Universidad de la República

Facultad de Ciencias Económicas y de Administración

Trabajo final de grado

---

metaSurvey

Paquete de R para el procesamiento de encuestas por muestreo con generación de recetas mediante metaprogramación y estimación de varianzas.

Estudiante:
Mauro Loprete

Tutora:
Dra. Natalia da Silva

Trabajo final de grado presentado como requisito para la obtención del titulo Licenciado en Estadística

"*Blabla*"

Pepito

Abstract

metaSurvey

by Mauro Loprete

El trabajo presenta *metaSurvey*, un paquete de R diseñado para mejorar el procesamiento de encuestas por muestreo y la estimación de parámetros poblacionales. Utiliza meta-programación y técnicas de remuestreo para ofrecer resultados precisos, evaluar la incertidumbre y fomentar la reproducibilidad. A diferencia de otras bibliotecas, *metaSurvey* combina flexibilidad mediante meta-programación con las capacidades de procesamiento de encuestas del paquete *survey*. Los objetivos incluyen proporcionar una herramienta útil, incorporar técnicas de remuestreo para usuarios no expertos, permitir la generación de 'recetas' personalizadas, y fomentar la contribución de la comunidad. Se destaca como alternativa a paquetes propietarios, enfocándose en la transparencia y reproducibilidad para mejorar la confiabilidad de las estimaciones poblacionales.

Acknowledgements

Acá le voy a agradcer a alguien?

Table of contents

List of Figures

List of Tables

List of Abbreviations

LAH **L**ist **A**bbreviations **H**ere
WSF **W**hat (it) **S**tands **F**or

List of Symbols

a	distance	m
P	power	W (J s^1)
ω	angular frequency	rad

Acá va la dedicatoria

Descripción del proyecto

> **Warning**
>
> Este capítulo está en proceso de escritura. Consulte la rama de desarrollo para ver el avance del capítulo

Chapter 1

Introducción

> **Note**
>
> Este capítulo está en proceso de validación. Cualquier comentario es bienvenido

El presente trabajo final de grado tiene como objetivo presentar el desarrollo del paquete *metaSurvey* disponible en R (**ref-rcoreteam2023**). R es un leguaje de programación de código abierto ampliamente utilizado en la comunidad científica para el análisis de datos, estadística y aprendizaje automático, y en general se utiliza el concepto *paquete* para referirse a una colección de funciones, métodos y clases que extienden las funcionalidades de R propuestas por la misma comunidad de usuarios. A lo largo de deste trabajo se abordarán varios conceptos clave tanto para el desarrollo del paquete como para el análisis de encuestas por muestreo.

Actualemente existen varios esfuerzos para facilitar el procesamiento de encuestas, entre ellos existen principalmente dos tipos de paquetes, aquellos que implementan la metodología de inferencia en muestreo de poblaciones finitas como puede ser el paquete *survey* (**ref-lumley2024**), *gustave* (Incluir cita), *vardpoor*, *svrep*, *weights* y aquellos que permiten acceder y manipular encuestas específicas como *ech* (**ref-detomasi2020**), *eph* (**ref-kozlowski2020**), *tidycensus* (**ref-walker2024**), *casen* (Incluir cita) entre otros. Sin embargo, estos ultimos tienen limitaciones en cuanto a la flexibilidad y transparencia del proceso de transformación de los microdatos a indicadores de interés, como puede ser el indice de pobreza, tasas del mercado laboral, ingreso salarial, etc. En general, sus implementaciones son muy sensibles a la estructura y las variables que componen la encuesta, un cambio en la estructura de la encuesta suele implicar una actualización del paquete utilizado para obtener los indicadores en la nueva edicición de la encuesta, lo que resulta poco flexible ante cambios en la estructura, que pueden ser frecuentes en la práctica. Además en las implementaciones actuales, el usuario cuenta con una función de alto nivel que actúa como una caja negra, donde no se permite modificar el código para adaptarlo a sus necesidades o entender cada paso que se realiza para obtener el indicador sin tener que leer el código fuente o la documentación adjunta.

Este tipo de problemas puede verse en *ech* (**ref-detomasi2020**), donde existen funciones para crear variables de mercado laboral, educación o ingresos, pero estas funciones dependen de la existencia de ciertas variables en la encuesta, cuya estructura puede cambiar de una versión a otra de la encuesta. Sin revisar el cuerpo de la función, no se conoce el proceso de construcción de variables. Algo similar ocurre con *eph* (**ref-kozlowski2020**), donde se tienen funciones de alto nivel que no permiten modificar el código para adaptarlo a sus necesidades o entender cada paso que se

realiza para obtener el indicador sin inspeccionar a fondo cómo se construyen las funciones del paquete. Esta inspección del código fuente, como consultar el repositorio de GitHub del paquete o revisar la definición de la función, puede ser una tarea tediosa y no garantiza que el usuario pueda entender el proceso de construcción de variables. Esto se debe a que el código puede ser muy extenso o que el usuario no tenga el conocimiento suficiente para entender el código o se empleen ciertos frameworks que el usuario no conozca, como el uso de las librerías *dplyr* (**ref-wickham2023**) o *tidyr* (**ref-wickham2024**), muy populares en R para el manejo de datos. También puede ser difícil aislar el proceso de manipulación de la encuesta de la implementación específica de la función para manejar la forma de presentación, estructura del objeto a devolver, etc. Un claro ejemplo de esto puede verse en *tidycensus* (**ref-walker2024**), donde existe una función para obtener datos sobre la migración de la comunidad estadounidense, pero en la misma función también se encuentran pasos para mejorar la estructura del conjunto de datos a devolver. En este sentido, el usuario no puede aislar el proceso de recodificación/construcción de variables sobre variables originales y la obtención de datos geográficos y presentación.

