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

“*wwww*”

wwww

Resumen

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.

Agradecimientos

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

Notación

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

Capítulo 1

Introducción

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

En este trabajo se presenta el paquete `metasurvey`, una herramienta para el procesamiento y obtención de indicadores a partir de encuestas por muestreo. El paquete `metasurvey` permite al usuario tener un control total sobre el proceso de transformación de los microdatos a indicadores, permitiendo 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 fácil de referenciar en la implementación. Es crucial que este proceso sea transparente y entendible para el usuario.

1.1 Motivación

Las encuestas por muestreo son una herramienta fundamental para la obtención de información sobre cierta población de interés, ya que permiten obtener información a partir de una muestra representativa de la misma. Cada encuesta por muestreo tiene una estructura y un proceso generador de datos que permite obtener estimaciones puntuales y sus errores asociados. En general, el procesamiento de encuestas puede ser tedioso y propenso a errores o difíciles de brindar transparencia y reproducibilidad, especialmente si se quiere obtener indicadores que requieren varios pasos como tasas de mercado laboral, ingreso salarial, índices de pobreza, entre otros (Vilhuber 2020).

En general, 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. Si bien existen diferentes esfuerzos para facilitar el procesamiento de encuestas, en general estos paquetes tienen limitaciones en cuanto a la flexibilidad y transparencia del proceso de transformación de los microdatos a indicadores de interés. 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 edición de la encuesta, lo que resulta poco flexible ante cambios en la estructura, que pueden ser frecuentes en la práctica.

En este sentido, es importante construir una herramienta que permita al usuarios 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 fácil de referenciar en la implementación. Es crucial que este proceso sea transparente y entendible para el usuario y esto sera plasmado en el paquete `metasurvey`, disponible en R (R Core Team 2023).

1.2 Contexto

A lo que refire a la teoría de la estimación, es importante tener en cuenta la incertidumbre y errores asociados a las estimaciones producidas, en general esto no es considerado por los usuarios no expertos en metodología de muestreo. Esto puede llevar a conclusiones erróneas ya que en algunos casos el estimador asociado a la estimación cuenta con una alta variabilidad o fue calculado sin tener en cuenta el diseño muestral correcto. Antes de continuar, es importante distinguir dentro de la teoría de la estimación el enfoque **model-based inference** y **desing-based inference** (Lumley 2011). En el primer enfoque, se asume que la población de interés se puede modelar mediante un modelo probabilístico y se pueden obtener estimaciones de los parámetros del modelo mediante técnicas de inferencia estadística. En el segundo enfoque, se asume que la población de interés es finita y se puede obtener estimaciones de los parámetros de la población mediante técnicas de muestreo.

Dentro de este trabajo se mencionara de forma intensiva el concepto de peso o ponderador y su importancia en la estimación de varianzas y errores asociados a las estimaciones. Dentro de la estadística existen diferentes conceptos referidos a ponderadores o pesos, entre ellos (en base (Lumley 2011)):

- **Pesos muestrales**: Los pesos muestrales refiere a la cantidad de veces que un individuo de la población de interés está representado en la muestra. Estos pesos muestrales son los que provienen del diseño muestral, ya sea por el inverso de las probabilidades de selección, ajustes por no respuesta, entre otros.
- **Pesos de precisión**: El concepto de precisión puede relacionarse con la variabilidad que tiene una observación sobre la estimación de un parámetro.
- **Pesos de frecuencia**: Refire a la cantidad de veces que aparece un individuo en una muestra y este es resumido para incluir en un único registro.

Es importante hacer esta distinción ya que tomando en cuenta los pesos en cualquiera de sus definiciones o consideraciones, en la mayoría de los casos **se puede obtener estimaciones puntuales correctas**, sin embargo como se mencionó anteriormente llegar a medidas de incertidumbre como errores estándar, intervalos de confianza totalmente incorrectos.

Una vez considerado el proceso de inferencia también es crucial tener en cuenta el proceso de transformación de los microdatos a indicadores ya que es importante para interpretar los indicadores de manera correcta y realizar comparaciones a lo largo del tiempo para formalizar la metodología de su construcción. Muchas veces,

diferentes usuarios hacen el mismo esfuerzo 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 los últimos años, el uso de R (R Core Team 2023) ha crecido exponencialmente en la comunidad científica, en especial en el área de la estadística y la ciencia de datos. R es un lenguaje 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. En este sentido, `metasurvey` busca ser una herramienta relevante para el trabajo con encuestas por muestreo en general ya sea en las ciencias sociales o el uso genérico para otras disciplinas, buscando solucionar las limitaciones anteriormente mencionadas.

Antes de continuar es importante definir el concepto de Estadística Computacional y su diferencia con Computación Estadística (Cook 2014), siendo este trabajo un aporte a la Estadística Computacional. La Estadística Computacional se refiere a la implementación de algoritmos y métodos estadísticos en un lenguaje de programación, mientras que la Computación Estadística se refiere a la utilización de herramientas computacionales para resolver problemas estadísticos. En este sentido, R es un lenguaje de programación que permite realizar Estadística Computacional y Computación Estadística, ya que cuenta con una amplia variedad de paquetes que permiten implementar algoritmos y métodos estadísticos y realizar análisis de datos de manera eficiente y reproducible.

1.3 Antecedentes e implementaciones similares

Actualmente 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` (Lumley 2024), `gustave` (Chevalier 2023), `vardpoor` (Breidaks, Liberts, and Ivanova 2020), `svrep` (Schneider 2023), `weights` y aquellos que permiten acceder y manipular encuestas específicas como `ech` (Detomasi 2020), `eph` (Kozlowski et al. 2020), `tidycensus` (Walker and Herman 2024), `casen` (Vargas 2024) entre otros. Sin embargo, estos últimos 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 edición 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` (Detomasi 2020), 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` (Kozlowski et al. 2020), 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` (Wickham et al. 2023) o `tidyr` (Wickham, Vaughan, and Girlich 2024), 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` (Walker and Herman 2024), donde existe una función para obtener datos sobre la migración de la comunidad estadounidense, y en la misma implementación se encuentran pasos para mejorar la estructura del conjunto de datos a devolver. En este sentido, el usuario no puede aislar el proceso de re-codificación/construcción de variables sobre variables originales y la obtención de datos geográficos y presentación.

