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 título Licenciado en Estadística

Resumen

metasurvey

por Mauro Loprete

El presente trabajo presenta *metasurvey*, un paquete de R diseñado para mejorar el procesamiento de encuestas por muestreo y la estimación de parámetros poblacionales junto con sus respectivos errores estándar. Utiliza meta-programación y técnicas de remuestreo, evaluar la incertidumbre de las estimaciones de los diferentes parámetros 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.

El tribunal docente integrado por los abajo firmantes aprueba el 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.

Mauro Loprete

Tutora: Natalia da Silva

Licenciatura en Estadística

Calificación:

Fecha: 20/12/2024

Tribunal:

Ignacio Alvarez-Castro _____

Juan Pablo Ferreira_____

Natalia da Silva _____

Agradecimientos

Con profundo agradecimiento y emoción, quiero comenzar reconociendo a mi tutora, Natalia da Silva. Desde la clase de Nuevas Tecnologías para el Análisis de Datos, ella despertó en mí una pasión por el aprendizaje continuo y me motivó a explorar la importancia del uso y desarrollo de herramientas computacionales en la profesión de un estadístico. Además, Natalia siempre ha sido para mí una referente en lo que respecta al aprendizaje estadístico y el desarrollo de la comunidad de R, demostrando su compromiso con la innovación y el crecimiento de nuestra disciplina. Su apoyo incondicional, paciencia y disposición para guiarme en cada etapa de este proyecto fueron fundamentales, así como su confianza en mis capacidades. Más adelante, tuve la oportunidad de ser ayudante en la nueva versión llamada Ciencia de Datos con R, lo que me permitió continuar creciendo en este apasionante campo.

Quiero extender mi gratitud a todos los profesores de la Licenciatura en Estadística, quienes me brindaron las herramientas necesarias para llevar a cabo este trabajo. En particular, agradezco a Alejandra Marroig, cuyas ideas y consejos enriquecieron este proyecto al vincularlo con casos de uso de encuestas continuas de hogares. También a Ramón Álvarez, quien confió en mis capacidades desde el principio, permitiéndome aplicar herramientas computacionales que no solo alimentaron este proyecto, sino que también me dieron la oportunidad de compartir ese conocimiento en charlas internas en el IESTA.

A mis compañeros de la Licenciatura en Estadística, quiero agradecerles por los momentos compartidos a lo largo de esta carrera. Juntos atravesamos largas jornadas de estudio, debates y aprendizaje mutuo, mientras este proyecto comenzaba a tomar forma. En especial, quiero destacar a mis amigos Ana Vignolo y Maximiliano Saldaña, quienes siempre estuvieron allí para escucharme, brindarme su apoyo y ayudarme a mantener el enfoque en los momentos más desafiantes.

A mis compañeros de trabajo en Cognus, les agradezco profundamente por su comprensión y respaldo, permitiéndome equilibrar mis responsabilidades laborales con la dedicación necesaria para este proyecto. Su apoyo y ánimo fueron esenciales para alcanzar este objetivo.

No puedo dejar de mencionar a mis amigos, quienes contribuyeron desde sus propias perspectivas. A Fabricio Machado, por sus ideas, consejos y la posibilidad de probar ejemplos reales basados en encuestas continuas de hogares; a Matías Sesser, quien, aunque nunca entendió del todo qué trataba este proyecto, siempre me ofreció su apoyo incondicional y valiosos consejos sobre herramientas computacionales; y a Giannina, quien siempre estuvo dispuesta a escucharme en los momentos más difíciles, cuando la desmotivación me afectaba, dándome ánimo para seguir adelante. Su paciencia y comprensión significaron mucho para mí.

También quiero expresar mi más profundo agradecimiento a mis alumnos, quienes han sido una constante fuente de inspiración. Sus preguntas, curiosidad y entusiasmo me recordaron la importancia de seguir aprendiendo y compartiendo conocimientos. Cada clase fue una oportunidad de crecimiento mutuo, y su energía me impulsó a dar lo mejor de mí. Este trabajo también es para ellos, porque fueron y seguirán siendo una parte fundamental de mi camino.

Finalmente, quiero agradecer profundamente a mi familia, quienes siempre me brindaron su apoyo constante. A mi padre, Oscar, y a su esposa, Gabriela, quienes siempre me dieron confianza, aliento y amor incondicional. También a mis hermanos

y sobrinos, quienes llenaron mi camino de cariño y energía, aunque debo admitir que, como el resto de la familia, nunca terminaron de entender del todo qué hace exactamente un estadístico. Sin embargo, su apoyo y celebraciones en cada pequeño avance hicieron toda la diferencia. Y de manera muy especial, a mi sobrina Florencia, quien siempre estuvo dispuesta a escucharme en mis momentos de duda y frustración, dándome ánimo y fuerza para seguir adelante. Este logro también es de todos ellos.

A todos, gracias por ser parte de este viaje tan significativo en mi vida.

Índice

Lista de Tablas

Lista de códigos

Capítulo 1

Introducción

Las encuestas por muestreo aleatorios se consolidan como instrumentos esenciales en la investigación estadística, facilitando la obtención de información detallada sobre poblaciones de interés a partir de muestras 'representativas'. No obstante, el procesamiento y análisis de estos datos enfrentan desafíos significativos, al derivar indicadores que involucran revisar a fondo los formularios y metadatos de la encuesta donde para algunos indicadores esto puede cambiar de forma continua. tales como tasas de mercado laboral, ingresos salariales o índices de pobreza (Vilhuber, 2020). La complejidad inherente a estos procesos puede propiciar errores de interpretación que puede llevar a interpretaciones erróneas y obstaculizar la reproducibilidad y transparencia de los resultados.

1.1 Encuestas por muestreo en R

Actualmente, existen diversos paquetes en R orientados al análisis de encuestas por muestreo, como `survey` (Lumley, 2024), `srvyr` (Freedman Ellis and Schneider, 2024), `gustave` (Chevalier, 2023), `vardpoor` (Breidaks, Liberts, and Ivanova, 2020), `svrep` (Schneider, 2023) y `weights`. No obstante, estos no abordan el proceso de creación de variables a partir de los formularios de las encuestas, lo que obliga a los usuarios a realizar este procedimiento manualmente cada vez que desean obtener un indicador. Por otra parte, herramientas específicas para encuestas particulares, como `ech` (Detomasi, 2020) para la Encuesta Continua de Hogares de Uruguay (ECH), `eph` (Kozlowski et al., 2020) para la Encuesta Permanente de Hogares de Argentina (EPH), `tidycensus` (Walker and Herman, 2024) para el Censo de Estados Unidos y `casen` (Vargas, 2024) para la Encuesta CASEN de Chile, presentan limitaciones en cuanto a flexibilidad y transparencia, y son sensibles a cambios en la estructura de las encuestas.

La Tabla 1.1 presenta un análisis detallado de los principales paquetes de R utilizados para el análisis de encuestas por muestreo. Esta comparación revela aspectos fundamentales de cada implementación. En cuanto a los paquetes base, `survey` se destaca como el componente fundamental para el análisis de encuestas complejas, implementando los estimadores más comunes y sirviendo como dependencia clave para otros paquetes, mientras que `srvyr` ofrece una interfaz más moderna usando `tidyverse`, aunque esto implica incorporar múltiples dependencias.

En el ámbito de los paquetes especializados en varianza, gustave implementa métodos específicos como Deville y Tillé, mientras que `vardpoor` se enfoca en el método de último conglomerado. Por su parte, `svrep` provee métodos de replicación como Bootstrap y Jackknife. En cuanto a paquetes para análisis específicos, `convey` se especializa en medidas de desigualdad y se integra eficientemente con el diseño muestral.

Un aspecto crucial que se evidencia en la tabla Tabla 1.1 es que, si bien cada paquete tiene sus fortalezas específicas, ninguno aborda completamente la necesidad de reutilización de procesos de limpieza y transformación, visualización del flujo de trabajo, ni documentación integrada del proceso. Esta comparación fue fundamental para decidir las dependencias de metasurvey, optando por survey como motor principal de estimación y svrep para métodos de replicación, mientras se evitan dependencias innecesarias que podrían comprometer la mantenibilidad.

1.2 Monitores y plataformas de datos públicos

En Uruguay existen numerosos portales de datos abiertos o monitores de indicadores publican estadísticas derivadas de encuestas por muestreo. Es usual que los resultados no detallen el proceso de construcción o recodificación de los indicadores, lo que dificulta la reproducibilidad y transparencia de los análisis. Esta situación también aparece en artículos académicos que, al trabajar con datos de encuestas, no especifican la metodología empleada en la obtención de los resultados, limitándose a referenciar la fuente de datos.

Es fundamental que los usuarios puedan obtener estimaciones puntuales y sus errores asociados de manera sencilla y confiable. Sin embargo, es común que se reporten estimaciones puntuales sin una medida de calidad o, en el peor de los casos, que se incorporen errores estándar calculados incorrectamente, sin considerar el diseño muestral adecuado. Esto puede llevar a una subestimación o sobrestimación de la variabilidad de la estimación, afectando la interpretación de los resultados y la construcción de intervalos de confianza.

Algunos de los principales monitores disponibles en Uruguay son:

- El **Instituto Nacional de Estadística (INE)** con sus monitores especializados:
  - Monitor de Mercado de Trabajo
  - Monitor de Ingresos de hogares y personas
  - Mercado de trabajo por área geográfica de referencia
- El **Ministerio de Desarrollo Social (MIDES)** a través de su Observatorio Social
- La **Oficina de Planeamiento y Presupuesto (OPP)** mediante el Observatorio Territorio Uruguay
- La **Agencia Nacional de Investigación e Innovación (ANII)** con su portal PRISMA

Por otra parte, el Instituto de Economía (IECON) de la Facultad de Ciencias Económicas y de Administración (FCEA) de la Universidad de la República (UdelaR) cuentan con un trabajo de compatibilización de Encuestas Continuas de Hogares (Instituto de Economía, Universidad de la República 2020) para construir y homogeneizar series de tiempo de indicadores de mercado laboral, ingresos y pobreza.

Si bien estas plataformas cumplen un rol fundamental en la diseminación de información estadística, presentan limitaciones significativas en términos de documentación metodológica y capacidades de reproducción. El trabajo del (IECON) sobre la compatibilización de Encuestas Continuas de Hogares (Instituto de Economía, Universidad

Cuadro 1.1: Comparación de paquetes para análisis de encuestas

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`.	2025-02-03T16:07:57Z
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`.	2024-10-17T13:19:17Z
svrep	Extensión del paquete `survey` para generar 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** en el caso de necesitar generar nuevas replicas de los pesos muestrales, aunque esto no es necesario para la mayoría de los casos o no se cuenta con la información necesaria para realizarlo.	2025-02-09T17:00:02Z

Datos extraídos desde GitHub a través de la API.

de la República 2020) representa un avance importante, pero la ausencia de herramientas estandarizadas para la documentación y reproducción de procesos sigue siendo un desafío pendiente.