Para cientificos sociales, es importante tener en cuenta que el proceso de transformación de los microdatos a indicadores requiere de un conocimiento profundo de la encuesta y en su mayoría no es de conocimiento general. Es de interés obtener información histórica de indicadores y en general es un proceso tedioso y propenso a errores, especialmente si proviene de encuestas donde su estructura y/o forma de preguntar o su codificación puede cambiar con el tiempo. Esto resulta en un proceso extenso y difícil de entender hasta llegar a la construcción de esta serie de indicadores. Muchas veces, diferentes usuarios hacen el mismo proceso de construcción de indicadores de manera independiente y sin compartir el código fuente o la metodología de construcción de indicadores, ya que cada uno utiliza su propio estilo de programación o hasta diferentes paquetes estadísticos, en su mayoría propietarios como SPSS, SAS o STATA, donde si bien el usuario puede compartir la sintaxis para su construcción, esta está ligada al software y depende de que el usuario tenga el software instalado con una licencia activa y pueda correr el código.

En este sentido, es importante que el usuario pueda tener un control total sobre el proceso de transformación de los microdatos a indicadores, ya que esto permite que el usuario pueda validar y entender el proceso de construcción de indicadores, además de brindar una herramienta común libre de estilos de programación y definiendo con simples pasos el proceso de construcción de variables sintéticas, como recodificar variables creando grupos en base a criterios complejos, tratamiento de variables continuas como el ingreso salarial en base a una metodología rigurosa y facil de referenciar en la implementación. Es crucial que este proceso sea transparente y entendible para el usuario. En capitulos posteriores se abordarán ejemplos con los paquetes mencionados anteriormente y se presentará el paquete *metaSurvey* y su implementación de *recetas* para la construcción de indicadores mediante la meta-programación.

Al trabajar con encuestas por muestreo, es importante tener en cuenta la forma en la que se obtuvierón los datos y su proceso generador para poder realizar inferencias sobre la población de interés. En general, obtener estimaciones puntuales de estadísticos de totales, promedios o proporciones es relativamente sencillo, pero puede ser que se reporte una estimación donde no exista un tamaño de muestra suficiente para obtener una estimación confiable y/o que la variabilidad de la estimación sea alta y no sea recomendable su uso. En este sentido, es importante

que el usuario no experto tenga de forma nativa una forma de obtener estimaciones puntuales y sus errores asociados de manera sencilla. Es común utilizar estimaciones puntuales sin tener una medida de incertidumbre o aún peor incluir una estimación del error estándar sin tener en cuenta el diseño muestral correcto, lo que puede llevar a conclusiones erróneas sobre la variabilidad de la estimación. **metaSurvey** permite que el usuario pueda obtener estimaciones puntuales y sus errores asociados de forma nativa y con estos resultados hacer recomendaciones sobre la utilidad y confianza de la estimación mediante coeficientes de variación, intervalos de confianza, tamaño de muestra efectivo, entre otros sin tener que ser un experto en metodología de estimación de varianzas y remuestreo. En capitulos posteriores se abordarán ejemplos con los paquetes mencionados anteriormente y se presentará el paquete *metaSurvey* y su implementación de estimaciones puntuales y sus errores asociados.

El desarrollo de un paquete en R es un proceso que requiere contar con una idea bien formada y los medios para llevarla a cabo es por esto que es importante contar con una metodología de trabajo ordenada, heredada del desarrollo de software convencional ya que para la publicación y difusión del paquete se tiene que cumplir con ciertos estándares de calidad y documentación para que otros usuarios puedan utilizarlo. En este sentido, es importante tener en cuenta que el desarrollo de un paquete en R puede llevar tiempo y esfuerzo, a consecuencia de esto, en el documento se presentarán diferentes conceptos sobre metodología para el desarrollo de paquetes en R y se aboraran ejemplos con la implementación de **metaSurvey**.

En este sentido, *metaSurvey* pretende ser una herramienta relevante para el trabajo con encuestas en ciencias sociales, buscando solucionar las limitaciones anteriormente mencionadas. Todo el proceso de transformación de los microdatos a indicadores se realiza a través de una serie de funciones que permiten al usuario tener un control total y transparente sobre el proceso de transformación de los microdatos a indicadores. Además, *metaSurvey* permite que el usuario pueda realizar el proceso de transformación de los microdatos a indicadores de manera reproducible y transparente. El usuario puede compartir el código de una forma entendible, casi como un "recetario de cocina". El procedimiento aplicado a los datos utilizados para obtener los indicadores se realiza mediante lo que denominamos steps y recipes, conformando así una especie de camino transparente para la construcción de indicadores. Esto permite compartir en forma visual un DAG (Directed Acyclic Graph) que permite visualizar el proceso de construcción de indicadores sin tener que abrir un script de R. En complemento al proceso de creación de variables, *metaSurvey* permite que el usuario pueda obtener estimaciones puntuales y sus errores asociados de manera sencilla y brindar recomendaciones sobre la utilidad de la estimación en el caso de que se cuente con una variabilidad alta en la estimación, en base a recomendaciones a su coeficiente de variación o métricas similares.

El enfoque que permite la flexibilidad a la hora de construir los indicadores es la meta-programación. La meta-programación es un paradigma de programación que permite que un programa pueda modificar su estructura interna en tiempo de ejecución. En R, la meta-programación se realiza a través de las funciones `eval`, `parse`, `substitute`, `do.call` y `quote`, que permiten evaluar y parsear código de manera dinámica. En este sentido, *metaSurvey* utiliza la meta-programación para permitir que el usuario pueda modificar el código que se utiliza para transformar los microdatos a indicadores, teniendo funciones de alto nivel similares a las que se utilizan en el paquete recipes de la librería tidymodels (**ref-kuhn2024**).