1.4 Propuesta

Para científicos 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 fácil de referenciar en la implementación. Es crucial que este proceso sea transparente y entendible para el usuario. En capítulos 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 metaprogramación.

Al trabajar con encuestas por muestreo, es importante tener en cuenta la forma en la que se obtuvieron 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 capítulos 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.

1.5 Desarrollo del paquete `metasurvey`

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 abordaran ejemplos con la implementación de `metasurvey`.

En este sentido, `metasurvey` pretende ser una herramienta relevante para el trabajo con encuestas por muestreo en general ya sea en las ciencias sociales o el uso genérico para otras disciplinas, 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` (Kuhn, Wickham, and Hvitfeldt 2024).

1.6 Esquema del documento

El documento se estructura de la siguiente manera: en el siguiente capítulo se presentará 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 ahondará 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**.

Este documento puede leerse en su formato de pagina web o en su formato de documento PDF. Tanto el código fuente del paquete se encuentran disponibles de forma pública en el repositorio de Github y el código fuente de este documento se encuentra disponible en el repositorio. Para la realización de este documento se utilizó `quarto` (Publishing 2024) para la generación de documentos dinámicos que permiten escribir texto junto con código R.

Para finalizar, es importante mencionar que el paquete `metasurvey` es un proyecto en desarrollo y se encuentra en una etapa temprana de desarrollo, por lo que se espera que en el futuro se realicen mejoras y se agreguen nuevas funcionalidades, por lo que se invita a la comunidad a colaborar en el desarrollo del paquete a través de la creación de *issues* en el repositorio de GitHub o mediante *pull requests* con mejoras o nuevas funcionalidades.

Es recomendable instalar la versión de desarrollo para continuar con el documento, para ello se puede instalar el paquete `metasurvey` desde el repositorio de GitHub con el siguiente código:

```
remotes::install_github("metasurveyr/metasurvey")
```

Aunque también se puede instalar la versión de CRAN con el siguiente código:

```
is_available <- "metasurvey" %in% rownames(available.packages(repos = "https://cloud.

if (is_available) {
  install.packages("metasurvey")
```

```
} else {
  remotes::install_github("metasurveyr/metasurvey")
  message("Se instalo la versión de desarrollo de metasurvey")
}
```

Capítulo 2

Marco conceptual

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

El objetivo principal de este capítulo es presentar los conceptos básicos que se utilizarán a lo largo de este trabajo, 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 estimación de parámetros poblacionales y sus respectivos errores estándar. Se hará una primera introducción a diseños sencillos y se propondrán diferentes estimadores y se hará mención a su diferencia con los diseños complejos, situación común en Encuestas Socioeconómicas. 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 indicadores socio-demográficos y económicos, en este sentido, es importante tener en cuenta un marco teórico para realizar estas 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 (Särndal, Swensson, and Wretman 2003).

2.1.1 Diseño muestral

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

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

La definición matemática 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 comúnmente 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 estimadores 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 (Särndal, Swensson, and Wretman 2003) y (Horvitz and Thompson 1952).

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 representa el individuo k en la población. Este valor es el que comúnmente se publica junto a los microdatos y el estándar 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 construir 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 estimadores 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 particular 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 muéstrales 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 ciertos ponderadores que no son precisamente los ponderadores originales definidos en la sección anterior sino que son sometidos a un proceso de **calibración** donde se intenta ajustar a ciertas variables de control y mejorar problemas causados por la no respuesta. Al realizar el proceso de calibración los ponderadores calibrados son lo mas cercano posible a los ponderadores originales, de forma que si los ponderadores originales son insesgados, los ponderadores calibrados serán próximos a ser insesgados.

En la practica para diseños complejos no se dispone de las probabilidades de selección de segundo orden insumo principal para calcular los errores estándar, por esto es que se requiere optar con metodologías alternativas como el método del ultimo conglomerado, método de replicación jackknife, método de bootstrap, entre otros. En este sentido, es importante tener en cuenta que la varianza de los estimadores es un componente fundamental para realizar inferencias y cuantificar la confiabilidad de los resultados.

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 estadísticos junto a la estimación puntual se presentan los errores estándar aunque 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 erróneo.**

Una vez presentado conceptos básicos de muestreo es importante entender como esto estará disponible en el paquete metaSurvey, en este sentido, se presentarán los conceptos básicos de programación funcional y orientada a objetos en R para luego enfocarnos en la meta-programación.

2.2 Desarrollo de paquetes en R

R es un lenguaje de código abierto y además cuenta con una gran comunidad de usuarios, en diferentes áreas de investigación, esto ha permitido que se desarrollen una gran cantidad de paquetes que permiten realizar diferentes tareas de análisis de datos, visualización, bioinformática, aprendizaje automático y ramas afines a la estadística. Dentro de la comunidad existen diferentes organizaciones que se encargan de mantener la calidad de los paquetes y de asegurar que los paquetes cumplan con ciertos estándares de calidad, una de estas organizaciones es el **Comprehensive R Archive Network** (**CRAN**), que es un repositorio de paquetes de R que contiene versiones estables de los paquetes de R, bioconductor, que es un repositorio de paquetes de R que contiene paquetes para el análisis de datos biológicos, y rOpenSci que 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* (Wickham et al. 2024) 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* (Wickham 2011) 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 (Chambers 2014), 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 (Wickham 2019, 204–81). 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 (Wickham 2019, 285–370; Mailund 2017). 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 `Survey` 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* (Chang 2022) 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 (Wickham 2019, 373–500; Thomas Mailund 2017) . En R, la meta-programación se realiza utilizando el sistema de meta-programación 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`.