Además, la integración de datos provenientes de múltiples fuentes institucionales a menudo requiere procesos de limpieza y transformación manuales, lo que no solo incrementa la carga de trabajo de los analistas sino que también introduce potenciales sesgos y errores humanos. La falta de una estandarización en los procedimientos de procesamiento de datos complica la comparación de resultados entre diferentes estudios y limita la capacidad de realizar análisis longitudinales robustos.

Este contexto se resalta la necesidad de desarrollar herramientas que no solo faciliten el acceso a los datos, sino que también promuevan la transparencia metodológica y la reproducibilidad de los análisis. En este sentido, `metasurvey` surge como una respuesta innovadora, orientada a intentar superar estas limitaciones mediante la automatización y estandarización de los procesos de limpieza y transformación de los datos.

1.3 Aporte del trabajo

El análisis del ecosistema actual revela cuatro limitaciones principales: la dificultad en la reutilización de procesos, la falta de visualización del flujo de trabajo, la rigidez ante cambios estructurales y la opacidad en la implementación. `metasurvey` se desarrolló como respuesta a estas limitaciones, adoptando un enfoque de meta-programación que permite abstraer los procesos de transformación en elementos reutilizables, visualizar y documentar el flujo de trabajo, mantener la flexibilidad ante cambios en las estructuras de datos y aprovechar las capacidades de paquetes establecidos como `survey` y `svrep`.

En este contexto, el presente trabajo introduce el desarrollo de `metasurvey`, un paquete innovador en R diseñado para simplificar y agilizar el procesamiento de encuestas por muestreo. `metasurvey` proporciona a científicos sociales, una herramienta robusta para transformar microdatos en indicadores de manera transparente, flexible y reproducible. Al ofrecer funciones avanzadas para la construcción de variables sintéticas y y trabajar con variables continuas, el paquete supera las limitaciones de las herramientas existentes, permitiendo a los usuarios validar y comprender el proceso de construcción de indicadores de forma modular y clara.

Es crucial que este proceso sea accesible y comprensible para los usuarios, dado que la transformación de microdatos en indicadores, demanda un conocimiento profundo de las encuestas tanto en el diseño como en los formularios, conocimiento que no siempre está ampliamente distribuido. A pesar de los esfuerzos previos por facilitar este procesamiento, muchas herramientas disponibles carecen de flexibilidad y transparencia, y son sensibles a cambios en la estructura y variables de los microdatos de las encuestas, lo que dificulta su actualización y adaptación.

1.3.1 Aspectos estadísticos de metasurvey

En el ámbito de la inferencia estadística de poblaciones finitas, resulta esencial considerar la incertidumbre y los errores asociados a las estimaciones producto del diseño muestral. Con frecuencia, estos aspectos son ignorados por usuarios no expertos en metodología de muestreo, lo que puede conducir a conclusiones erróneas. `metasurvey` aborda esta problemática al permitir obtener estimaciones puntuales y sus errores

asociados en la misma implementación, ofreciendo herramientas para evaluar la confiabilidad de las estimaciones mediante coeficientes de variación, intervalos de confianza y otros indicadores, sin requerir un conocimiento profundo en estimación de varianzas.

Es pertinente distinguir entre los enfoques de inferencia estadística basados en modelos (model-based inference) y aquellos fundamentados en el diseño muestral (design-based inference), según lo expuesto por Valliant, Dorfman, and Royall (, 2000) . Mientras que la inferencia basada en el diseño asume que la población es finita y fija, y que la aleatoriedad junto a las propiedades de los estimadores proviene del proceso de selección muestral, la inferencia basada en modelos considera que los valores de la población son realizaciones de un modelo probabilístico subyacente.

La utilización de pesos muestrales permite obtener estimaciones puntuales insesgadas en la mayoría de los casos, su papel en la estimación de varianzas y medidas de incertidumbre, como errores estándar e intervalos de confianza, es más complejo. Métodos como el método linealizado de Taylor o los métodos de replicación consideran no solo los pesos muestrales, sino también la estructura del diseño de la muestra, como la estratificación y la conglomeración. En este sentido, aunque los pesos w_i son esenciales para la estimación puntual, su impacto en la estimación de varianzas depende del método utilizado, lo que debe considerarse al interpretar los errores estándar y los intervalos de confianza.

En países como Uruguay, numerosos portales de datos abiertos o monitores de indicadores publican estadísticas derivadas de encuestas por muestreo sin detallar el proceso de construcción o recodificación de los indicadores, lo que dificulta la reproducibilidad y transparencia de los análisis. Esta situación es similar en artículos académicos que, al trabajar con datos de encuestas, no especifican la metodología empleada en la obtención de los resultados, limitándose a referenciar la fuente de datos.

Es fundamental que los usuarios puedan obtener estimaciones puntuales y sus errores asociados de manera sencilla y confiable. Sin embargo, es común que se reporten estimaciones puntuales sin una medida de calidad o, en el peor de los casos, que se incorporen errores estándar calculados incorrectamente, sin considerar el diseño muestral adecuado. Esto puede llevar a una subestimación o sobrestimación de la variabilidad de la estimación, afectando la interpretación de los resultados y la construcción de intervalos de confianza.

En el contexto del muestreo el **error estándar (SE)** de un estimador $\hat{\theta}$ varianza del estimador bajo el diseño muestral. En términos generales, el **SE** se define como:

$$SE(\hat{\theta}) = \sqrt{\text{Var}(\hat{\theta})}$$

donde $Var\hat{\theta}$ se calcula considerando la estructura del diseño de la muestra, que puede incluir estratificación, conglomerados y pesos muestrales. Métodos como la **aproximación de Taylor**, el **bootstrap** y las **técnicas de replicación (Jackknife, BRR)** permiten obtener estimaciones adecuadas de esta varianza.

`metasurvey` permite superar estas limitaciones al proporcionar herramientas integrales para el cálculo seguro y transparente de estimaciones puntuales junto con sus errores estándar correctamente ajustados al diseño muestral, asegurando inferencias válidas y precisas.

1.3.2 Aspectos sobre la implementación de metasurvey

El desarrollo de un paquete en R como `metasurvey` requiere una idea bien definida y los medios adecuados para llevarla a cabo. Es vital contar con una metodología de trabajo ordenada, heredada del desarrollo de software convencional, ya que para la publicación y difusión del paquete se deben cumplir estrictos estándares de calidad y documentación. En este sentido, `metasurvey` ha sido desarrollado siguiendo las mejores prácticas de desarrollo de software, incorporando conceptos avanzados de programación orientada a objetos, programación funcional y metaprogramación para brindar flexibilidad y potencia al usuario.

El enfoque que permite la flexibilidad en la construcción de indicadores es la **metaprogramación**. Este paradigma de programación posibilita que un programa modifique su propia estructura en tiempo de ejecución. En R, la metaprogramación se implementa a través de funciones como `eval()`, `parse()`, `substitute()`, `do.call()` y `quote()`, que permiten evaluar y manipular código de manera dinámica. `metasurvey` utiliza la metaprogramación para ofrecer funciones de alto nivel que facilitan la transformación de microdatos en indicadores. En particular, se ha adoptado una aproximación similar a la del paquete `recipes` de la librería `tidymodels` (Kuhn, Wickham, and Hvitfeldt, 2024), donde se emplean "recipes" y "steps" para definir secuencias de operaciones de preprocesamiento de datos.

En `metasurvey`, una **recipe** encapsula una serie de transformaciones a aplicar sobre los datos. Cada **step** representa una transformación específica, permitiendo a los usuarios construir pipelines de procesamiento modulares y fácilmente comprensibles. Esta estructura proporciona una gran flexibilidad, ya que se pueden añadir, modificar o eliminar steps según sea necesario, adaptándose a distintos tipos de encuestas y requerimientos analíticos.

1.4 Estructura del documento

Este documento se estructura de la siguiente manera: en el siguiente capítulo se presenta un marco conceptual detallado sobre el muestreo de poblaciones finitas y los paradigmas de programación utilizados en el desarrollo de `metasurvey`, incluyendo una explicación más profunda sobre el uso de recipes y steps en la metaprogramación. Posteriormente, se profundiza en los antecedentes relacionados con metodologías de estimación de varianzas y se examinan otros paquetes en R que han servido de base para su creación. Finalmente, se ofrecen ejemplos prácticos de cómo utilizar `metasurvey` para construir indicadores de mercado laboral a partir de los microdatos de la **ECH** y, para demostrar su flexibilidad, se incluye un ejemplo con la **EPH**. [1]

[1] Este documento puede leerse en su formato de página web o en su formato de documento PDF. Tanto el código fuente del paquete como el de este documento se encuentran disponibles públicamente en los repositorios de GitHub. Para la realización de este documento se utilizó `quarto` (Publishing, 2024), que permite escribir texto junto con código R.

Capítulo 2

Marco teórico

Este capítulo dara una introducción a diferentes conceptos para la comprensión de la inferencia estadística en el contexto de muestreo de poblaciones finitas, un pilar fundamental en la generación de indicadores sociales y estadísticas oficiales. Se presenta una progresión sistemática y rigurosa desde los conceptos fundamentales hasta las técnicas más sofisticadas de estimación de la varianza, con especial énfasis en su aplicación práctica en encuestas socioeconómicas complejas.

El marco teórico se estructura en tres componentes principales interrelacionados:

1. La teoría fundamental de inferencia en muestreo, incluyendo los principios probabilísticos subyacentes.
2. Los métodos de estimación de parámetros poblacionales y sus propiedades estadísticas asintóticas.
3. Las técnicas específicas para el tratamiento de diseños muestrales complejos y sus implicaciones metodológicas.

Esta base teórica rigurosa resulta fundamental no solo para comprender la implementación de los métodos de estimación de varianza en el paquete `metasurvey`, sino también para garantizar la validez estadística y robustez de las inferencias realizadas.

2.1 Inferencia en muestreo de poblaciones finitas

Las encuestas por muestreo constituyen la fuente principal para la construcción de indicadores socio-demográficos y económicos en la estadística oficial contemporánea. Realizar inferencias estadísticas a partir de encuestas implica estimar parámetros poblacionales a partir de muestras finitas, lo que introduce una serie de desafíos metodológicos y técnicos que deben ser abordados de manera rigurosa y sistemática.

