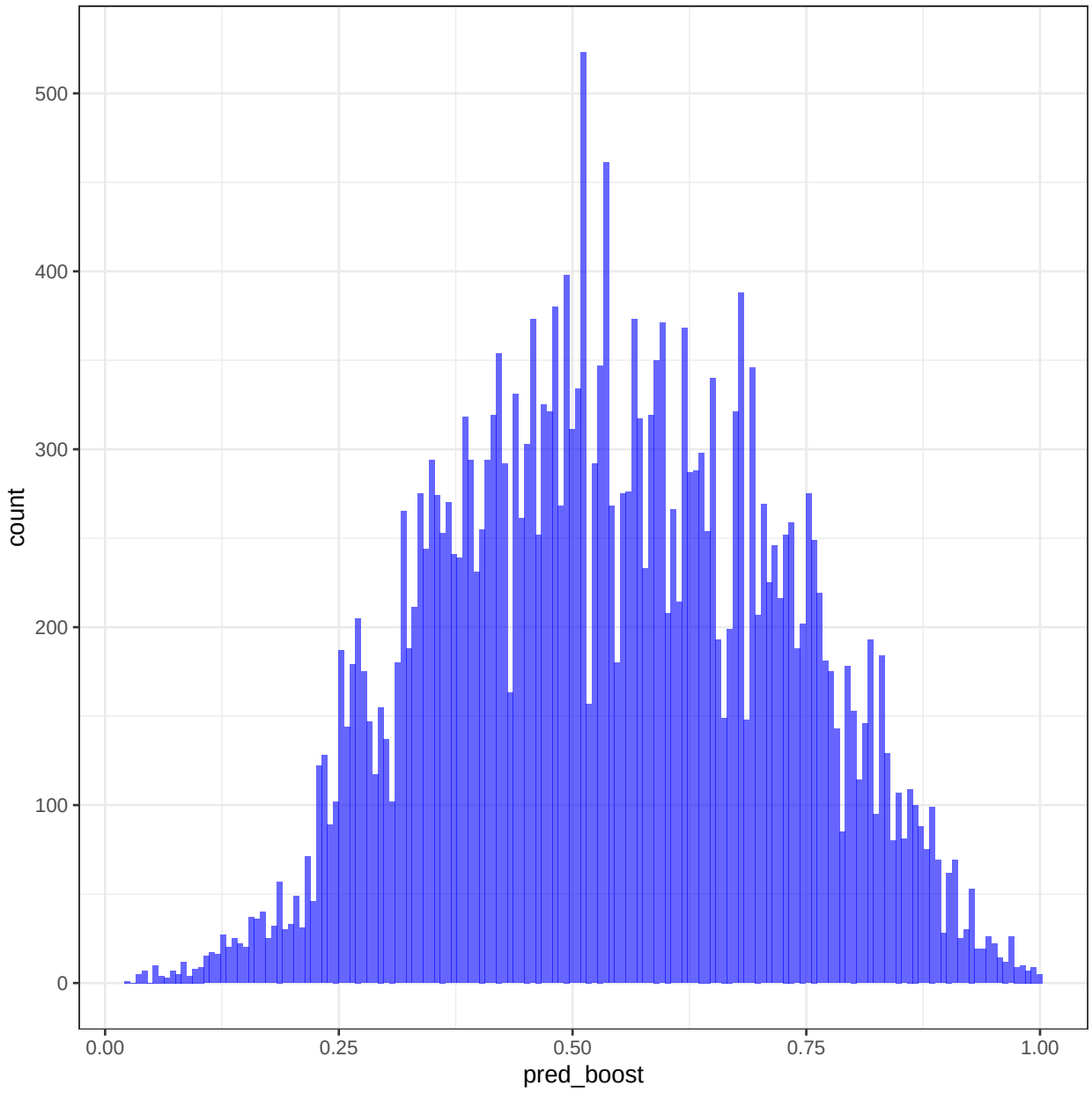
En base a este gráfico puede verse como se aumentarón los ponderadores algunos de ellos de forma considerable,si bien esto puede sonar alarmante los encuestados respondentes tienen que brindar información sobre los que no respondierón, al tener tasas de respuesta bajas, la variación de los ponderadores, tiene que reflejar esa situación.

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

Parte c

Con el mismo modelo de la parte anterior vamos a ajustar la no respuesta médiante propensiones estratificadas, para ello vamos a dividir las propensiones en sus quntiles y luego para cada grupo le asignaremos la propensión mediana.