En metasurvey se utiliza la meta-programación para generar código de manera dinámica y realizar operaciones sobre los datos de manera eficiente. En particular se utiliza la función `eval` para evaluar expresiones y la función `substitute` para reemplazar variables en expresiones. Además, se utilizan las funciones `lapply`, `sapply`, `mapply` y `do.call` para aplicar funciones a listas y vectores de manera eficiente. En general, la meta-programación es una técnica muy útil para realizar operaciones sobre los datos de manera eficiente y sencilla.

En el capítulo 3 se presentarán los antecedentes de metodologías de estimación de varianzas, revisión de medidas de incertidumbre, paquetes similares y mejoras que son incorporadas en el paquete metasurvey. En el capítulo 4 se hablara sobre la implementación de las diferentes partes que conforman el paquete, una breve reseña del esquema de test, la API para almacenar las recetas junto a su interacción con el usuario. Posteriormente se mostrara un ejemplo de uso del paquete y se presentarán las conclusiones y trabajos futuros.

Capítulo 3

Marco teórico

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

En este capitulo se presentan los antecedentes y conceptos aplicados en el desarrollo de `metasurvey` considerando conceptos de investigación reproducible, la importancia de R como herramienta para la investigación reproducible, la revisión de paquetes de R para el procesamiento de encuestas por muestreo y la importancia del diseño muestral, la importancia de la estimación de las varianzas en la generación de indicadores y los trabajos previos en los que se basa el paquete. Estos conceptos son fundamentales para poder entender el desarrollo y la importancia de tener un flujo de trabajo para la generación de indicadores sociales.

En la actualidad, la generación de indicadores sociales se ha vuelto una tarea fundamental tanto para la toma de decisiones como para la investigación. Sin embargo, este proceso puede ser complejo, requiriendo conocimiento sobre el formulario de la encuesta, formas de construir ciertos indices o variables auxiliares que no necesariamente sea trivial y depende de la experiencia del usuario.

Este proceso de generación de indicadores en algunos casos no es transparente o no se documenta de manera adecuada, en parte por la falta de herramientas que lo permitan y en otra parte por la falta de cultura de la reproducibilidad en la generación de indicadores ya que en la mayoría de los casos se hace referencia a los datos y no al proceso de generación de los indicadores.

3.1 Investigación reproducible

El concepto de investigación reproducible ha cobrado relevancia en los últimos años, tanto en la academia como en la industria y esto se debe a la fricción que puede llegar a existir al momento de presentar resultados de investigación o generación indicadores relevantes para la toma de decisiones debido al proceso de generación de los mismos. Dentro de las diferentes disciplinas generar ambientes de trabajo reproducibles puede llegar a ser un desafío, ya que en la mayoría de los casos se utilizan diferentes herramientas, lenguajes de programación y bases de datos.

En la actualidad existen diferentes revistas científicas que promueven la investigación reproducible, herramientas, guías para buenas prácticas para trabajar con datos y código fuente como Sumatra (Davison and Huth 2012), implementaciones de programación literal (Knuth 1984) como RMarkdown (Allaire et al. 2024) o Jupyter

Notebook (Kluyver et al. 2024) y diferentes implementaciones para gestionar dependencias de software como Anaconda (Anaconda 2024), aunque algunas de ellas se han vuelto herramientas de pago o ya no existen en la actualidad, mas referencias y casos de uso pueden encontrarse en (Stodden, Leisch, and Peng 2014).

Antes de continuar es necesario definir conceptos fundamentales en el ámbito de la investigación reproducible, tales como la *Reproducibilidad* que refiere a la capacidad de poder repetir los resultados de un estudio, experimento o la obtención de un indicador. Si bien la reproducibilidad en un artículo de investigación científica al utilizar indicadores tanto en contextos académicos como en aplicaciones de monitoreo o divulgación de información, rara vez se documenta o se menciona de que manera se generó ese resultado haciendo referencia únicamente a los datos y rara vez al código fuente. Aún compartiendo el código fuente, esto aún no suficiente para poder reproducir un estudio o un indicador por incompatibilidades de versiones de software, cambios en la estructura de los datos interpretaciones de los datos, estilos de programación, entre otros pudiendo llevar mucho tiempo y esfuerzo para poder replicar un resultado.

El proceso de tratamiento de datos y limpieza forma parte de lo que se conoce como *publicaciones grises* (Vilhuber 2020). Este concepto se refiere a la publicación de datos, código y reportes que no son publicaciones formales, pero son esenciales para generar conocimiento científico. En su mayoría al no tener una revisión por pares o una forma estandarizada esto se incluye de forma muy dispar o sin ningún tipo de documentación para poder ser reproducido y esto forma una gran parte de la investigación científica que no se encuentra aprovechada.

Existen diversas iniciativas destinadas a fomentar la reproducibilidad en la ciencia, lo que ha llevado a las revistas a establecer políticas de datos y código abierto. Sin embargo, persisten desafíos en la generación de indicadores sociales, ya que como se menciono anteriormente no basta con hacer referencia a los datos, como se señala en (Bechhofer et al. 2013); además de publicar el artículo junto a los datos, es necesario vincular los objetos de investigación (Research Objects **RO**), existen diferentes plataformas que permiten la publicación de estos objetos como **Zenodo** y **Figshare** o **OSF** que permiten la integración de datos, código e interacción con repositorios con control de versiones como GitHub o GitLab.