El proceso de inferencia no solo implica la estimación puntual de parámetros de interés, sino también la cuantificación de la variabilidad y la incertidumbre asociada a estas estimaciones, ya sea por medio de intervalos de confianza, pruebas de hipótesis o medidas de precisión como el error estándar (**SE**), el coeficiente de variación (**CV**) o el efecto diseño (**deff**) (Särndal, 1992), entre otros, determinados por la estructura del diseño muestral. El efecto diseño (*deff*) es una medida que compara la varianza de un estimador bajo un diseño muestral complejo con la varianza del mismo estimador bajo muestreo aleatorio simple con el mismo tamaño de muestra. Un *deff* mayor que 1 indica que el diseño muestral introduce mayor variabilidad en la estimación en comparación con el muestreo aleatorio simple, mientras que un *deff* cercano a 1 sugiere que la eficiencia del diseño es similar a la del muestreo aleatorio simple.

2.1.1 Diseño muestral

El diseño muestral representa la piedra angular del proceso inferencial en poblaciones finitas, constituyendo el mecanismo probabilístico que determina la selección de unidades muestrales y, consecuentemente, las propiedades estadísticas fundamentales de los estimadores derivados.

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 $s = \{u_1, u_2, \cdots, u_n\}$ y S es el conjunto de todos los subconjuntos de U de tamaño n, el cual comúnmente denominamos **muestra**, el diseño muestral puede definirse de la siguiente forma:

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

$$\sum_{s \in S} p(s) = 1$$

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 más comunes:.

- **Diseño Aleatorio Simple (SI)**

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

$$p(s) = \begin{cases} \binom{N}{n}^{-1} & \text{si } s \in S \\ 0 & \text{en otro caso} \end{cases}$$

Ejemplo: Si se tiene una población de 1000 individuos y se desea seleccionar una muestra de 100 de manera aleatoria, cada combinación de 100 individuos tiene la misma probabilidad de ser seleccionada.

- **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. Los estratos son subgrupos homogéneos dentro del universo, definidos de tal manera que dentro de cada estrato las unidades son más similares entre sí que con las unidades de otros estratos. Esto permite obtener estimaciones más precisas al reducir la variabilidad dentro de cada estrato.

Realizados los estratos tenemos a la población U dividida en H estratos $U_1, U_2, \cdots, U_H$ donde $U = U_1 \cup U_2 \cup \cdots \cup U_H$ y $U_i \cap U_j = \emptyset$ para $i \neq j$ y $N = N_1 + N_2 + \cdots + N_H$, donde N_h es el tamaño del estrato h y $N = \sum_{h=1}^{H} N_h$

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**, más conocido **STSI** (Stratified Simple Random Sampling). En este caso cada $p_h(s_h)$ es el diseño aleatorio simple en el estrato h. Cabe destacar que este diseño solo es posible cuando se realiza un muestreo directo de la población objetivo y se conoce la variable de estratificación.

2.1.2 Diseños Complejos

En algunas situaciones, el muestreo directo no es factible, ya sea porque no se dispone de un marco muestral adecuado, el costo de crearlo es demasiado elevado o porque la población objetivo no está distribuida de manera homogénea dentro del territorio. En estos casos, la selección aleatoria de individuos resulta complicada, y se recurre a diseños muestrales más complejos, como el muestreo por conglomerados o el muestreo por etapas.

- **Muestreo por Conglomerados**

En el muestreo por conglomerados, en lugar de seleccionar individuos directamente, se eligen grupos o conglomerados de elementos, denominados Unidades Primarias de Muestreo (UPM o PSU). Luego, dentro de cada conglomerado seleccionado, se incluyen todos los elementos de la población.

Un ejemplo común en encuestas de hogares es definir las UPM como manzanas o sectores censales. La selección de estos conglomerados puede realizarse mediante diferentes estrategias, como muestreo aleatorio simple, muestreo proporcional al tamaño o muestreo estratificado.

Este diseño tiene la ventaja de reducir los costos de recolección de datos, ya que permite concentrar el trabajo de campo en áreas específicas, disminuyendo los recursos necesarios para desplazamientos. Sin embargo, presenta una desventaja en términos de eficiencia estadística, ya que suele generar estimaciones con mayor varianza en comparación con un muestreo aleatorio simple. Además, el tamaño final de la muestra puede ser incierto hasta que se lleva a cabo el muestreo, ya que depende del número de elementos dentro de los conglomerados seleccionados.

- **Muestreo por Etapas**

En el muestreo por etapas, la selección de unidades se realiza de manera secuencial, en varias fases. En cada etapa, se selecciona una muestra de unidades, y en la siguiente etapa se elige otra muestra dentro de las unidades previamente seleccionadas.

Este diseño es muy utilizado en encuestas de hogares. Por ejemplo, en una encuesta de tres etapas: 1. Se seleccionan conglomerados (ej. sectores censales). 2. Dentro de cada conglomerado, se elige una muestra de hogares. 3. Finalmente, dentro de los hogares seleccionados, se selecciona una muestra de personas.

El muestreo por etapas es útil cuando los conglomerados son demasiado grandes y se requiere una mejor distribución de la muestra. Además, permite controlar el tamaño final de la muestra, ajustando la cantidad de unidades seleccionadas en cada etapa.

En términos de eficiencia, este diseño es más preciso que el muestreo por conglomerados cuando la variable de interés es homogénea dentro de los conglomerados, pero menos eficiente que el muestreo aleatorio simple. Además, el cálculo de los errores estándar en este tipo de diseño es más complejo. Dependiendo del método de selección empleado en cada etapa, el cálculo exacto de la varianza puede ser difícil o incluso imposible. Por esta razón, se suele recurrir a aproximaciones lineales o a técnicas

de remuestreo, como el Bootstrap o el Jackknife, para estimar la variabilidad de las estimaciones.

2.1.3 Probabilidades de Inclusión y el Estimador Horvitz-Thompson

Una vez definido el concepto de diseño muestral, es posible determinar 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 manera:

2.1.3.1 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. Estas variables indicadoras son útiles para analizar el comportamiento de los estimadores bajo el diseño muestral y permiten definir estimadores en U y no en S. Es claro que $I_k \sim Bernoulli(\pi_k)$ y que su esperanza es $E(I_k) = Pr(I_k) = \pi_k$.

Esta probabilidad es fundamental, ya que constituye la base para la construcción de estimadores insesgados. En este contexto, es posible definir el **estimador de Horvitz-Thompson (HT)** para estimar un total $t = \sum_1^N t_k$, de la siguiente manera:

$$\hat{t}_y = \sum_{k=1}^{n} \frac{y_k}{\pi_k}$$

Este estimador, propuesto por Horvitz y Thompson en 1952, es insesgado en el diseño, en el sentido de que $E(\hat{t}_y) = t$. Sin embargo, no es muy utilizado en la práctica, ya que rara vez se emplean los ponderadores originales w_i. Para más detalles sobre sus propiedades, se pueden consultar (Särndal, 1992) y (Horvitz and Thompson, 1952).

2.1.4 Ponderación basada en el diseño y estimadores más comunes

En general, el concepto de **ponderador** se utiliza para realizar estimaciones de totales, medias, proporciones, varianzas, entre otros. En este sentido, el ponderador basado en el diseño muestral se define como:

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

Este ponderador puede interpretarse como el número de individuos que representa el elemento k en la población. A partir de esta ponderación, se pueden definir estimadores para medias, proporciones y razones.

Inferencia sobre el tamaño de la población

Una vez definidos los estimadores, es posible observar que los estimadores de medias y proporciones son casos particulares del **estimador de razón**. Un detalle importante es que se asume que N es fijo pero desconocido. Por ello, al calcular proporciones, se ajusta el total en función de un estimador del tamaño de la población:

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

Existen ciertos diseños muestrales en los que se cumple que $\sum_{k=1}^{N} w_k = N$. En estos casos, el estimador de la media poblacional $y_U = \frac{\sum_U y}{N}$ y el estimador de proporciones se convierten en casos particulares del estimador de total. Así, se puede definir el estimador de la media como:

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

2.1.5 Estimadores específicos

Estimador de una media

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

Este estimador se utiliza en encuestas de hogares cuando se desea estimar, por ejemplo, el ingreso promedio de los hogares de manera anual o mensual.

Estimador de una proporción

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

Este estimador es útil para calcular proporciones, como la fracción de hogares con acceso a internet en una región. Dado que se aplica a variables binarias, el estimador de proporción es un caso particular del estimador de media descrito anteriormente.

Estimador de una razón

Si se desea estimar la razón $R = \frac{\sum_{k=1}^{N} y_k}{\sum_{k=1}^{N} z_k}$, donde y_k y z_k son variables de interés en la muestra, se puede definir el estimador de razón de la siguiente forma:

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

El estimador de razón se emplea en la construcción de indicadores del mercado laboral, como la **tasa de desempleo** y la **tasa de ocupación**, entre otros.

2.1.6 Medidas de incertidumbre y errores estándar

2.1.6.1 Momentos muéstrales y estimadores de varianza

Para un estadístico θ y un estimador en la muestra s θ_s 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 la varianza del estimador 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 identidad para simplificar cálculos:

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

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:

$$\begin{aligned} V(\hat{t}_y) &= \sum_U (y_k w_k)^2 + \sum_U \sum_{k \neq l} \Delta_{kl} \times y_k w_k \times y_l w_l \\ &= \sum_U \sum \Delta_{kl} \times y_k w_k \times y_l w_l \end{aligned}$$

Podemos verificar que este estimador de varianza es insesgado con las 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:

$$V(\hat{\bar{y}}) = \frac{1}{N^2} \times \sum_U \sum_{k \neq l} \Delta_{kl} \times y_k w_k \times y_l w_l$$

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 optarse por métodos de estimación de varianzas alternativos 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 muestral**. En la práctica, el marco muestral es imperfecto y se debe considerar la no respuesta, la cobertura y la falta de actualización del marco. En general, los microdatos publicados incluyen 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 más 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 práctica, para diseños complejos no se dispone de las probabilidades de selección de segundo orden, insumo principal para calcular los errores estándar como se expuso en las formulaciones anteriores. Por esto, se requiere optar por metodologías alternativas como el método del último conglomerado, método de remuestreo como Jackknife, 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 estimadores anteriormente mencionados. Pero para realizar un proceso de inferencia completo se requiere calcular sus errores estándar, construir intervalos de confianza y/o poder medir la estabilidad de nuestros resultados. En este sentido, es importante tener al alcance herramientas que permitan realizar este tipo de cálculos, ya que en diferentes softwares estadísticos junto a la estimación puntual se presentan los errores estándar asumiendo diseños sencillos ya sea por omisión del usuario, por limitaciones de los paquetes estadísticos o por no tener información suficiente para calcularlos.