En el siguiente capitulo se presentara un marco conceptual básico sobre el muestreo de poblaciones finitas, diferentes paradigmas de programación como puede ser la programación orientada a objetos, programación funcional y la meta-programación y como se utilizan en el desarrollo del paquete. Luego, se ahondara en antecedentes previos tanto en la parte de metodología de estimación de varianzas y paquetes e ideas similares donde se basa el desarrollo del paquete. Finalmente, se presentarán ejemplos de cómo utilizar el paquete *metaSurvey* para construir indicadores de mercado laboral a partir de los microdatos de la **ECH** y para mostrar su flexibilidad, se incluirá un ejemplo con la **EPH**.

Chapter 2

Marco conceptual

> **!** Important
>
> Este capítulo está en borrador. Revise la rama de desarrollo

El objetivo principal de este capítulo es presentar los conceptos básicos que se utilizarán a lo largo de este documento, en específico en las secciones de antecedentes y metodología. En primer lugar se presentara un marco básico de inferencia en muestreo de poblaciones finitas para luego presentar diferentes métodos de estimacón de parámetros poblacionales y sus respectivos errores estándar. Se hará una primera introducción a diseños sencillos y se propondran diferentes estimadores y se hará mención a diferencia con los diseños complejos, situación común en Encuestas Socioeconomícas. Luego, se presentarán los conceptos básicos de la programación funcional y orientada a objetos en R para luego enfocarnos en la meta-programación. Finalmente, se presentará un breve resumen de cómo crear un paquete en R, los componentes mínimos para su publicación en **CRAN** (repositorio donde se encuentran disponibles versiones estables de diferentes paquetes de R), y las herramientas que se pueden utilizar para su desarrollo.

2.1 Inferencia en muestreo de poblaciones finitas

Como fue mencionado anteriormente las encuestas por muestreo son la principal fuente de información para la construcción de indicdores sociodemográficos y economicos, en este sentido, es importante tener en cuenta un marco teórico para realizar inferencias. Es sumamente sencillo obtener estimaciones puntuales de estadísticos usuales aunque es importante considerar la variabilidad de los estimadores, tanto para poder realizar un proceso de inferencia completo así como también para poder cuantificar la confiabilidad de la estimación. A continuación, se definen los conceptos básicos de inferencia en muestreo de poblaciones finitas como son el diseño muestral, probabilidades de inclusión basadas en el diseño, estimadores de Horvitz-Thompson **HT**, ponderación, medidas de incertidumbre y errores estándar basados en (**ref-suxe4rndal2003**).

2.1.1 Diseño muestral

El concepto de diseño muestral refiere al mecanismo mediante el cual se selecciona una muestra donde aquí se determinan propiedades estadísticas claves como puede ser la distribución en el muestreo, valores esperados y varianzas de estimadores poblacionales. En diseños sencillos es posible calcular esta función o encontrar una

expresión análitica con facilidad mientras que en diseños mas complejos como pueden ser los multietapicos es necesario abordar el problema de otra forma y asumir ciertas hipoesis para poder construir probabilidades de inclusión tanto de primer órden como segundo órden.

La definición matematica se basa en que dado un universo U de N elementos (puede ser conocido o no) $\{u_1, u_2, \cdots, u_N\}$ y se considera un conjunto de tamaño n de elementos de U que se denota como $s = \{u_1, u_2, \cdots, u_n\}$ al cual comunmente denominamos **muestra**, el diseño muestral puede definirse de la siguiente forma:

$$Pr(S = s) = p(s)$$

Realizando un poco de inspección en la definición anterior se puede observar que el diseño muestral es una función de probabilidad que asigna una probabilidad a cada subconjunto de U de tamaño n. En este sentido, es posible definir diferentes tipos de diseño, entre ellos los mas comunes:.

- **Diseño Aleatorio Simple (SI)**

El diseño aleatorio simple es el diseño más sencillo y se define de la siguiente forma:

$$p(s) = \frac{1}{\binom{N}{n}}$$

Donde $\binom{N}{n}$ es el número de subconjuntos posibles de U de tamaño n.

- **Diseño Bernoulli (BE)**

El (**BE**) es un diseño sencillo que se utiliza cuando se desea seleccionar una muestra de un universo de tamaño N además de considerar una una probabilidad de inclusión π para cada elemento de U. Se define el diseño Bernoulli de la siguiente forma:

$$p(s) = \underbrace{\pi \times \pi \times \cdots \times \pi}_{n_s} \times \underbrace{(1-\pi) \times (1-\pi) \times \cdots \times (1-\pi)}_{N-n_s} = \pi^{n_s}(1-\pi)^{N-n_s}$$

Una diferencia fundamental entre el diseño (BE) y el diseño **SI** es que en el **BE** el tamaño de muestra es aleatorio y su distribución es binomial, mientras que en el diseño SI el tamaño de muestra es fijo.

- **Diseño Estratificado (ST)**

El diseño estratificado es un diseño que se utiliza cuando se desea seleccionar una muestra de tamaño n de un universo de tamaño N donde además se quiere dividir el universo en H estratos $U_1, U_2, \cdots, U_H$. Dentro de cada estrato se selecciona una muestra de tamaño n_h y se define el diseño estratificado de la siguiente forma:

$$p(s) = \prod_{l=1}^{H} p(s_H)$$

En cada estrato se puede utilizar un diseño diferente pero en general se utiliza el diseño **SI**, mas conocido **STSI** (Stratified Simple Random Sampling). En este caso cada $p_h(s_h)$ es el diseño aleatorio simple en el estrato h.