Primero realizaremos un histograma de estas propensiones :



Podemos ver una distribución simetrica, centrada en valores poco mas grandes que 1/2 aproximandose al valor de la tasa de respuesta.

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

```
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

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]
)
```

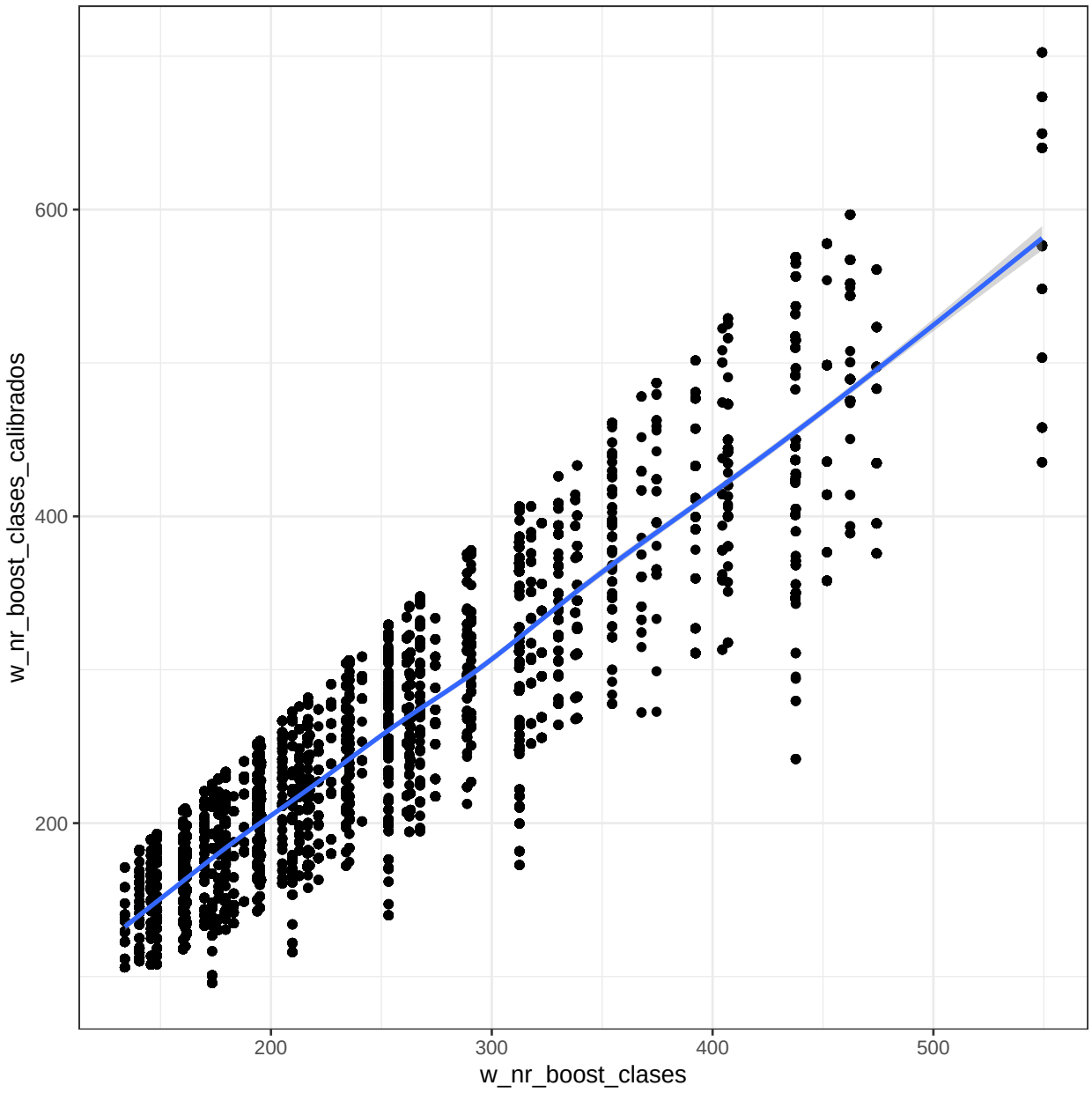
Ahora construiremos los nuevos ponderadores calibrados usando el método de Raking o post-estratificación incompleta, recortamos los mismos y aumentamos el tamaño de iteraciones por defecto.

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 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.


600
400
200
w_nr_boost_clases_calibrados
200
300
400
500
w_nr_boost_clases

Parte 4

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.