En base a lo expuesto en la sección anterior, es posible definir los errores estándar de los estimadores de forma teórica. Sin embargo, en la práctica, la estimación de la varianza de los estimadores es un problema complejo, especialmente en diseños de muestreo por conglomerados o en varias etapas de selección. En estos casos, las probabilidades de inclusión de segundo orden son difíciles de calcular y se requieren métodos alternativos para estimar la varianza de los estimadores.

Cada estimador tiene asociado un error estándar que permite cuantificar la variabilidad de la estimación, debido a que la muestra es aleatoria esta medida es una variable aleatoria. Dentro de la incertidumbre puede separarse en errores muestrales y no muestrales. Los primeros refieren a la variabilidad de la estimación debido a la selección de la muestra y los segundos refieren a la variabilidad de la estimación debido a errores de medición, errores de no respuesta, cobertura, entre otros (Särndal, 1992).

En este trabajo se centra en la estimación de los errores muestrales, ya que los errores no muestrales son difíciles de cuantificar. Los errores muestrales se pueden cuantificar mediante la varianza de la estimación. Esta varianza depende del diseño muestral ya que, como se mencionó anteriormente, el diseño muestral induce propiedades estadísticas claves como la distribución en el muestreo, valores esperados y varianzas de estimadores poblacionales. El paquete `survey` permite estimar la varianza de la estimación de forma sencilla y eficiente, sin embargo, en algunos casos la estimación de la varianza no es correcta, ya que el paquete `survey` asume un muestreo simple con probabilidades de inclusión desiguales y con reposición, es decir, con una fracción de muestreo $f = \frac{n}{N} \approx 0$ (Lumley, 2004).

Para diseños multietápicos, las probabilidades de segundo orden son muy complejas de calcular, por lo que una estimación directa no es muy factible. Además, estos ponderadores no son exactamente los pesos muestrales definidos en los capítulos anteriores, ya que se ajustan para tener en cuenta la no respuesta y la calibración, lo cual permite una estimación más precisa de ciertas variables de interés. En el caso

de que se cuente con un mecanismo para obtener las probabilidades de inclusión de segundo orden, este no tendría en cuenta el proceso posterior de calibración, por lo que la estimación de la varianza no sería correcta.

En general, para este tipo de casos se utilizan principalmente las siguientes estrategias: el método del último conglomerado, donde se asume que la variabilidad proviene únicamente de la selección en la primera etapa, y métodos de remuestreo como el Bootstrap o Jackknife. En este trabajo se propone la implementación de forma nativa de diferentes métodos utilizando solamente un argumento al cargar la encuesta, permitiendo a usuarios no expertos en metodología de muestreo obtener estimaciones de varianzas correctas y confiables.

Adicionalmente, para estimadores no lineales se utiliza el método de Linearización de Taylor, que permite aproximar el estimador como función de estimadores lineales. Un caso típico es la tasa de desempleo, que se calcula como el cociente entre la población desocupada y la población económicamente activa. En este caso, se puede aproximar la tasa de desempleo como función de estimadores lineales y obtener una estimación de la varianza de la tasa de desempleo o, de forma similar, un estimador de medias o proporciones.

2.1.7 Métodos de remuestreo

La estimación del error estándar de una media u otros resúmenes poblacionales se basa en la desviación estándar de dicho estimador a través de múltiples muestras independientes. Sin embargo, en encuestas reales solo se cuenta con una muestra. El enfoque de **pesos replicados** ofrece una alternativa, al calcular la variabilidad del estimador a partir de múltiples subconjuntos que se comportan de manera parcialmente independiente, y luego extrapolar esta variabilidad para obtener una estimación que se asemeje a la que se obtendría si se tuvieran múltiples muestras independientes.

Réplicas de Mitad de Muestra

Para entender mejor este método, se puede considerar un diseño estratificado en el que se seleccionan dos unidades por estrato. Si se dividen los datos en dos mitades, tomando una unidad de cada estrato, se crean subconjuntos que se pueden considerar como “mitades” independientes. Si la corrección por población finita no es relevante, la varianza de un estimador basado en una mitad de muestra es aproximadamente el doble de la varianza de la muestra completa. Dado que se tienen dos mitades, se puede usar la diferencia entre sus estimaciones para calcular la varianza:

$$\mathrm{Var}(\hat{\theta}) \approx \frac{1}{2}(\hat{\theta}_A - \hat{\theta}_B)^2,$$

donde $\hat{\theta}_A$ y $\hat{\theta}_B$ son las estimaciones de cada mitad de la muestra. Este enfoque es sencillo pero puede ser inestable, por lo que se suelen usar múltiples conjuntos de divisiones para obtener un promedio más preciso.

Balanced Repeated Replication (BRR)

El método de **Balanced Repeated Replication (BRR)** es una forma sistemática de elegir subconjuntos de la muestra, garantizando que cada unidad se incluya de manera equilibrada en las réplicas. Esto se logra mediante un balanceo ortogonal, donde cada observación está presente en aproximadamente la mitad de las réplicas,

y cada par de unidades de diferentes estratos aparece en las réplicas de forma equilibrada. Con (K) estratos, se puede generar un conjunto de hasta (K + 4) réplicas que produzca una estimación de la varianza que es prácticamente idéntica a la que se obtendría usando todas las (2^K) combinaciones posibles.

La varianza utilizando BRR se calcula así:

$$\mathrm{Var}_{\mathrm{BRR}}(\hat{\theta}) = \frac{1}{R}\sum_{r=1}^{R}(\hat{\theta}_r - \hat{\theta})^2,$$

donde R es el número de réplicas seleccionadas y $\hat{\theta}_r$ es el estimador obtenido de cada réplica.

Pesos Replicados en Diseños Multietápicos y Complejos

El enfoque de pesos replicados no solo se aplica a diseños simples, sino que también se adapta a **diseños de muestreo multietápicos** y **diseños complejos**. En estos casos, la estructura de la muestra se complica, ya que puede involucrar varias etapas de selección (por ejemplo, seleccionar primero conglomerados como municipios, luego hogares dentro de los municipios, y finalmente personas dentro de los hogares).

Para estos diseños, se utilizan métodos como el **Jackknife** y el **Bootstrap**, que permiten manejar la estructura multietápica. Por ejemplo:

- Con el método de **Jackknife**, se ajustan los pesos eliminando una observación o un conglomerado completo en cada réplica, y recalculando el estimador con los datos restantes. Esto puede ajustarse para considerar la estructura de estratos y conglomerados.

$$\mathrm{Var}_{\mathrm{JK}}(\hat{\theta}) = \frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \hat{\theta})^2$$

donde (n) es el número de observaciones o conglomerados, $\hat{\theta}_i$ es la estimación obtenida cuando se omite la i-ésima unidad, y $\hat{\theta}$ es la estimación con todos los datos.

- En el **Bootstrap**, se seleccionan subconjuntos con reemplazo de cada conglomerado, y se ajustan los pesos según el número de veces que cada unidad aparece en la réplica. Esto es especialmente útil cuando las unidades de muestreo tienen una estructura jerárquica, como es el caso de los diseños multietápicos.

$$\mathrm{Var}_{\mathrm{Boot}}(\hat{\theta}) = \frac{1}{B}\sum_{b=1}^{B}(\hat{\theta}_b - \hat{\theta})^2,$$

donde B es el número de réplicas y $\hat{\theta}_b$ es el estimador obtenido en la b-ésima réplica.

Ventajas de los Pesos Replicados

Aunque estos métodos requieren más esfuerzo computacional comparados con métodos tradicionales como el estimador de Horvitz-Thompson, son muy versátiles. Facilitan la estimación de errores estándar para diferentes estimadores, no solo para medias o totales, y son especialmente útiles cuando se trabaja con diseños de muestreo

complejos. Además, permiten obtener errores estándar precisos para estimaciones de subpoblaciones sin necesidad de ajustes adicionales. Esto los convierte en una herramienta poderosa para el análisis de encuestas complejas, especialmente con el soporte de software estadístico moderno.

El paquete `survey` con `svrep` proporciona una implementación robusta de varios métodos de pesos replicados, incluyendo Balanced Repeated Replication (BRR), Jackknife, y Bootstrap. Sin embargo, el uso adecuado de estos métodos a menudo no es tan conocido por usuarios que no son expertos en muestreo. La correcta especificación del diseño y la interpretación de los resultados pueden ser complejas, especialmente en el caso de diseños de muestreo multietápicos o aquellos que requieren calibración.

Dentro de `metasurvey` se busca simplificar el uso de estos métodos, pudiendo especificar el tipo de réplica deseado con un solo argumento al cargar la encuesta o utilizar réplicas brindadas por la institución que publica los microdatos. Además, se busca incorporar medidas de calidad de las estimaciones como el coeficiente de variación, el error relativo y el error absoluto, para facilitar la interpretación de los resultados y la comparación entre diferentes estimaciones y subpoblaciones.

Medidas de calidad de las estimaciones

Brindar medidas de calidad de las estimaciones es fundamental para evaluar la confiabilidad de los resultados y comparar diferentes estimaciones. Estas medidas permiten cuantificar la precisión de los estimadores y evaluar la variabilidad de los resultados. Algunas de las medidas más comunes son el **coeficiente de variación** (CV) y el **efecto diseño** (*deff*), ya que en algunos casos, si bien se pueden obtener estimaciones, estas pueden no ser precisas o el tamaño de la muestra en el dominio de interés puede ser pequeño, lo que puede llevar a estimaciones poco confiables.

Herramientas para la estimación de varianza

Si bien para una persona con experiencia en muestreo estas medidas son familiares, para un usuario no experto, en caso de no contar con herramientas que permitan calcular estas medidas, se pueden pasar por alto. En este sentido, es importante contar con herramientas que permitan calcular estas medidas de forma sencilla y eficiente, para facilitar la interpretación de los resultados y la toma de decisiones.

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 Bootstrap o el Jackknife, existen diferentes ideas o propuestas como se menciona en (Deville and Tille, 1998) y (Deville and Tillé, 2005), que demuestran con resultados numéricos que estimadores del tipo **H-T** bajo un diseño balanceado pueden aproximarse desde el enfoque de regresión o calibración. Además, existen estimadores alternativos que complementan métodos de remuestreo para aproximar probabilidades de inclusión de segundo orden (Escobar and Berger, 2013) utilizando ciertas aproximaciones límites (Hajek, 1964).