2.1.2 Probabilidades de inclusión y estimador de Horvitz-Thompson

Una vez definido el concepto de diseño muestral es posible definir la probabilidad de que un elemento de la población sea seleccionado en la muestra, esta probabilidad se conoce como probabilidad de inclusión y se define de la siguiente forma:

- **Probabilidad de inclusión de primer orden**

$$\pi_k = Pr(u_k \in s) = Pr(I_k = 1)$$

Donde I_k es una variable aleatoria que toma el valor de 1 si el elemento u_k es seleccionado en la muestra y 0 en caso contrario. Definir estas variables indicadoras son de utilizada para entender el comportamiento de los estimadores bajo el diseño muestral y nos permite definir los estimaodres en U y no en S. Es claro que $I_k \sim Bernoulli(\pi_k)$ y $E(I_k) = Pr(I_k) = \pi_k$.

Esta probabilidad es importante ya que es la la base para la construcción de estimadores insesgados y eficientes, en este sentido, es posible definir el estimador de Horvitz-Thompson (**HT**) para estimar un total $t = \sum_U t_k$ de la siguiente forma:

$$\hat{t}_y = \sum_{k=1}^{N} \frac{y_k}{\pi_k} \times I_k$$

Este estimador es propuesto por Horvitz y Thompson en 1952 y es un estimador insesgado en el diseño, en el sentido de que $E(\hat{t}_y) = t$ y es eficiente en el sentido de que $Var(\hat{t}_y)$ es el menor posible entre los estimadores insesgados. Este estimador es muy utilizado en la práctica y es la base para la construcción de otros estadísticos,como medias, proporciones, varianzas, entre otros. Para mas detalles sobre las propiedades de Horvitz-Thompson (**HT**) se puede consultar en (**ref-suxe4rndal2003**) y (**ref-horvitz1952**).

2.1.3 Ponderación basada en el diseño y estimadores más comunes

En general es utilizado el concepto de ponderador para realizar estimaciones de totales, medias, proporciones, varianzas, entre otros. En este sentido, es posible definir el ponderador inducido por el diseño muestral de la siguiente forma:

$$w_k = \frac{1}{\pi_k}$$

Este ponderador puede interpretarse como el número individuos que representra el individuo k en la población. Este valor es el que comunmente se publica junto a los microdatos y el estandar en los diferentes softwares para procesar encuestas. Junto al estimador de un total es posible definir el estimador de un promedio, proporción o razón en el contexto de la $-expansión.

Estimador de un promedio

$$\hat{\bar{y}} = \frac{\sum_{k=1}^{N} w_k I_k y_k}{\sum_{k=1}^{N} w_k I_k}$$

Este estimador puede ser utilizados en encuestas de hogares, donde se desea estimar el ingreso promedio de los hogares de una región de forma anual, o mensual.

Estimador de una proporción

$$\hat{p} = \frac{\sum_{k=1}^{N} I_k w_k y_k}{\sum_{k=1}^{N} w_k I_k} = \frac{\sum_{k=1}^{N} I_k w_k y_k}{\hat{N}}$$

Puede ser de interés estimar la proporción de hogares que tienen acceso a internet en una región, en este caso se puede utilizar el estimador de proporción.

Estimador de una razón

Se quiere estimar la razón $R = \frac{\sum_{k=1}^{N} y_k}{\sum_{k=1}^{N} z_k}$. En este caso se puede definir el estimador de la razón de la siguiente forma:

$$\hat{R} = \frac{\sum_{k=1}^{N} w_k y_k}{\sum_{k=1}^{N} w_k z_k} = \frac{\sum_{k=1}^{N} w_k y_k}{\hat{N}}$$

El estimador de razón es utilizado para constuir variables de mercado de trabajo como la tasa de desempleo, tasa de ocupación, entre otros.

Inferencia sobre el tamaño de la población

Una vez definidos los estimadores, podemos ver que los estimaodres de medias y proporciones son un caso particular del estimador de razón. Un detalle no menor es que asumimos N fijo pero desconocido, por esto al realizar proporciones se ajusta el total sobre un estimador del tamaño de la población:

$$\hat{N} = \sum_{k=1}^{N} I_k w_k$$

Existen diseños denominados **auto-ponderados** donde por definición $\sum_{k=1}^{N} w_k = N$, en este caso particular el estimador de medidas y proporciones es un caso parciular del estimador de total, ya que el estadístico puede definirse de la siguiente forma:

$$\hat{\bar{y}}_s = \frac{\sum_{k=1}^{N} I_k w_k y_k}{\sum_{k=1}^{N} w_k I_k} = \frac{\sum_{k=1}^{N} I_k w_k y_k}{N} = \frac{1}{N} \times \sum_{k=1}^{N} I_k w_k y_k = a \times \hat{t}_y$$

2.1.4 Medidas de incertidumbre y errores estándar

Se puede medir la variabilidad de los estimadores y calcular su varianza. Esto es útil para entender cuán confiables son estos estimadores. Veamos cómo se calcula la varianza de diferentes tipos de estimadores, como el total, promedio, proporción o razón.