De conceptos generales sobre reproducibilidad es importante contar con un flujo de trabajo (*Workflow managment System* (Prabhu and Fox 2020)) para la obtención de estimadores en el procesamiento de encuestas por muestreo ya que el indicador final es el resultado de una serie de pasos que se deben seguir de manera ordenada y documentada para poder ser auditados y replicados en diferentes contextos, inspirado en (Sandve et al. 2013) se pueden considerar algunas buenas prácticas para la generación de indicadores:

- **Para cada resultado, se debe tener un respaldo de como fue construido**: Al trabajar con lenguajes de programación como R, los script de código fuente son un respaldo de como obtener cierto resultado, sin embargo, esto puede estar ligado a tu estilo de programación y la versión de los paquetes que se utilizan.
- **Crear manuales en la manipulación de datos**: Es importante resumir cada paso por mas mínimo que sea en la transformación de variables, esto permite entender todo el proceso de generación de un indicador.

- **Guardar las versiones de los paquetes utilizados**: Al trabajar con R, es importante guardar las versiones de los paquetes que se utilizan, esto permite que en un futuro se pueda replicar el proceso de generación de indicadores, para esto puede utilizarse herramientas como `renv` (Ushey and Wickham 2023) un paquete que permite crear ambientes locales con versiones especificas de paquetes de R, `venv` (Python Software Foundation 2024) que son ambientes virtuales en python o Docker (Merkel 2014) para poder emular un ambiente de trabajo en diferentes sistemas operativos.
- **Guardar pasos intermedios, en un formato estándar**: Al trabajar con encuestas por muestreo y para crear indicadores sencillos se realizan dos grandes tipos de operaciones: crear grupos o categorías o realizar operaciones matemáticas, es importante guardar estos pasos en un formato estándar para poder ser reutilizados en diferentes contextos.
- **Compartir las ejecuciones y scripts**: Es importante que los scripts de código fuente estén disponibles para que puedan ser auditados y replicados en diferentes contextos.

3.1.1 Conceptos clave

`metasurvey` se basa en las buenas prácticas mencionadas anteriormente y permite crear herramientas de flujo de trabajo siguiendo los siguientes principios:

- **Reusable**: Se separa el proceso de transformación de variables en `Steps` que refiere a transformaciones de columnas, estos procedimientos pueden ser comunes tanto en diferentes encuestas como en diferentes indicadores. Estos `Steps` pueden ser reutilizados en diferentes `Recipes` para calcular indicadores de mercados de trabajo, pobreza, e incluso aplicarlos en varias encuestas simultáneamente mediante un `Workflow`.
- **Repetible**: Al tener un proceso definido en un `Workflow`, es posible repetir el proceso de generación de indicadores de la misma manera y automatizar la generación de reportes.
- **Referenciable** y **Acreditable**: Al contar con un `Workflow`, es posible hacer referencia al proceso de generación de indicadores indicando todos los pasos seguidos y el autor o equipo que lo realizó. Además, se puede acreditar a los autores de los `Steps` y `Recipes` que se utilizaron en el proceso.

3.1.2 Workflow reproducible

El concepto de *Workflow* no es nuevo y exclusivo en la comunidad científica, en la actualidad en la industria de la ciencia de datos se han desarrollado diferentes herramientas para la gestión de flujos de trabajo para el procesamiento de datos, con diferentes enfoques y objetivos. `metaSurvey` se inspira en diferentes herramientas como **Apache AirFlow** ("Apache Airflow Documentation," n.d.) que es una plataforma de orquestación de flujos de trabajo de código abierto, **Great Expectations** (Expectations 2024) que es una biblioteca de validación de datos para la generación de reportes de calidad de datos y **Make** que es una herramienta de automatización de flujos de trabajo que se basa en la definición de reglas y dependencias.

En el ámbito del aprendizaje automático existe un gran esfuerzo para poder desgranar y documentar los modelos conocido como **Model Cards** (Mitchell et al. 2019) donde se hace un detalle de los algoritmos utilizados, las métricas de evaluación, los datos utilizados y su procesamiento, siendo esto el análogo a los `Steps` y `Recipes` de `metaSurvey`. Este concepto se ha extendido siendo un estándar en la industria y siendo adoptado por diferentes organizaciones como **Google** y **Hugging Face**.

Tomando en cuenta estos conceptos, `metaSurvey` tiene disponible la posibilidad de generar, compartir y visualizar los flujos de trabajo de manera gráfica permitiendo la transparencia y auditabilidad de los procesos de generación de indicadores.

3.2 Investigación reproducible en R

Dentro de CRAN existe una guía sobre conjunto de paquetes y herramientas con objetivos comunes denominado **Task Views** que agrupa paquetes de R que se utilizan para un propósito específico. En el Task View de Reproducible Research se encuentran diferentes paquetes que permiten la generación de reportes dinámicos, la gestión de flujos de trabajo y la generación de documentos interactivos aunque también existen herramientas para la gestión de flujos de trabajo generales como `targets` (Landau 2021) y `drake` (Landau 2018), `metaSurvey` fue inspirado en los conceptos y la forma de trabajo de estos paquetes.

Los conceptos de meta-programación y programación orientada a objetos fue inspirado en el paquete `mlr3pipelines` (Binder et al. 2021) que permite la creación de flujos de trabajo para el preprocesamiento de datos y la generación de modelos de aprendizaje automático, aquí se definen `PipeOps` que son operaciones que se pueden aplicar a los datos y se pueden combinar en un `Graph` que define el flujo de trabajo para ello se definen clases y métodos que permiten una fácil extensión por parte del usuario y la creación de flujos de trabajo complejos.

Dentro de la comunidad existen organizaciones como ROpenSci que promueven la ciencia abierta y la reproducibilidad en la investigación científica, proporcionando herramientas y guías para promover la ciencia abierta mediante R. Esta organización promueve la creación de paquetes donde además de la guías sobre el desarrollo de paquetes y la revisión de los mismos, se promueve la creación de paquetes que sean de utilidad para la comunidad científica definiendo estándares de calidad y documentación. Para formar parte de ROpenSci, se sigue una evaluación entre pares y una revisión de la calidad del paquete, además de la documentación y la calidad del código complementado con tests automatizados.