En este sentido, `metasurvey` busca ser una herramienta que permita a usuarios no expertos en muestreo realizar análisis de encuestas de manera sencilla, proporcionando una interfaz amigable y herramientas que faciliten la interpretación de los resultados. Además, busca integrar diferentes métodos de estimación de varianzas y medidas de calidad de las estimaciones, para que los usuarios puedan obtener resultados confiables y precisos, sin necesidad de ser expertos en muestreo.

En el Capítulo 3 se presentarán conceptos relacionados al desarrollo de paquetes en R junto a la presentación de diferentes herramientas para desarrollar paquetes. Se mencionarán paquetes que permiten realizar estimaciones de varianzas y que complementarán el paquete `metasurvey` así como también paquetes similares o trabajos similares la sección 1.1. En el Capítulo 4 se presentará cómo se han integrado los conceptos de muestreo y estimación de varianzas, en conjunto con las dependencias e implementaciones realizadas en el paquete `metasurvey` para facilitar el proceso de obtención de indicadores socioeconómicos y demográficos.

Capítulo 3

Métodos computacionales

En este capítulo se presentan los antecedentes y conceptos aplicados en el desarrollo de `metasurvey`. También se abordan conceptos de investigación reproducible, la importancia de R como herramienta para el análisis de datos y la comunidad estadística, la revisión de paquetes para el procesamiento de encuestas por muestreo y la relevancia de la especificación diseño muestral. Se mencionan paquetes y herramientas en las cual fue inspirado el desarrollo del paquete y herramientas similares junto a sus limitaciones.

La generación de indicadores sociales se ha convertido en una tarea fundamental tanto para la toma de decisiones a nivel de una empresa, para la política pública así como para la investigación. Sin embargo, este proceso puede resultar complejo, ya que requiere conocimiento específico sobre el formulario de la encuesta y formas de definir ciertos índices o variables auxiliares que no necesariamente son triviales y dependen de cierta forma de la experiencia de los usuarios. Este proceso de generación de indicadores frecuentemente carece de transparencia o documentación adecuada, en parte por la ausencia de herramientas apropiadas y en parte por la falta de una cultura de reproducibilidad, ya que generalmente solo se hace referencia a los datos y no al proceso completo de generación de los indicadores.

3.1 Desarrollo de paquetes en R

R es un lenguaje de código abierto con una gran comunidad de usuarios en diversas áreas de investigación. Esto ha permitido el desarrollo de una extensa colección de paquetes que facilitan tareas de análisis de datos, visualización, bioinformática, aprendizaje automático y otras ramas afines a la estadística. El ecosistema de R se destaca por:

1. **Repositorios centralizados**: CRAN (Comprehensive R Archive Network) actúa como el principal repositorio de paquetes verificados y estables. Otros repositorios especializados incluyen Bioconductor para análisis biológicos y rOpenSci para ciencia abierta (Gentle, 2009).
2. **Estándares de calidad**: Los paquetes deben pasar rigurosas verificaciones antes de ser aceptados en CRAN, incluyendo pruebas en múltiples plataformas, documentación completa y coherencia en el código.
3. **Comunidad activa**: La comunidad de R mantiene foros activos, conferencias regulares (useR!, rstudio::conf) y grupos de trabajo que continuamente mejoran el ecosistema (R Core Team, 2023).

3.1.1 ¿Por qué desarrollar un paquete en R?

Desarrollar un paquete en R posee varias ventajas:

- **Reutilización de código**: Es posible que alguien ya haya escrito una función o paquete que es de utilidad. Por lo tanto, siempre es bueno buscar si existe algún paquete que ya tenga las funcionalidades requeridas.
- **Compartir código**: La comunidad de R es muy activa y siempre está dispuesta a compartir código, lo que fomenta el desarrollo continuo de paquetes.
- **Colaboración**: El trabajo colaborativo es esencial en el desarrollo de paquetes en R, permitiendo que diferentes personas aporten nuevas funcionalidades, correcciones de errores, entre otros.

3.1.2 Elementos básicos de un paquete en R

La estructura de un paquete en R sigue convenciones estrictas que facilitan su distribución y mantenimiento. Los componentes esenciales incluyen:

1. **Estructura de directorios**:
   - `R/`: Código fuente en R.
   - `man/`: Documentación en formato .Rd.
   - `src/`: Código en otros lenguajes (C++, Fortran).
   - `tests/`: Pruebas unitarias.
   - `data/`: Conjuntos de datos.
   - `vignettes/`: Tutoriales y ejemplos extensos.
2. **Archivos de control**:
   - `DESCRIPTION`: Metadatos del paquete.
   - `NAMESPACE`: Control de exportación de funciones.
   - `LICENSE`: Términos de uso y distribución.
   - `.Rbuildignore`: Archivos excluidos del build.
3. **Control de versiones**:
   - Sistema de versionado semántico (MAJOR.MINOR.PATCH).
   - Integración con Git/GitHub.
   - Registro de cambios en `NEWS.md`.

- **Documentación**: es esencial para que los usuarios puedan entender el funcionamiento de cada herramienta dentro del paquete. Se realiza utilizando el sistema de documentación de R, basado en comentarios en el código fuente.

- **Pruebas**: es importante que el paquete tenga pruebas que verifiquen que las funciones se comportan según lo esperado. Las pruebas se realizan utilizando el paquete *testthat* (Wickham, 2011).

- **Control de versiones**: permite llevar un registro de los cambios realizados en el paquete. El sistema de control de versiones más utilizado en la comunidad de R es `git` (Git, 2025).

- **Licencia**: Permite a los usuarios utilizar, modificar y distribuir el paquete. La licencia más utilizada en la comunidad de R es la licencia MIT (Open Source Initiative, 1988).

El proceso de subir un paquete a CRAN puede llegar a ser tedioso, ya que se deben cumplir ciertos requisitos revisados por los mantenedores de CRAN. Sin embargo, es un proceso valioso, ya que permite que el paquete sea utilizado por una gran cantidad de usuarios.

El proceso de chequeo fue automatizado por medio de 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 los requisitos para ser publicado. En caso de que no cumpla con los requisitos, se notifica los errores y el mismo no puede ser incluido en la rama principal del repositorio hasta que estos errores sean corregidos.

Todo el proceso y código fuente del paquete se encuentra disponible en el repositorio de GitHub del paquete. Si está interesado en colaborar con el desarrollo del paquete, puede consultar la guía de contribución.

3.2 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.

3.2.1 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.

3.2.2 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 mas utilizadas provienen del sistema de programación orientada a objetos llamado `S3` las definen utilizando la función `setClass` y los objetos se crean utilizando la función `new`. También se pueden utilizar las clases del sistema `S4` aunque este tiene una sintaxis mas compleja y no es tan utilizado.
- **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 la creación de una clase específica 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 definen las clases `Step`, `Recipe` y `Survey`, entre otras, 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 intuitiva y eficiente, además de permitir la herencia de clases y la definición de métodos y atributos de manera sencilla.

3.2.3 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 se puede ver en Código 3.1.

Código 3.1 Se define una función que toma dos argumentos `x` e `y` y retorna la suma de los mismos.

```
function(x, y) {
  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.

3.3 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, más 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 obtener los mismos resultados de un estudio, experimento o un cálculo en particular. Si bien en la actualidad la reproducibilidad se incentiva en algunas revistas científicas es de suma importancia en otras instancias donde se comparten datos, se monitorean indicadores entre otro. Cuando se comparten datos o ya sea de una encuesta o datos recolectados para una investigación, rara vez se documenta y rara se menciona o explica el código fuente para hacer las transformaciones. Aún compartiendo el código fuente, esto aún no es 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 más 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 específicas 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.

3.3.1 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 especificando 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.

3.3.2 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," 2024) 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.

3.3.3 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.

3.4 Herramientas para el desarrollo de paquetes en R

Cualquier usuario puede desarrollar un paquete en R, aunque existen diferentes guías y estándares para el desarrollo de paquetes en R, como se menciona en ("R Packages (2e)," n.d.), además de la guía de rOpenSci (rOpenSci et al., 2024) que promueve la creación de paquetes que sean de utilidad para la comunidad científica definiendo estándares de calidad y documentación.

Para el desarrollo de `metasurvey` se utilizaron paquetes como `usethis` (Wickham, Bryan, et al., 2024) que permite la creación de paquetes en R, `roxygen2` (Wickham, Danenberg, et al., 2024) que permite la documentación de funciones y la creación de manuales, `testthat` (Wickham, 2011) que permite la creación de tests automatizados, `pkgdown` (Wickham, Hesselberth, et al., 2024) que permite la creación de sitios web para paquetes de R, `devtools` (Wickham et al., 2022) que permite la instalación y la carga de paquetes en R, `renv` (Ushey and Wickham, 2023) que permite la creación de ambientes locales con versiones especificas de paquetes de R junto a herramientas herramientas como pre-commit (Sottile and Contributors, 2024) que permite la ejecución de scripts antes de realizar un commit en un repositorio de git esto con el fin de mantener la calidad del código y la documentación antes de realizar un cambio en el repositorio. De forma conjunta se utilizó GitFlow (Driessen, 2010) que es una metodología de trabajo con git que permite la colaboración y la integración continua de los cambios en un repositorio de git. Para la automatización de los tests en diferentes sistemas operativos se utilizó GitHub Actions (*GitHub Actions: Automate Your Workflow*, 2024) que permite la ejecución de scripts en diferentes sistemas operativos y la generación de reportes de cobertura de código con Codecov (*Codecov: Code Coverage Insights*, 2024).

Todo estas herramientas permiten que la creación de paquetes sea sencilla y permita a los usuarios enfocarse en la implementación con cierto grado de calidad y documentación, además de permitir la colaboración y la integración continua de los cambios en un repositorio de git.

Implementación de tests automatizados

El paquete incluye tests automatizados creados con `testthat`, lo que asegura la robustez del código fuente al ser utilizado en diversos contextos. Un ejemplo es el test de la función `extract_time_pattern`, que analiza el nombre de una encuesta y retorna su tipo, año y periodicidad. Esta función clave utiliza expresiones regulares para manejar distintos formatos de tiempo y es fundamental para tareas como definir la edición de encuestas o emparejar réplicas bootstrap. Su implementación completa puede encontrarse aquí: extract_time_pattern.

Al tener una serie de test automatizados como el presentado en el código 3.2 permite que los mismos se ejecuten ante una nueva versión del código fuente y poder revisar si implementaciones anteriores funcionan de la misma manera luego de realizar estos

Código 3.2 Ejemplo de un test automatizado para verificar el correcto funcionamiento de la función extract_time_pattern.