2.1.4.1 Momentos muestrales y estimadores de varianza

Para un estadístico θ, su varianza bajo un diseño muestral $p(s)$ se define como:

$$V(\hat{\theta}) = E((\theta - E(\hat{\theta}))^2) = \sum_{s \in S} p(s) \left(\hat{\theta}_s - E(\hat{\theta}_s)\right)$$

La forma de calcular la varianza depende del estimador $\hat{\theta}$. Por ejemplo, para el estimador de varianza de un total, se utiliza la siguiente fórmula:

$$V(\hat{t}_y) = \sum_U V(I_k \times y_k \times w_k) + \sum_U \sum_{k \neq l} Cov(I_k \times y_k \times w_k, I_l \times y_l \times w_l)$$

Después de simplificar, obtenemos:

$$V(\hat{t}_y) = \sum_U V(I_k) \times w_k \times y_k^2 + \sum_U \sum_{k \neq l} Cov(I_k, I_l) \times y_k \times w_k \times y_l \times w_l$$

Donde definimos las siguientes identidades para simplificar cálculos:

$$Cov(I_k, I_l) = \Delta_{kl} = \pi_{kl} - \pi_k \times \pi_l$$

$$\breve{y}_k = y_k \times w_k$$

$$\breve{\Delta}_{kl} = \Delta_{kl} \times \frac{1}{\pi_{kl}} = \Delta_{kl} \times w_{kl}$$

Una vez definida la varianza del estimador, necesitamos estimar su varianza. Para esto, utilizamos la técnica de π-expansión. Después de algunas manipulaciones algebraicas, obtenemos la varianza del estimador:

$$V(\hat{t}_y) = \sum_U \breve{y}_k^2 + \sum_U \sum_{k \neq l} \Delta_{kl} \times \breve{y}_k \times \breve{y}_l = \sum_U \sum \Delta_{kl} \times \breve{y}_k \times \breve{y}_l$$

Podemos verificar que este estimador de varianza es insesgado con la definiciones de $E(I_k I_l)$ y tomando esperanzas. Es decir, se verifica que $E(\hat{V}(\hat{t}_y)) = V(\hat{t}_y)$. Al ser un estimador insesgado, su eficiencia depende del diseño muestral y de la varianza de los ponderadores, es decir, de la varianza de las probabilidades de inclusión. En algunos casos es donde entra en juego dividir grupos heterogéneos en estratos o realizar muestreos en varias etapas.

Para el caso de un estimador de un promedio, la varianza se define de la siguiente forma:

$$V(\hat{\bar{y}}) = \frac{1}{N^2} \times \sum_U \sum_{k \neq l} \Delta_{kl} \times \breve{y}_k \times \breve{y}_l$$

Esto es válido en el caso de contar con un tamaño de población conocido, en otro caso el estimador de la media no es un estimador lineal y para calcular su varianza deben optar por métodos de estimación de varianzas más complejos como el de linealización de Taylor.

Es importante considerar que en esta sección se presenta un caso ideal donde la muestra es obtenida de un listado **perfecto** de la población objetivo denominado **marco de muestreo**. En la práctica, el marco de muestreo es imperfecto y se debe considerar la no respuesta, la cobertura y la falta de actualización del marco de muestreo. En general para la publicación de microdatos se publican ponderadores los ponderadores originales sometidos a un proceso de **calibración** que ajusta los ponderadores para que los totales de la muestra coincidan con los totales de la población en algunas variables de control y permita mejorar el sesgo de no respuesta. El objetivo principal es crear ponderadores calibrados lo mas cercano posible a los ponderadores originales, de forma que si los ponderadores originales son insesgados, los ponderadores calibrados seran proximos a ser insesgados.

Además de contar con ponderadores calibrados para calcular varianzas de los estimaodres **HT** es necesario contar con las probabilidades de inclusión de segundo orden, donde dependiendo del diseño este puede ser imposibles de calcular o pueden existir probabilidades de inclusión de segundo orden que sean cero. Por esto es necesario contar con diferentes estrategias de estimación de varianzas como pueden ser el Método del Ultimo Conglomerado, el Método de Jackknife, el Método de Bootstrap, entre otros.

En resumen, para realizar estimaciones puntuales ya sean totales, medias, proporciones o razones simplemente debemos ponderar los datos con los estadísticos anteriormente mencionadas pero para realizar un proceso de inferencia completo se requiere calcular sus errores estándar, construir intervalos de confianza y/o poder medir estabilidad de nuestros resultados. En este sentido, es importante tener al alcance herramientas que permitan realizar este tipo de cálculos, ya que si bien en diferentes softwares estadísiticos junto a la estimación puntual se presentan los errores estándar pero por defecto se asumen diseños sencillos como por ejemplo, el diseño **BE** donde la probabilidad de inclusión de segundo orden es sencilla de calcular y unicamente es necesario las probabilidades de inclusión de primer orden para computar estimadores del error estándar, **siendo un valor completamente erroneo.**

2.2 Desarrollo de paquetes en R

R al ser un lenguaje de código abierto y además cuenta con una gran comunidad de usuarios, en diferentes áreas de investigación, ha permitido que se desarrollen una gran cantidad de paquetes que permiten realizar diferentes tareas de análisis de datos, visualización, modelado, entre otros. En este sentido, el desarrollo de paquetes en R es una tarea que se ha vuelto muy común entre los usuarios de R, ya que permite compartir código, documentación y datos de manera sencilla.

Para casi cualquier disciplina científica o en la industria se puede encontrar una comunidad de usuarios que desarrollan paquetes en R, en este sentido, el desarrollo de paquetes en R es una tarea que se ha vuelto muy común entre los usuarios de R y es muy sencillo de realizar. A continuación, se presentan los conceptos básicos para el desarrollo de paquetes en R.

2.2.1 ¿Por qué desarrollar un paquete en R?