3.2.1 Herramientas para el procesamiento de encuestas

En el ámbito de las encuestas por muestreo, existen diferentes paquetes que permiten el procesamiento de encuestas por muestreo o la generación de estadísticas oficiales, esto se puede ver en el Task View de Official Statistics & Survey Methodology donde se encuentran diferentes tipos de paquetes desde la preparación de formularios, calibración, análisis de datos, acceso a datos oficiales, entre otros.

Para el procesamiento de encuestas por muestreo, existe una serie de paquetes que permiten implementar la metodología de encuestas por muestreo como puede ser el caso de `survey` (Lumley 2024) que permite el análisis de encuestas complejas, `srvyr` (Ellis and Schneider 2023) aunque estos son utilizados en el proceso final o de inferencia y no en el proceso de la construcción y limpieza de los datos como si lo

hace `ech` (Detomasi 2020) que tiene diferentes funciones para la ECH y permite al usuario crear variables referidas a Vivienda, Educación, Mercado de Trabajo, Ingresos y Pobreza algo similar con `eph` (Kozlowski et al. 2020) que permite la descarga de datos de la EPH y la creación de variables para analizar la pobreza y el mercado de trabajo.

Este ultimo grupo de paquetes o **caja de herramientas** tienen la limitación que no permiten la reutilización de los pasos de limpieza y transformación de los datos de forma sencilla y nativa, además de no poder visualizar el flujo de trabajo de manera gráfica, lo que dificulta la auditoría y la replicabilidad de los procesos de generación indicadores, `metasurvey` busca llenar este vacío permitiendo la reutilización de los pasos de limpieza y transformación de los datos, la visualización del flujo de trabajo y la generación de reportes de manera sencilla.

3.3 Diseño de encuestas y estimación de varianza

Como fue introducido en el capitulo anterior y en la sección de antecedentes es sencillo obtener estimaciones puntuales, sin embargo, es necesario presentar una medida de precisión de la estimación ya que en algunos casos puede ser que el tamaño de la muestra no sea suficiente para obtener estimaciones precisas. En el caso de las encuestas por muestreo, es necesario tener en cuenta el diseño de la encuesta, la estratificación, la ponderación y el efecto de conglomerados, ya que estos factores influyen en la precisión de la estimación. Para ello, es necesario contar con alguna metodología que permita estimar varianzas ya que para diseños complejos o estadísticos no lineales, la estimación de varianzas no es trivial.

En la actualidad, existen diferentes métodos para la estimación de varianzas, aunque en la mayoría de los casos se utilizan métodos de remuestreo como el Boostrap o el Jackknife, sin embargo existen diferentes ideas o propuestas como se menciona en (Deville and Tille 1998) y (Deville and Tillé 2005) que demuestran con resultados numéricos estimadores del tipo **H-T** bajo un diseño balanceado puede aproximarse desde el enfoque de regresión o calibración. Además existen estimadores alternativos donde complementan métodos de remuestreo para aproximar probabilidades de inclusión de segundo orden (Emilio L. Escobar and Berger 2013) utilizando ciertas aproximaciones límites (Hajek 1964).

Cada metodología depende de cada diseño y variables a estimar, por esto es que existen diferentes metodologías y paquetes como `gustave` (Chevalier 2023) , `vardpoor` (Breidaks, Liberts, and Ivanova 2020), `svrep` (Schneider 2023) y `samplingVarEst` (Emilio Lopez Escobar, Zamudio, and Rosas 2023), aunque existen similitudes entre implementaciones y métodos es difícil encontrar una implementación que permita la estimación de varianzas de manera sencilla y que permita la reutilización de los pasos de limpieza y transformación de los datos.

3.3.1 Resumen de las implementaciones

En la tabla a continuación se presenta un resumen de las implementaciones de los paquetes mencionados anteriormente:

Comparación de Paquetes para Análisis de Encuestas

Información actualizada sobre versiones y últimos commits

Paquete	Descripción	Último Commit
survey	Permite el análisis de encuestas complejas, aquí se pueden definir los estratos, los conglomerados y las ponderaciones, además tiene implementaciones de los estimadores más comunes como el estimador de Horvitz-Thompson, el estimador de regresión y el estimador de calibración. Sin embargo, dentro de este mismo paquete no existe una forma de integrar los pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo. Este paquete es utilizado como dependencia **clave** para ocupar los principales estimadores poblacionales y diseño muestral.	2024-03-20T15:30:02Z
srvyr	Es una interfaz para el paquete `survey` que permite trabajar con `dplyr` y `tidyverse`, sin embargo, no permite de forma nativa la reutilización de los pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo aunque permite la integración con `dplyr` y `tidyverse`. Este paquete cuenta con muchas dependencias debido al uso de `tidyverse` y va en contra de la filosofía de `metasurvey` de tener la menor cantidad de dependencias posibles. Además metasurvey tiene su propia capa de abstracción para trabajar con `survey`.	2024-10-05T17:06:52Z
gustave	Cuenta con una función principal `qvar` que no utiliza el diseño muestral como argumento sino que se deben de ingresar las probabilidades de inclusión y no queda del todo claro los estadísticos que se pueden obtener. Además, no permite la reutilización de los pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo. Si bien tiene diferentes implementaciones de aproximación de varianzas como la de Deville y Tillé, no es la forma de estimación que utiliza `metasurvey` por defecto aunque puede ser una alternativa para futuras implementaciones.	2024-01-15T16:56:39Z
vardpoor	Este paquete utiliza la estimación de varianzas por el método de último conglomerado, otras métricas como el efecto diseño, linealización del estimador de razón, coeficiente de Gini y diferentes estadísticos enfocados en la desigualdad. Si bien es una implementación interesante, su sintaxis es particular y ajena a la del paquete `survey`, tiene muy pocas dependencias pero puede ser implementado como motor de estimación de varianzas en futuras implementaciones.	2022-02-17T14:50:27Z
convey	Este paquete sí utiliza el diseño muestral como argumento e implementa algunos estadísticos que se incluyen en `vardpoor`, sin embargo, no permite la reutilización de pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo. Sin embargo, al tener una implementación del diseño muestral, se sincroniza muy bien con `metasurvey`. Los estadísticos pueden ser utilizados de forma nativa en `metasurvey`.	2024-07-14T09:51:04Z
svrep	Extensión del paquete `survey` para trabajar con réplicas sobre los pesos muestrales, permite la estimación de varianzas por el método de Bootstrap, Jackknife y replicación de balanceo. La integración con el diseño del tipo `bootstrap` y `jackknife`. Este paquete es utilizado como dependencia **clave** para ocupar los principales estimadores poblacionales y diseño muestral y es utilizado en `metasurvey` para la estimación de varianzas.	2024-04-26T18:40:02Z