```
test_that(
  "Probar extraer time pattern anual",
  {
    testthat::expect_equal(
      metasurvey:::extract_time_pattern("ECH_2023"),
      list(
        type = "ECH",
        year = 2023,
        periodicity = "Annual"
      )
    )
  }
)
```

cambios. Para el envió a CRAN se realizan estos test automático y una serie de pruebas de calidad y documentación para que el paquete sea aceptado en CRAN. Actualmente `metasurvey` no tiene errores, mensajes de advertencia o notas en CRAN, la cobertura del código es baste baja ya que realizar test para todas las funciones es un trabajo arduo y que requiere tiempo, sin embargo en futuras versiones se espera tener una cobertura del código mayor al 80%. La cobertura puede ser verificada en el siguiente enlace: codecov.

Documentación

Para la documentación se utilizó `roxygen2` que permite la documentación de funciones y la creación de manuales, además de `pkgdown` que permite la creación de sitios web para paquetes de R, esto permite que los usuarios puedan tener una guía de referencia para utilizar el paquete y que los desarrolladores puedan tener una guía de referencia para la implementación de nuevas funciones o la modificación de las existentes.

Como se puede observar en el código 3.3, la documentación de las funciones se realiza con comentarios en el código fuente, esto permite que `roxygen2` pueda generar la documentación de las funciones y los manuales de manera automática, simplemente hay que añadir comentarios con ciertas etiquetas que dependiendo si la función es exportada o no, es un requisito para la aceptación en CRAN que las funciones que se exporten estén documentadas y que la documentación sea clara y concisa. La documentación puede ser consultada en el siguiente enlace: Documentación de metasurvey o en la ayuda de R con `?load_survey`.

Código 3.3 Extracto de la documentación de la función `load_survey` con las etiquetas necesarias para la generación de la documentación.

```
#' @title Load survey
#' @param path Path to the survey file
#' @param svy_type Type of survey
#' @param svy_edition Edition of the survey
#' @param svy_weight Weight of the survey
#' @param svy_psu Primary sampling unit
#' @param ... Additional arguments
#' @param bake Logical
#' @return Survey object
#' @keywords preprocessing
#' @export
load_survey <- function(
    path = NULL,
    svy_type = NULL,
    svy_edition = NULL,
    svy_weight = NULL,
    svy_psu = NULL,
    ..., bake = FALSE) {
      # Ejemplo no mostrado debido a la extensión del
      # código puede ser consultado en
      # el repositorio de GitHub
}
```

Pruebas en diferentes sistemas operativos y versiones de R junto a GitHub Actions

En muchos casos, los paquetes de R pueden tener problemas de compatibilidad con diferentes versiones de R o con diferentes sistemas operativos, para evitar estos problemas se utilizó GitHub Actions que permite la ejecución de scripts en diferentes sistemas operativos y la generación de reportes de cobertura de código con Codecov. En el caso de `metasurvey` se realizan pruebas en diferentes versiones de R y en diferentes sistemas operativos como Windows, MacOS y Linux, esto permite que el paquete sea compatible con diferentes versiones de R y sistemas operativos.

Todo esto fue realizado en GitHub Actions donde se define un archivo de configuración que permite definir en que situaciones se deben de ejecutar los test junto a las diferentes plataformas y versiones de R. En este caso al utilizar GitFlow que es una metodología de trabajo con git que permite la colaboración y la integración continua de los cambios en un repositorio de git, se puede tener una rama de desarrollo y una rama de producción, donde en la rama de desarrollo se realizan los cambios y se ejecutan los test y en la rama de producción se realiza la publicación en CRAN. Todo esto permite que el paquete sea robusto y que los cambios sean integrados de manera continua en el repositorio. Para la integración de una nueva versión se realizan pull request que son un pedido de integración de cambios en la rama de desarrollo, esto permite que los cambios sean revisados y auditados antes de ser integrados en la rama principal.

Como se puede observar en el código 3.4, se define un archivo de configuración que permite definir en que situaciones se deben de ejecutar los test junto a las diferentes

plataformas y versiones de R, esto es lanzado automáticamente cuando se hace un cambio en la rama de desarrollo o en la rama de producción puede aquí verse el historial y ejemplos del mismo paquete.

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.

Código 3.4 Archivo de configuración de GitHub Actions para la ejecución de tests en diferentes sistemas operativos y versiones de R.

```yaml
on:
  push:
    branches:
      - main
      - develop
  pull_request:
    branches: [develop]

name: R-CMD-check

jobs:
  R-CMD-check:
    runs-on: ${{ matrix.config.os }}

    name: ${{ matrix.config.os }} (${{ matrix.config.r }})

    strategy:
      fail-fast: false
      matrix:
        config:
          - {os: macos-latest,   r: 'release'}
          - {os: windows-latest, r: 'release'}
          - {os: ubuntu-latest,   r: 'devel', http-user-agent: 'release'}
          - {os: ubuntu-latest,   r: 'release'}
          - {os: ubuntu-latest,   r: 'oldrel-1'}

    env:
      GITHUB_PAT: ${{ secrets.GITHUB_TOKEN }}
      R_KEEP_PKG_SOURCE: yes

    steps:
      - uses: actions/checkout@v4

      - uses: r-lib/actions/setup-pandoc@v2

      - uses: r-lib/actions/setup-r@v2
        with:
          r-version: ${{ matrix.config.r }}
          http-user-agent: ${{ matrix.config.http-user-agent }}
          use-public-rspm: true

      - uses: r-lib/actions/setup-r-dependencies@v2
        with:
          extra-packages: any::rcmdcheck
          needs: check

      - uses: r-lib/actions/check-r-package@v2
        with:
          upload-snapshots: true
```

Capítulo 4

Metodología y desarrollo

En este capítulo se divide en dos partes, la primera parte se centra en la metodología de los métodos de estimación de los errores estándar (SE), donde se describen los diferentes métodos de replicas de los pesos muestrales y se explican las ventajas del uso de los pesos replicados. La segunda parte se centra en el desarrollo e implementación de las diferentes partes del paquete. Los ejemplos de código se muestran con código real del paquete `metasurvey` y donde allí se explican las diferentes partes del código y su funcionamiento intentando ser lo más expositivo posible para que el lector pueda entender el funcionamiento de la meta-programación aunque estos no sean los ejemplos más complejos que se encuentran en el paquete.

4.1 Desarrollo e Implementación

El desarrollo de `metasurvey` se fundamenta en tres pilares principales: una gestión eficiente de dependencias, una arquitectura modular basada en clases, y el uso de técnicas avanzadas de meta-programación. A continuación, se detalla cada uno de estos aspectos.

4.1.1 Gestión de Dependencias

El diseño de `metasurvey` prioriza la eficiencia y minimalismo en sus dependencias externas. El núcleo del paquete se apoya en survey para el procesamiento de encuestas complejas, data.table para la manipulación eficiente de datos mediante referencias, y `R6` para la implementación orientada a objetos. Para mejorar la legibilidad del código se utiliza `glue`, mientras que `jsonlite` y `httr` facilitan la comunicación con la API. Esta selección cuidadosa contrasta con versiones anteriores que incluían dependencias más pesadas como son `tidyverse` y `rlang`, priorizando ahora la eficiencia y ligereza del paquete.

4.1.2 Arquitectura y Diseño de Clases

El paquete utiliza una arquitectura orientada a objetos mediante `R6`, implementando las siguientes clases principales:

1. **Clase Survey**
   - Almacena la encuesta, metadatos y diseño muestral.
   - Mantiene registro de recetas y pasos aplicados.
2. **Clase Step**
   - Define transformaciones individuales.
   - Gestiona dependencias y evaluación perezosa.

3. **Clase Recipe**
   - Agrupa steps y metadatos del proceso.
   - Facilita compartir flujos de trabajo.
   - Incluye DOI para citación académica.
4. **Clase RotativePanelSurvey** y **PoolSurvey**
   - Extensiones para casos específicos (paneles rotativos y combinación de encuestas)
   - Implementan estimadores específicos para estos casos y permiten calcular medias moviles.

La arquitectura modular facilita la extensión del paquete y protege los objetos mediante encapsulamiento.

4.1.3 Implementación de Meta-programación

La meta-programación en `metasurvey` cumple tres objetivos principales:

1. **Registro Automático de Transformaciones**
   - Cada operación realizada en una encuesta se registra automáticamente
   - Facilita la replicabilidad y transparencia de los análisis
2. **Evaluación Perezosa**
   - Las transformaciones se aplican solo cuando es necesario
   - Optimiza el rendimiento y uso de memoria
3. **Gestión de Dependencias**
   - Identificación automática de variables dependientes
   - Asegura la correcta aplicación de transformaciones en el orden adecuado

Implementar la evaluación perezosa y dependencias resultó un gran desafío, ya que requirió un diseño cuidadoso de los métodos para poder trabajar con las expresiones de R y extraer información de las mismas. La meta-programación se encuentra en casi toda la implementación de los `Step` y de las funciones auxiliares que permiten la creación y aplicación de recetas y pasos. Dentro de esta misma sintaxis se permite extraer las expresiones para luego poder compartir las recetas o poder decidir en que momento se aplican los pasos y recetas a la encuesta.

4.1.4 Ejemplos de la implementación

Carga de una encuesta

El paquete puede dividirse en dos partes principales: la carga y procesamiento de encuestas, y la estimación de los errores estándar. Dentro de lo que es la carga y procesamiento de las encuestas, se incluyen funciones para cargar las encuestas en diferentes formatos, como son SPSS, STATA, CSV, y RDS, y para realizar operaciones básicas como la selección de variables, la recodificación de categorías, y la creación de indicadores.

Esta implementación puede verse en load_survey.R donde aquí se define la función principal `load_survey` esta misma carga la encuesta y realiza las operaciones básicas mencionadas anteriormente. Dentro de ella se puede ver que es simplemente un wrapper de diferentes paquetes para cargar la encuesta ya sea `read.spss` del paquete `foregin` (R Core Team, 2023) para cargar encuestas provenientes en formato SAV o DTA, `fread` de `data.table` (Barrett et al., 2024) para archivos CSV y por último loadWorkbook del paquete `openxlsx` (Schauberger and Walker 2024), todas

estas funciones se encargan de cargar la encuesta en base a la extensión del archivo, el usuario puede modificar cambiando el `engine` como por ejemplo a `tidyverse` donde la lectura CSV se realiza con `read_csv` del paquete `readr` (Wickham, 2023), o `haven` (Wickham, Miller, and Smith, 2023) para cargar encuestas en formato SPSS o STATA.

Al cargar la encuesta el usuario debe de especificar el tipo de encuesta que está cargando y la edición de la misma como el tipo de encuesta y la edición de la misma como se puede ver en Código 4.1, estos metadatos serán cruciales para poder obtener recetas y pasos de la API de metasurvey. Además, se puede especificar el tipo de réplica que se desea utilizar, por defecto se utiliza el método de BRR, pero el usuario puede especificar el método de réplica que desee, ya sea Jackknife o Bootstrap, en el Capítulo 5 se especifica como utilizar replicas brindadas por la institución que publica los microdatos y estimadores de medias móviles.

Una vez definida la carga de datos, dentro de la misma implementación se crea un objeto de la clase `Survey`, la cual, se encuentra definida en survey.R. Esta clase es realizada con el paquete `R6` (Chang, 2022) y se encarga de almacenar los microdatos de la encuesta, los metadatos, las recetas y los pasos junto al diseño muestral, el usuario puede obtener información con wrappers de cada método para que sea más sencillo de utilizar, como por ejemplo `cat_steps` donde se obtiene todos los pasos que fueron aplicados a la encuesta, `cat_recipes` donde se obtienen todas las recetas que fueron aplicadas a la encuesta, `cat_design` donde se obtiene el diseño muestral, entre otros.

En el código 4.1 se muestra un ejemplo de cómo se carga una encuesta y se obtiene la información de la misma.

Código 4.1 Lectura de la encuesta ECH 2022, fijación del ponderador y obtención de recetas.