Desarrollar un paquete en R tiene varias ventajas, entre las cuales se pueden mencionar las siguientes:

- **Reutilización de código**: Es importante tener en cuenta que existe una comunidad que hace cosas similares a las que uno hace, por lo que es posible que alguien ya haya escrito una función que uno necesita. Por lo tanto, siempre es buena buscar si existe algún paquete que ya tenga las funcionalidades que se requieren.
- **Compartir código**: La comunidad de R es muy activa y siempre está dispuesta a compartir código, por esta razón es que se mantienen en constante desarrollo de paquetes.
- **Colaboración**: El trabajo colaborativo es esencial en el desarrollo de paquetes en R, ya que permite que diferentes personas puedan aportar con nuevas funcionalidades, correcciones de errores, entre otros.

2.2.2 Elementos básicos de un paquete en R

Para que nuestro conjunto de funciones, datos y documentación sea considerado un paquete en R, es necesario que cumpla con ciertos requisitos mínimos. A continuación, se presentan los componentes mínimos que debe tener un paquete en R para ser publicado en CRAN.

- **Directorio**: Un paquete en R debe estar contenido en un directorio que contenga al menos los siguientes archivos y directorios:
  - `R/`: Directorio que contiene los archivos con las funciones que se desean incluir en el paquete.
  - `man/`: Directorio que contiene los archivos con la documentación de las funciones que se encuentran en el directorio `R/`. En general se utiliza *Roxygen2* (**ref-roxygen2**) para generar la documentación de las funciones.
  - `DESCRIPTION`: Archivo que contiene la descripción del paquete, incluyendo el nombre, versión, descripción, autor, entre otros.
  - `NAMESPACE`: Archivo que contiene la información sobre las funciones que se exportan y las dependencias del paquete.
  - `LICENSE`: Archivo que contiene la licencia bajo la cual se distribuye el paquete.
  - `README.md`: Archivo que contiene información general sobre el paquete.
- **Documentación**: La documentación de las funciones es un componente esencial de un paquete en R, ya que permite que los usuarios puedan entender el funcionamiento de las funciones que se encuentran en el paquete. La documentación de las funciones se realiza utilizando el sistema de documentación de R, que se basa en el uso de comentarios en el código fuente de las funciones.
- **Pruebas**: Es importante que el paquete tenga pruebas que permitan verificar que las funciones se comportan de la manera esperada. Las pruebas se realizan utilizando el paquete *testthat* (**ref-wickham2011**) que permite realizar pruebas unitarias.
- **Control de versiones**: Es importante que el paquete tenga un sistema de control de versiones que permita llevar un registro de los cambios que se realizan en el paquete. El sistema de control de versiones más utilizado en la comunidad de R es `git`.

- **Licencia**: Es importante que el paquete tenga una licencia que permita a los usuarios utilizar, modificar y distribuir el paquete. La licencia más utilizada en la comunidad de R es la licencia MIT.

El proceso de subir un paquete a CRAN es un proceso que puede ser tedioso, ya que se deben cumplir con ciertos requisitos que son revisados por los mantenedores de CRAN, no es trivial y puede tomar tiempo, sin embargo, es un proceso que vale la pena ya que permite que el paquete sea utilizado por una gran cantidad de usuarios.

El proceso de chequeo fue automatizado con github actions, por lo que cada vez que se realiza un cambio en el repositorio, se ejecutan los chequeos de CRAN y se notifica si el paquete cumple con los requisitos para ser publicado en caso de que no cumpla con los requisitos se notifica el error y no puede ser incluido en la rama principal del repositorio hasta que se corrija el error.

Todo el proceso y código fuente del paquete se encuentra disponible en el repositorio de github del paquete. En el caso que este interesado en colaborar con el desarrollo del paquete puede consultar la guía de contribución

2.3 Paradigmas de programación en R

R es un lenguaje de programación que permite realizar programación funcional y orientada a objetos, lo que permite que los usuarios puedan utilizar diferentes paradigmas de programación para resolver problemas. A continuación, se presentan los conceptos básicos de la programación funcional y orientada a objetos en R.

2.3.1 Programación funcional

La programación funcional es un paradigma de programación que se basa en el uso de funciones para resolver problemas. En R, las funciones son objetos de primera clase, lo que significa que se pueden utilizar como argumentos de otras funciones, se pueden asignar a variables, entre otros (**ref-wickham2019**). A continuación, se presentan los conceptos básicos de la programación funcional en R.

- **Funciones de orden superior**: En R, las funciones de orden superior son funciones que toman como argumento una o más funciones y/o retornan una función. Un ejemplo de una función de orden superior en R es la función `lapply` que toma como argumento una lista y una función y retorna una lista con los resultados de aplicar la función a cada elemento de la lista.
- **Funciones anónimas**: En R, las funciones anónimas son funciones que no tienen nombre y se crean utilizando la función `function`. Un ejemplo de una función anónima en R es la función `function(x) x^2` que toma como argumento `x` y retorna `x^2`.
- **Funciones puras**: En R, las funciones puras son funciones que no tienen efectos secundarios y retornan el mismo resultado para los mismos argumentos. Un ejemplo de una función pura en R es la función `sqrt` que toma como argumento un número y retorna la raíz cuadrada de ese número.

Este paradigma de programación es muy útil para realizar análisis de datos, ya que permite que los usuarios puedan utilizar funciones para realizar operaciones sobre los datos de manera sencilla y eficiente, dentro de metaSurvey no existe una presencia

fuerte de programación funcional, sin embargo, se utilizan algunas funciones de orden superior para realizar operaciones sobre los datos.

2.3.2 Programación orientada a objetos

La programación orientada a objetos es un paradigma de programación que se basa en el uso de objetos para resolver problemas. En R, los objetos son instancias de clases que tienen atributos y métodos (**ref-mailund2017**; **ref-wickham2019**). A continuación, se presentan los conceptos básicos de la programación orientada a objetos en R.