Datos extraídos desde GitHub a través de la API.

En capítulos posteriores se presentará la implementación de conceptos de workflows, meta-programación y metodologías de estimación de varianzas en `metasurvey` para la generación de indicadores sociales.

Capítulo 4

Desarrollo y metodología

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

En este capítulo se divide en tres partes, la primera parte se centra en la metodología de los métodos de estimación de varianzas, la segunda parte en el desarrollo e implementación de los métodos mencionados y la meta-programación, la tercera parte refiere a la infraestructura, la automatización de pruebas y el envío a CRAN. x1 ## Estimación de varianzas

4.0.1 Replicas bootstrap

4.0.2

4.1 Desarrollo e implementación

4.2 Infraestructura

Capítulo 5

Resultados

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

```
#> Welcome to: metasurvey version 0.0.1.9000
#> use_copy: TRUE
#> metasurvey.engine: data.table
#> metasurvey.api.key: ********
#> metasurvey.user: public
```

En este capitulo se va a hacer uso del paquete para replicar diferentes informes donde se utilizan encuestas por muestreo a nivel nacional, ya sea informes de mercado de trabajo, de innovación o de ingresos de los hogares. Antes de realizar cualquier análisis se hara mención al diseño de la encuesta y luego se procederá a realizar los pasos necesarios para replicar los resultados, ya sea re-codificación de variables, cálculo de indicadores y análisis de los mismos.

5.1 Encuesta Continua de Hogares

La Encuesta Continua de Hogares (ECH) es la principal fuente de información referida al mercado de trabajo en Uruguay. La encuesta se realiza en forma continua con periodicidad mensual desde el 1968. En sus primeros años la encuesta solo consideraba como universo de hogares a Montevideo sin embargo luego en 1980 se extendió a todo el país mediante un programa de las Naciones Unidas para el Desarrollo y el Fondo de las Naciones Unidad para Actividades de Población llegando a cubrir todo el territorio nacional.

Actualmente el INE tiene publicado en su página web microdatos de la encuesta desde el año 2006, en el portal ANDA se pueden encontrar junto a los microdatos los códigos de las variables y las definiciones de las mismas junto a la descripción del diseño de la encuesta.

La encuesta a lo largo de los años ha ido incorporando nuevas variables y modificando las existentes, por lo que es importante tener en cuenta la versión de la encuesta que se está utilizando para realizar los análisis y dependiendo del grupo de variables que se quiera analizar puede que sea mas o menos tedioso el proceso de re-codificación de variables y cálculo de indicadores. Con la ayuda de recetas y el paquete metasurvey se puede automatizar el proceso de re-codificación de variables y cálculo de indicadores para poder calcular los indicadores de interés.

En lo que sigue se van a utilizar los microdatos de la ECH del año 2024 Abril, para replicar los resultados presentados en el Boletin Técnico, Actividad, Empleo y Desempleo. Abril 2024 y Informe diferencial de mercado de trabajo referidos a variables de mercado de trabajo a nivel mensual. A continuación se hara lo mismo con el informe Mercado de trabajo por área geográfica de residencia y Boletín Técnico Ingresos de los Hogares y de las Personas

5.1.1 Actividad, empleo y desempleo (Mensual)

En este boletín se encuentran las tres variables principales del mercado de trabajo, la tasa de actividad, la tasa de empleo y la tasa de desempleo. La tasa de actividad se calcula como el cociente entre la población económicamente activa y la población en edad de trabajar, la tasa de empleo se calcula como el cociente entre la población ocupada y la población en edad de trabajar y la tasa de desempleo se calcula como el cociente entre la población desocupada y la población económicamente activa.

Para calcular estas tasas se necesita re-codificar las variables de la encuesta para poder calcular las tasas de interés. A continuación se muestra el código para re-codificar las variables y calcular las tasas de interés.

```
#> Loading required package: data.table
```