```
library(metasurvey)

# Cargar encuesta

## Encuesta ECH 2022
## Se fija el ponderador de la encuesta
## Se obtienen las recetas de la encuesta

ech_2022 <- load_survey(
  metasurvey::load_survey_example(
    "ech",
    "ech_2022"
  ),
  svy_edition = "2022",
  svy_type = "ech",
  svy_weight = add_weight(annual = "w_ano"),
  recipes = get_recipe(
    "ech",
    "2022"
  )
)
```

Dentro de este mismo ejemplo se puede ver que se fija el ponderador de la encuesta, en este caso se fija el ponderador `w_ano` que es el ponderador anual de la encuesta ECH 2022, este ponderador es crucial para la estimación de errores estándar y para la obtención de recetas y pasos de la encuesta.

Clase `Step`

La clase `Step` es una clase que se encarga de almacenar los pasos que se aplican a la encuesta, esta clase se encuentra definida en step.R y se encarga de almacenar los pasos que se aplican a la encuesta, los pasos se aplican a través de recetas que se obtienen de la API de metasurvey, los pasos pueden ser de diferentes tipos, como por ejemplo `step_compute` que se encarga de calcular una variable, `step_recode` que se encarga de recodificar una variable, entre otros.

En el Código 4.2 se muestra un ejemplo de cómo se crea un objeto de la clase `Step` con una clase de `R6` (Chang, 2022) paquete que permite la programación orientada a objetos en R donde tiene aquí se encuentran los atributos de la clase `Step` como `name`, `edition`, `survey_type`, `type`, `new_var`, `exprs`, `call`, `svy_before`, `default_engine`, `depends_on`, `comments`, `bake` en conjunto con el método `initialize` que se encarga de inicializar la clase `Step` con los atributos mencionados anteriormente. Los objetos `Survey`, `Recipe`, `PanelRotativeSurvey` y `PoolSurvey` son clases que se encuentran definidas en el paquete `metasurvey` donde modelar en una clase los diferentes objetos que se utilizan en el paquete permite una mejor organización y estructura del código y facilita la implementación de la meta-programación.

Es importante mencionar que la clase `Step` es una clase que se utiliza internamente en el paquete y no es necesario que el usuario la utilice directamente, sin embargo, es importante mencionarla ya que es una parte esencial del paquete y es utilizada en la implementación de las recetas y pasos de la encuesta, además de que es un ejemplo claro de cómo se puede utilizar la programación orientada a objetos en R para modelar diferentes objetos y clases.

El principal uso de `R6` y no de `S3` o `S4` es que `R6` permite la creación de objetos con estado en conjunto con su propiedad de encapsulamiento, lo cual es esencial para la implementación de la meta-programación, ya que permite almacenar el estado de los objetos y modificarlos a través de métodos, lo cual es esencial para la implementación de las recetas y pasos de la encuesta. Además se puede manejar las copias de los objetos de forma más sencilla siendo algo que mejora el rendimiento y la eficiencia del paquete.

Lazy evaluation

La evaluación perezosa es una técnica de programación que consiste en retrasar la evaluación de una expresión hasta que sea necesaria. En el contexto de `metasurvey`, la evaluación perezosa se utiliza para retrasar la aplicación de recetas y pasos a la encuesta hasta que sea necesario, lo cual permite optimizar el rendimiento y la eficiencia del paquete. Esto hace que los `Steps` se ejecuten con una validación mínima en base a las dependencias de las variables, y se ejecuten solo cuando se necesiten o al cocinar la receta.

El paquete tiene por defecto la evaluación perezosa de los pasos y recetas, lo cual hace que sea más eficiente y rápido el procesamiento de las encuestas aunque puede llevar a confusiones al usuario cuando revisa los datos de forma manual ya que si bien se

Código 4.2 Definición de la clase `Step` con sus atributos y método `initialize` para inicializar la clase. Las clases de `R6` son encapsuladas y permiten que los métodos y atributos sean privados o públicos.

```
step <- R6Class(
  "Step",
  public = list(
    name = NULL,
    edition = NULL,
    survey_type = NULL,
    type = NULL,
    new_var = NULL,
    exprs = NULL,
    call = NULL,
    svy_before = NULL,
    default_engine = NULL,
    depends_on = list(),
    comments = NULL,
    bake = NULL,
    initialize = function(...args) {
      self$name <- name
      # Inicialización de los atributos de la clase
      # Se omiten por la extensión del código
      # Definir un constructor permite que siempre
      # se inicialicen los atributos de la clase
      # de forma correcta
    },
    method1 = function(...) {
      # Método 1 (Método generico no se muestra por extensión)
    },
    method2 = function(...) {
      # Método 2 (Método generico no se muestra por extensión)
    }
  )
)
```

aplican los `Step` los mismos no tienen efecto hasta que se cocine la receta, el usuario puede desactivar (código 4.3) la evaluación perezosa de los pasos y recetas si lo desea. La evaluación perezosa se implementa a través de la clase `Step` y de la función `bake` que se encarga de aplicar los pasos y recetas a la encuesta.

Código 4.3 Forma de desactivar la evaluación perezosa de los pasos y recetas. Esto hace que los pasos y recetas se apliquen de forma inmediata.

```
metasurvey::set_lazy_processing(FALSE)
```

Uso de copias y referencias

El paquete `data.table` (Barrett et al., 2024) logra una eficiencia en la manipulación de datos al utilizar referencias en lugar de copias. Dentro de `metasurvey` al utilizar como motor `data.table` permite que sea eficiente y rápido aunque puede llevar a confusiones al usuario donde se modifican las referencias de los objetos algo que no

es común en R, sin embargo, se puede utilizar la función `copy` para realizar copias de los objetos y no modificar las referencias de los mismos código 4.4.

Código 4.4 Desactivar el uso de referencias y utilizar copias de los objetos.

```
metasurvey::set_use_copy(TRUE)
```

4.1.5 Meta-programación

La meta-programación fue un aspecto clave en el desarrollo de `metasurvey`, ya que permitió que al realizar una operación en una encuesta, se generara un registro de los pasos y recetas aplicados, lo cual es esencial para la replicabilidad y la transparencia de los análisis. La meta-programación se implementó a través de la clase `Survey` y de las funciones auxiliares que permiten la creación y aplicación de recetas y pasos.

Código 4.5 Ejemplo de función para encontrar dependencias de variables en una expresión de un step. Puede encontrar más ejemplos en el código fuente en la implementación de los Steps.

```
find_dependencies <- function(call_expr, survey) {
  dependencies <- character()

  if (is.call(call_expr)) {
    for (i in seq_along(call_expr)) {
      result <- find_dependencies(call_expr[[i]], survey)
      if (!is.null(result)) {
        dependencies <- unique(c(dependencies, result))
      }
    }
  } else if (
    is.name(call_expr) && as.character(call_expr) %in% names(survey)
  ) {
    dependencies <- unique(c(dependencies, as.character(call_expr)))
  }

  return(unique(dependencies))
}
```

En el código 4.5 se muestra una función auxiliar que permite encontrar las dependencias de variables en una expresión de un `Step`. Esta función se utiliza para identificar las variables que se utilizan en un paso y que deben ser incluidas en el registro de recetas. La función `find_dependencies` recibe una expresión de un paso y un objeto de la clase `Survey`, y devuelve un vector con las variables que se utilizan en el paso. Esta función se utiliza de forma interna en las implementaciones de `step_compute` y `step_recode` para registrar las dependencias esto es un ejemplo claro de como con código se puede extraer información del mismo y utilizarla para otros fines.

La meta-programación también fue utilizada para que el usuario utilice la misma sintaxis del paquete `survey` para definir los parámetros o indicadores a estimar y no deba de incluir el diseño el cual ya se encuentra dentro del objeto `Survey`, donde aquí también se encuentran diferentes diseños en base a la periodicidad de la encuesta, algo que es útil cuando se trabaja con encuestas de panel rotativos como la ECH luego de cambio de metodología en 2021, en el Capítulo 5 se menciona como se puede

utilizar diferentes diseños en base a la periodicidad de los microdatos o parámetros es estimar y un ejemplo con la encuesta ECH.

Ejemplos Avanzados de Meta-programación

La meta-programación avanzada en R permite manipular ambientes y expresiones de forma dinámica, lo cual es esencial para la implementación de recetas y pasos en `metasurvey`. A continuación, se presentan casos típicos de meta-programación, la manipulación de ambientes y expresiones, y el uso de `substitute` y `bquote` para evaluar expresiones de forma parcial.

Ambientes y Evaluación

Código 4.6 En este ejemplo se crea un nuevo ambiente `env` y se asignan valores a las variables `x`

```
# Crear un nuevo ambiente
env <- new.env()

# Asignar valores en el nuevo ambiente
assign("x", 10, envir = env)
assign("y", 20, envir = env)

# Evaluar una expresión en el nuevo ambiente
eval(quote(x + y), envir = env)
```

Manipulación de Expresiones

Código 4.7