- **Clases y objetos**: En R, las clases son plantillas que definen la estructura y el comportamiento de los objetos y los objetos son instancias de clases. En R, las clases se definen utilizando la función `setClass` y los objetos se crean utilizando la función `new`.
- **Atributos y métodos**: En R, los atributos son variables que almacenan información sobre el estado de un objeto y los métodos son funciones que permiten modificar el estado de un objeto. En R, los atributos se definen utilizando la función `setClass` y los métodos se definen utilizando la función `setMethod`.

Dentro de metaSurvey se utiliza la programación orientada a objetos para definir las clases de los objetos que se utilizan para representar los datos de las encuestas mediante una creación de una clase especifica llamada `Survey` que permite además de almacenar los datos de la encuesta añadir atributos y métodos que permiten realizar operaciones sobre los datos de manera sencilla y eficiente.

De forma similar se modelan las clases `Step`, `Recipe` y `Workflow` elementos cruciales en el ecosistema de metaSurvey donde se definen los pasos de preprocesamiento, recetas de preprocesamiento y flujos de trabajo respectivamente. En este caso particular se utiliza el paquete *R6* (**ref-chang2022**) que permite definir clases de manera sencilla y eficiente además de permitir la herencia de clases y la definición de métodos y atributos de manera sencilla.

2.3.3 Meta-programación

La meta-programación es un paradigma de programación que se basa en el uso de código para manipular código (**ref-thomasmailund2017**; **ref-wickham2019**) . En R, la meta-programación se realiza utilizando el sistema de metaprogramación de R que se basa en el uso de expresiones, llamadas y funciones. A continuación, se presentan los conceptos básicos de la meta-programación en R.

- **Expresiones**: En R, las expresiones son objetos que representan código y se crean utilizando la función `quote`. Un ejemplo de una expresión en R es la expresión `quote(x + y)` que representa el código `x + y`.
- **Llamadas**: En R, las llamadas son objetos que representan la aplicación de una función a sus argumentos y se crean utilizando la función `call`. Un ejemplo de una llamada en R es la llamada `call("sum", 1, 2, 3)` que representa la aplicación de la función `sum` a los argumentos `1`, `2` y `3`.
- **Funciones**: En R, las funciones son objetos que representan código y se crean utilizando la función `function`. Un ejemplo de una función en R es la función `function(x, y) x + y` que representa el código `x + y`.

Chapter 3

Antecedentes

> **Warning**
>
> Este capítulo está en proceso de escritura. Consulte la rama de desarrollo para ver el avance del capítulo

Chapter 4

Metodología

> **Warning**
>
> Este capítulo está en proceso de escritura. Consulte la rama de desarrollo para ver el avance del capítulo

Chapter 5

Resultados

> **Warning**
>
> Este capítulo está en proceso de escritura. Consulte la rama de desarrollo para ver el avance del capítulo

Acá va la viñeta Use recipes

5.1 ECH

```
library(magrittr)

metaSurvey::set_engine("data.table")
```

```
Engine: data.table
```

```
ech_meta = metaSurvey::load_survey(
  path = metaSurvey::load_survey_example("ech_2018.csv"),
  svy_type = "ech",
  svy_edition = "2018",
  svy_weight = "pesoano"
)

ech_meta_steps = ech_meta %>%
  metaSurvey::step_recode(
    "pea",
    pobpcoac %in% 2:5 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "pet",
    pobpcoac != 1 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "po",
    pobpcoac == 2 ~ 1,
    .default = 0
  ) %>%
```

```
  metaSurvey::step_recode(
    "pd",
    pobpcoac %in% 3:5 ~ 1,
    .default = 0
  )
```

```
metaSurvey::view_graph(ech_meta_steps)
```