5.1.1.1 Re-codificación de variables

5.1.1.2 Estimación

```
#> [[1]]
#> [[1]]$survey
#>    Type: ech
#>   Edition: 2023
#>    Engine: data.table
#>   Design:
#>     * Type:
#>       * Package: survey
#>       * Variance estimation: Ultimate cluster
#>     * PSU: ~1
#>     * Strata: NULL
#>     * Weight: W
#>     * FPC: NULL
#>     * Calibrate formula: NULL
#>   Steps:
#>    - New group: pea
#>    - New group: pet
#>    - New group: po
#>    - New group: pd
#>    - New group: region_reco
#>   Recipes: None
#>
#> [[1]]$calls
#> list(survey::svyratio(~pea, ~pet), survey::svyratio(~pd, ~pea),
#>     survey::svyratio(~po, ~pet))
#>
#> [[1]]$result
```

```
#>                          stat      value          se          cv
#>                        <char>      <num>       <num>       <num>
#> 1: survey::svyratio: pea/pet 0.63844605 0.003650599 0.005717945
#> 2:  survey::svyratio: pd/pea 0.07807569 0.002709077 0.034698084
#> 3:  survey::svyratio: po/pet 0.58859894 0.003776409 0.006415930
#> Warning in melt.data.table(dt, measure.vars = est_cols, variable.name =
#> "stat", : 'measure.vars' [region_reco, pea/pea] are not all of the same
#> type. By order of hierarchy, the molten data value column will be of type
#> 'character'. All measure variables not of type 'character' will be coerced
#> too. Check DETAILS in ?melt.data.table for more on coercion.
#> [[1]]
#> [[1]]$survey
#>   Type: ech
#>  Edition: 2023
#>   Engine: data.table
#>  Design:
#>    * Type:
#>      * Package: survey
#>      * Variance estimation: Ultimate cluster
#>    * PSU: ~1
#>    * Strata: NULL
#>    * Weight: W
#>    * FPC: NULL
#>    * Calibrate formula: NULL
#>  Steps:
#>   - New group: pea
#>   - New group: pet
#>   - New group: po
#>   - New group: pd
#>   - New group: region_reco
#>  Recipes: None
#>
#> [[1]]$calls
#> list(survey::svyby(~pea, denominator = ~pea, by = ~region_reco,
#>     survey::svyratio))
#>
#> [[1]]$result
#>    se.pea/pea        stat      value    se
#>         <num>      <fctr>     <char> <num>
#> 1:          0 region_reco   Interior    NA
#> 2:          0 region_reco Montevideo    NA
#> 3:          0     pea/pea          1     0
#> 4:          0     pea/pea          1     0
```

Capítulo 6

TODO: Revisar como escribir la viñeta de uso de recipes y darle un contexto en el capitulo.

Acá va la viñeta Use recipes

6.1 ECH

```
metasurvey::set_engine("data.table")
#> Engine: data.table

ech_meta = metasurvey::load_survey(
  path = metasurvey::load_survey_example(
    "ech",
    "ech_2018"
  ),
  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)
```

6.2 EAII

```
svy_example = metasurvey::load_survey(
    svy_type = "eaii",
    svy_edition = "2019-2021",
    svy_weight = "w_trans",
    input = metasurvey::load_survey_example(
      "eaii",
      "2019-2021"
    ),
    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
#>   Design:
#>     * Type:
#>       * Package: survey
#>       * Variance estimation: Ultimate cluster
#>     * PSU: ~1
#>     * Strata: NULL
#>     * Weight: w_trans
#>     * FPC: NULL
#>     * Calibrate formula: NULL
#>   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
#>   Recipes: None
```

```
metasurvey::view_graph(new_svy)
```

6.3 EPH

```
eph2022_3 = metasurvey::load_survey(
  path = metasurvey::load_survey_example(
    "eph",
    "eph2022_3"
  ),
  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(eph2022_3)
```

6.4 ECH

6.4.1 Actividad, empleo y desempleo (Mensual)

6.4.2 Mercado de trabajo (Trimestral)

6.4.3 Ingreso de los hogares, pobreza y desigualdad

6.5 EAII

6.5.1 Dominios

6.5.2 Replicar resultados de la sección actual

6.5.3 Medio ambiente

Capítulo 7

Bibliografía

Allaire, JJ, Yihui Xie, Jade McPherson, Joseph Luraschi, Kevin Ushey, and Amber Atkins. 2024. *RMarkdown*. https://rmarkdown.rstudio.com/.

Anaconda, Inc. 2024. *Anaconda Distribution*. https://www.anaconda.com/.

"Apache Airflow Documentation." n.d. https://airflow.apache.org/docs/latest/.

Bechhofer, Sean, Iain Buchan, David De Roure, Paolo Missier, John Ainsworth, Jiten Bhagat, Philip Couch, et al. 2013. "Why Linked Data Is Not Enough for Scientists." *Future Generation Computer Systems*, Special section: Recent advances in e-science, 29 (2): 599–611. https://doi.org/10.1016/j.future.2011.08.004.

Binder, Martin, Florian Pfisterer, Michel Lang, Lennart Schneider, Lars Kotthoff, and Bernd Bischl. 2021. "Mlr3pipelines - Flexible Machine Learning Pipelines in r." *Journal of Machine Learning Research* 22 (184): 1–7. https://jmlr.org/papers/v22/21-0281.html.

Breidaks, Juris, Martins Liberts, and Santa Ivanova. 2020. *Vardpoor: Estimation of Indicators on Social Exclusion and Poverty and Its Linearization, Variance Estimation*. Riga, Latvia: Central Statistical Bureau of Latvia. https://csblatvia.github.io/vardpoor/.

Chambers, John M. 2014. "Object-Oriented Programming, Functional Programming and r." *Statistical Science* 29 (2). https://doi.org/10.1214/13-STS452.

Chang, Winston. 2022. *R6: Encapsulated Classes with Reference Semantics*.

Chevalier, Martin. 2023. *Gustave: A User-Oriented Statistical Toolkit for Analytical Variance Estimation*. https://CRAN.R-project.org/package=gustave.

Cook, Di. 2014. "Statistical Computing Research |." http://dicook.org/2014/10/05/content/post/2014-10-5-statistical-computing/.

Davison, Andrew P, and John E Huth. 2012. "Sumatra: A Toolkit for Reproducible Research." *arXiv Preprint arXiv:1207.5548*.

Detomasi, Gabriela Mathieu & Richard. 2020. "Ech: Caja de Herramientas Para Procesar La Encuesta Continua de Hogares." https://github.com/calcita/ech.

Deville, Jean-Claude, and Yves Tille. 1998. "Unequal Probability Sampling Without Replacement Through a Splitting Method." *Biometrika* 85 (1): 89–101. https://www.jstor.org/stable/2337311.

Deville, Jean-Claude, and Yves Tillé. 2005. "Variance Approximation Under Balanced Sampling." *Journal of Statistical Planning and Inference* 128 (2): 569–91. https://doi.org/10.1016/j.jspi.2003.11.011.