5.2 EAII

```
svy_example = metaSurvey::load_survey(
    svy_type = "eaii",
    svy_edition = "2019-2021",
    svy_weight = "w_trans",
    input = metaSurvey::load_survey_example("2019-2021.csv"),
    dec = ","
)

# as.data.frame(svy_example)
# as.tibble(svy_example)

new_svy = svy_example %>%
    metaSurvey::step_recode(
        new_var = "realiza_innovacion",
        B1_1_1 == 1 ~ 1,
        B1_2_1 == 1 ~ 1,
        B1_3_1 == 1 ~ 1,
        B1_4_1 == 1 ~ 1,
        B1_5_1 == 1 ~ 1,
        B1_6_1 == 1 ~ 1,
        B1_7_1 == 1 ~ 1,
        B1_8_1 == 1 ~ 1,
        B1_9_1 == 1 ~ 1,
        .default = 0
    ) %>%
    metaSurvey::step_recode(
        new_var = "sector",
        data.table::between(Division, 10, 33) ~ "Industria",
        data.table::between(Division, 34, 99) ~ "Servicios",
        Division == "C1" ~ "Industria",
        Division == "C2" ~ "Servicios",
        Division == "E1" ~ "Servicios"
    ) %>%
    metaSurvey::step_recode(
        new_var = "innovativa",
        E1_1_1 == 1 ~ 1,
        E1_2_1 == 1 ~ 1,
        .default = 0
    ) %>%
```

```
    metaSurvey::step_recode(
        new_var = "tipo_actividad",
        B1_1_1 == 1 ~ "I + D Interna",
        B1_2_1 == 1 ~ "I + D Externa",
        B1_3_1 == 1 ~ "Bienes de Capital",
        B1_4_1 == 1 ~ "Software",
        B1_5_1 == 1 ~ "Propiedad Intelectual",
        B1_6_1 == 1 ~ "Ingeniería",
        B1_7_1 == 1 ~ "Capacitación",
        B1_8_1 == 1 ~ "Marketing",
        B1_9_1 == 1 ~ "Gestión",
        .default = "Otra"
    ) %>%
    metaSurvey::step_recode(
        new_var = "tipo_innovacion",
        E1_1_1 == 1 ~ "Producto",
        E1_2_1 == 1 ~ "Proceso",
        .default = "Otra"
    ) %>%
    metaSurvey::step_recode(
        new_var = "cant_traba_tramo",
        data.table::between(IG_4_1_3, 0, 4) ~ "1",
        data.table::between(IG_4_1_3, 5, 19) ~ "2",
        data.table::between(IG_4_1_3, 20, 99) ~ "3",
        IG_4_1_3 > 99 ~ "4"
    ) %>%
    metaSurvey::step_recode(
        new_var = "ingreso_vta_pesos",
        data.table::between(IG_5_1_1_3, 0, 9942787) ~ "1",
        data.table::between(IG_5_1_1_3, 9942788, 49713934) ~ "2", # nolint
        data.table::between(IG_5_1_1_3, 49713935, 372854507) ~ "3", # nolint
        IG_5_1_1_3 > 372854507 ~ "4"
    ) %>%
    metaSurvey::step_recode(
        new_var = "tamanio",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "1" ~ "Pequenias",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "2" ~ "Pequenias",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "1" ~ "Pequenias",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "2" ~ "Pequenias",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "3" ~ "Medianas",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "2" ~ "Medianas",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "1" ~ "Medianas",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "3" ~ "Medianas",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "3" ~ "Medianas",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "4" ~ "Grandes",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "3" ~ "Grandes",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "2" ~ "Grandes",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "1" ~ "Grandes",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "4" ~ "Grandes",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "4" ~ "Grandes",
```

```
        cant_traba_tramo == "3" & ingreso_vta_pesos == "4" ~ "Grandes"
    ) %>%
    metaSurvey::step_compute(
        subsector = Division
    )

metaSurvey::get_metadata(new_svy)
```

```
   Type: eaii
  Edition: 2019-2021
   Engine: data.table
  Weight: w_trans
  Steps:
   - New group: realiza_innovacion
   - New group: sector
   - New group: innovativa
   - New group: tipo_actividad
   - New group: tipo_innovacion
   - New group: cant_traba_tramo
   - New group: ingreso_vta_pesos
   - New group: tamanio
   - New variable: subsector
```

```
metaSurvey::view_graph(new_svy)
```

5.3 EPH

```
ph2022_3 = metaSurvey::load_survey(
  path = metaSurvey::load_survey_example("eph2022_3.csv"),
  svy_type = "eph",
  svy_edition = "2022_3",
  svy_weight = "PONDERA"
) %>%
  metaSurvey::step_recode(
    "pea",
    ESTADO %in% 1:2 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "pet",
    ESTADO != 4 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "po",
    ESTADO == 1 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "pd",
```

```
    ESTADO == 2 ~ 1,
    .default = 0
  )
```

```
metaSurvey::view_graph(ph2022_3)
```

Chapter 6

Infraestructura

- Infra
- Docker
- Kubernetes
- Tests
- Envío a CRAN

> **Warning**
>
> Este capítulo está en proceso de escritura. Consulte la rama de desarrollo para ver el avance del capítulo

Chapter 7

Resultados

> **Warning**
>
> Este capítulo está en proceso de escritura. Consulte la rama de desarrollo para ver el avance del capítulo

Apendice aburrido de muestreo

Chang, W. (2022). *R6: Encapsulated classes with reference semantics.*

Detomasi, G. M. &. R. (2020). *Ech: Caja de herramientas para procesar la encuesta continua de hogares.* https://github.com/calcita/ech

Horvitz, D. G., & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, *47*(260), 663–685. https://doi.org/10.2307/2280784

Kozlowski, D., Tiscornia, P., Weksler, G., Rosati, G., & Shokida, N. (2020). *Eph: Argentina's permanent household survey data and manipulation utilities.* https://holatam.github.io/eph/

Kuhn, M., Wickham, H., & Hvitfeldt, E. (2024). *Recipes: Preprocessing and feature engineering steps for modeling.* https://github.com/tidymodels/recipes

Mailund, T. (2017). *Advanced object-oriented programming in r: Statistical programming for data science, analysis and finance.* SPRINGER.

R Core Team. (2023). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing. https://www.R-project.org/

Särndal, C.-E., Swensson, B., & Wretman, J. (2003). *Model Assisted Survey Sampling.* Springer Science & Business Media.

Thomas Mailund. (2017). *Metaprogramming in r* (1st ed.). Apress. https://www.amazon.com/Metaprogramming-Advanced-Statistical-Programming-Analysis/dp/1484228804

Walker, K., & Herman, M. (2024). *Tidycensus: Load US census boundary and attribute data as 'tidyverse' and 'sf'-ready data frames.* https://walker-data.com/tidycensus/

Wickham, H. (2011). Testthat: Get started with testing. *The R Journal*, *3*, 510. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf

Wickham, H. (2019). *Advanced r, second edition.* CRC Press.

Wickham, H., Danenberg, P., Csárdi, G., & Eugster, M. (2024). *roxygen2: In-Line Documentation for R.* https://roxygen2.r-lib.org/

Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). *Dplyr: A grammar of data manipulation.* https://dplyr.tidyverse.org

Wickham, H., Vaughan, D., & Girlich, M. (2024). *Tidyr: Tidy messy data.* https://tidyr.tidyverse.org