Ellis, Greg Freedman, and Ben Schneider. 2023. *Srvyr: 'Dplyr'-Like Syntax for Summary Statistics of Survey Data*. https://CRAN.R-project.org/package=srvyr.

Escobar, Emilio L., and Yves G. Berger. 2013. "A New Replicate Variance Estimator

for Unequal Probability Sampling Without Replacement." *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* 41 (3): 508–24. https://www.jstor.org/stable/43186201.

Escobar, Emilio Lopez, Ernesto Barrios Zamudio, and Juan Francisco Munoz Rosas. 2023. *samplingVarEst: Sampling Variance Estimation*.

Expectations, Great. 2024. *Great Expectations Documentation*. Superconductive. https://docs.greatexpectations.io.

Hajek, Jaroslav. 1964. "Asymptotic Theory of Rejective Sampling with Varying Probabilities from a Finite Population." *The Annals of Mathematical Statistics* 35 (4): 1491–1523. https://doi.org/10.1214/aoms/1177700375.

Horvitz, D. G., and D. J. Thompson. 1952. "A Generalization of Sampling Without Replacement from a Finite Universe." *Journal of the American Statistical Association* 47 (260): 663–85. https://doi.org/10.2307/2280784.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, et al. 2024. *Jupyter Notebook*. https://jupyter.org/.

Knuth, Donald E. 1984. "Literate Programming." *The Computer Journal* 27 (2): 97111.

Kozlowski, Diego, Pablo Tiscornia, Guido Weksler, German Rosati, and Natsumi Shokida. 2020. *Eph: Argentina's Permanent Household Survey Data and Manipulation Utilities*. https://holatam.github.io/eph/.

Kuhn, Max, Hadley Wickham, and Emil Hvitfeldt. 2024. *Recipes: Preprocessing and Feature Engineering Steps for Modeling*. https://github.com/tidymodels/recipes.

Landau, William Michael. 2018. "The Drake r Package: A Pipeline Toolkit for Reproducibility and High-Performance Computing." *Journal of Open Source Software* 3 (21). https://doi.org/10.21105/joss.00550.

———. 2021. "The Targets r Package: A Dynamic Make-Like Function-Oriented Pipeline Toolkit for Reproducibility and High-Performance Computing." *Journal of Open Source Software* 6 (57): 2959. https://doi.org/10.21105/joss.02959.

Lumley, Thomas. 2011. *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons.

———. 2024. "Survey: Analysis of Complex Survey Samples."

Mailund, Thomas. 2017. *Advanced Object-Oriented Programming in r: Statistical Programming for Data Science, Analysis and Finance*. SPRINGER.

Merkel, Dirk. 2014. "Docker: Lightweight Linux Containers for Consistent Development and Deployment." *Linux Journal* 2014 (239): 2.

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. "Model Cards for Model Reporting." In, 220–29. https://doi.org/10.1145/3287560.3287596.

Prabhu, Anirudh, and Peter Fox. 2020. "Reproducible Workflow," December. http://arxiv.org/abs/2012.13427.

Publishing, Quarto. 2024. *Quarto*. https://www.quartoknows.com/.

Python Software Foundation. 2024. *Python 3 Documentation: Venv - Creation of Virtual Environments*. Python Software Foundation. https://docs.python.org/3/library/venv.html.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Sandve, Geir Kjetil, Anton Nekrutenko, James Taylor, and Eivind Hovig. 2013. "Ten Simple Rules for Reproducible Computational Research." *PLOS Computational*

Biology 9 (10): e1003285. https://doi.org/10.1371/journal.pcbi.1003285.

Särndal, Carl-Erik, Bengt Swensson, and Jan Wretman. 2003. *Model Assisted Survey Sampling*. Springer Science & Business Media.

Schneider, Benjamin. 2023. "Svrep: Tools for Creating, Updating, and Analyzing Survey Replicate Weights." https://CRAN.R-project.org/package=svrep.

Stodden, Victoria, Friedrich Leisch, and Roger D. Peng. 2014. *Implementing Reproducible Research*. CRC Press.

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

Ushey, Kevin, and Hadley Wickham. 2023. *Renv: Project Environments*. https://CRAN.R-project.org/package=renv.

Vargas, Mauricio. 2024. *Casen: Metodos de Estimacion Con Disenio Probabilistico y Estratificado En Encuesta CASEN (Estimation Methods with Probabilistic Stratified Sampling in CASEN Survey)*. https://pacha.dev/casen/.

Vilhuber, Lars. 2020. "Reproducibility and Replicability in Economics." *Harvard Data Science Review* 2 (4). https://doi.org/10.1162/99608f92.4f6b9e67.

Walker, Kyle, and Matt Herman. 2024. *Tidycensus: Load US Census Boundary and Attribute Data as 'Tidyverse' and 'Sf'-Ready Data Frames*. https://walker-data.com/tidycensus/.

Wickham, Hadley. 2011. "Testthat: Get Started with Testing." *The R Journal* 3: 510. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

———. 2019. *Advanced r, Second Edition*. CRC Press.

Wickham, Hadley, Peter Danenberg, Gábor Csárdi, and Manuel Eugster. 2024. *Roxygen2: In-Line Documentation for R*. https://roxygen2.r-lib.org/.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. https://dplyr.tidyverse.org.

Wickham, Hadley, Davis Vaughan, and Maximilian Girlich. 2024. *Tidyr: Tidy Messy Data*. https://tidyr.tidyverse.org.

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

Pass in `urlcolor:` in yaml. Or set these in the include-in-header file.

If you want to completely hide the links, you can use:

{`\hypersetup{allcolors=.}`}, or even better:

{`\hypersetup{hidelinks}`}.

If you want to have obvious links in the PDF but not the printed text, use:

{`\hypersetup{colorlinks=false}`}.