```
# Crear una expresión
expr <- quote(x + y)

# Modificar la expresión
expr[[1]] <- quote(`*`)
expr[[2]] <- quote(x)
expr[[3]] <- quote(y)

# Evaluar la nueva expresión
eval(expr, envir = env)
```

Uso de `substitute` y `bquote`

Estos ejemplos muestran cómo se pueden manipular ambientes y expresiones en R para realizar meta-programación avanzada. La capacidad de modificar y evaluar expresiones dinámicamente es una herramienta poderosa que se utiliza en `metasurvey` para registrar y aplicar transformaciones y brindar la flexibilidad necesaria para adaptarse a diferentes tipos de encuestas y diseños muestrales.

4.1.6 Sistema de Comunicación y Optimización

El paquete implementa una API REST desarrollada en Node.js que permite compartir y gestionar recetas entre usuarios. La API proporciona endpoints para obtener, crear

Código 4.8 En este ejemplo se muestra cómo se pueden reemplazar variables en una expresión utilizando `substitute` y cómo se puede evaluar parcialmente una expresión utilizando `bquote`.

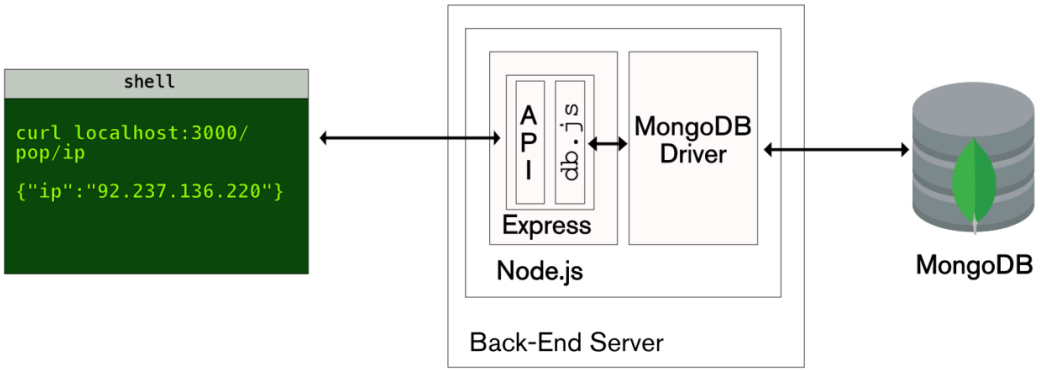
```
# Usar substitute para reemplazar variables en una expresión
expr <- quote(a + b)
substitute(expr, list(a = 1, b = 2))

# Usar bquote para evaluar parcialmente una expresión
bquote(a + .(b))
```

y actualizar recetas, almacenadas en MongoDB. El sistema incluye autenticación mediante tokens JWT, con planes futuros para implementar un portal web de registro automático.

MongoDB se eligió por su flexibilidad y escalabilidad, permitiendo almacenar recetas de forma estructurada y eficiente. Es un sistema de base de datos NoSQL (**Not Only SQL**) enfocada en documentos, en lugar de tablas, se agrupan en colecciones y se almacenan en formato JSON. Esto permite almacenar recetas de forma flexible y escalable, sin necesidad de definir un esquema fijo.

Si bien dentro de la configuración de la API se define un esquema semi-estructurado para las recetas, se permite la flexibilidad de agregar nuevos campos y estructuras de datos pero se mantiene una estructura mínima para mantener la consistencia de los datos. La API se comunica con el paquete `metasurvey` a través de solicitudes HTTP el usuario no necesita interactuar directamente con la API ya que el paquete se encarga de la comunicación con la misma.



Para optimizar el rendimiento, se implementó un sistema de cache que gestiona proactivamente la memoria mediante liberación de recursos y reutilización de cálculos intermedios. El sistema aprovecha la paralelización automática y el procesamiento por lotes cuando es posible.

La arquitectura modular del sistema facilita la incorporación de nuevos tipos de encuestas y diseños muestrales, permitiendo adaptarse a diferentes necesidades sin modificar el núcleo del paquete.

4.2 Síntesis del capítulo

En este capítulo se ha presentado los diferentes objetos dentro del paquete, así como su implementación en el paquete `metasurvey`. Además, se ha mostrado cómo

`metasurvey` facilita la creación y el manejo de encuestas, permitiendo a los usuarios obtener transformaciones transparentes y reproducibles.

Capítulo 5

Casos de uso

En este capítulo se presentan dos casos de uso que demuestran la versatilidad y eficacia de `metasurvey` en el análisis de encuestas de hogares llevadas a cabo bajo diseños complejos en América Latina. Se abordarán dos encuestas de la región: la Encuesta Continua de Hogares (ECH) de Uruguay y la Encuesta Permanente de Hogares (EPH) de Argentina. Estas encuestas son fuentes esenciales de información sobre el mercado laboral en sus respectivos países y han sido utilizadas en diversos estudios y análisis socioeconómicos.

5.1 Encuesta Continua de Hogares con Paneles Rotativos

5.1.1 Contexto y Cambios Metodológicos

En 2021, la ECH adoptó un diseño de panel rotativo, reemplazando su diseño **cross-section** anterior. Este cambio permite el seguimiento de hogares durante seis meses, tras lo cual se incorporan nuevos grupos de rotación en el panel, como se muestra en la Figura 5.1 (Instituto Nacional de Estadística, 2021). Esto mejora:

- La precisión de las estimaciones, en especial para brindar mayores aperturas para variables de mercado laboral.
- El análisis de dinámicas laborales.
- La medición de indicadores socioeconómicos.

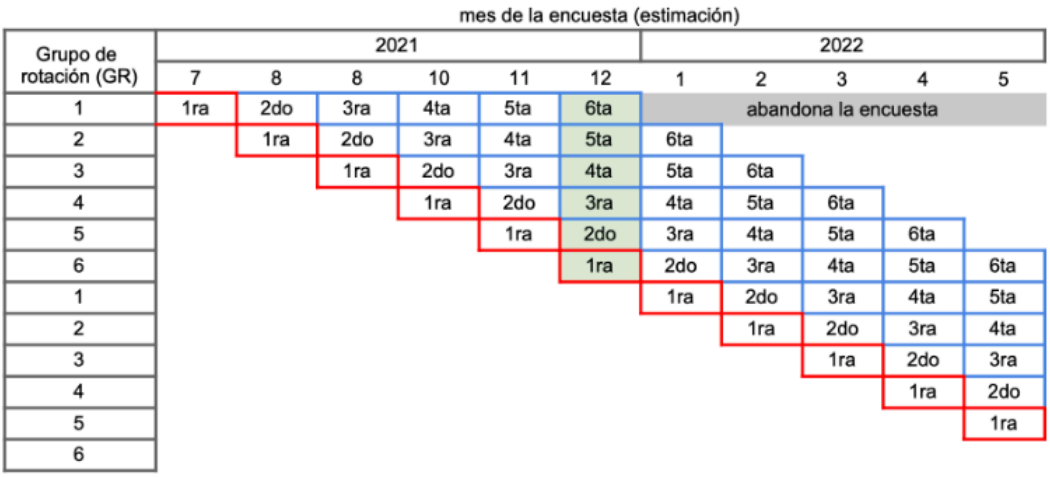


Figura 5.1: Paneles Rotativos, fuente documento metodológico de la ECH 2022

5.1.2 Carga y Procesamiento de Datos

La ECH es la principal fuente de información sobre el mercado laboral uruguayo desde 1968 y amplió su cobertura nacional en 2006. Los microdatos se encuentran disponibles desde 2006 en el portal ANDA del INE.

Para trabajar con la ECH 2023, se requieren:

- ECH_implantacion_2023.csv: Microdatos de implantación del panel rotativo. Aquí se tiene información de los hogares seleccionados para el panel y sus características junto a información relevada en la encuesta presencial.
- ECH Seguimiento[1-12].csv: Microdatos de seguimiento mensual del panel rotativo. Contienen el relevamiento de información relevada por vía telefónica, incluyendo características relacionadas al mercado laboral.
- Archivos de pesos replicados Bootstrap mensuales (enero-diciembre 2023): Archivos Excel con los pesos replicados para cada mes del año 2023, necesarios para el cálculo de los errores estándar en las estimaciones.

Un aspecto técnico importante es el manejo eficiente de los pesos muestrales replicados. `metasurvey` optimiza este proceso convirtiendo automáticamente los archivos Excel de pesos bootstrap a formato CSV, mejorando significativamente el rendimiento en análisis posteriores, un ejemplo de la carga de microdatos junto a las réplicas puede verse en Código 5.1.

5.1.3 Construcción de Variables e Indicadores

El paquete permite dos enfoques para la construcción de variables:

1. Utilizar recetas predefinidas disponibles a través de la API de `metasurvey`
2. Crear recetas personalizadas según necesidades específicas

Por ejemplo, para calcular indicadores del mercado laboral:

A continuación se presentan algunas recetas que se pueden utilizar para calcular las tasas de mercado de trabajo a nivel mensual, trimestral y anual.

En el código 5.3 se crean las variables necesarias para el cálculo de las tasas de mercado de trabajo a partir de los microdatos de seguimiento de la ECH 2023. Las variables creadas son: Población Económicamente Activa (PEA), Población Empleada (PET), Población Ocupada (PO) y Población Desocupada (PD). Una cosa importante es que la variable `POBPCOAC` es una variable construida por el INE que indica la condición de actividad, esta variable se construye a partir de los formularios dentro de `metasurvey` se encuentra a modo de receta la sintaxis publicada hasta la versión 2019 de la ECH, la misma fue basado en archivos PDF donde incluían un script de SPSS para la construcción de la variable. Para años posteriores no fue publicado este documento, se espera en breve dejar disponible la receta para los años 2020 considerando el nuevo formulario y revisando las versiones anteriores.

5.1.4 Estimación de Tasas de Mercado de Trabajo

Una vez creadas las variables básicas, podemos calcular las tasas de actividad, empleo y desempleo utilizando la función `workflow`. Esta función permite realizar estimaciones a diferentes niveles temporales (mensual, trimestral, anual) y con distintos tipos de agregación.

CUADRO 5.1: Tasas de mercado de trabajo a nivel trimestral a partir de las estimaciones mensuales de la ECH 2023 con sus respectivos errores estándar y coeficientes de variación.

Indicadores de Mercado de Trabajo
Estimaciones trimestrales a partir de datos mensuales

tasa	Indicador	Periodo	Valor	Error Estándar	CV	Evaluación
Tasa de Desempleo	survey::svyratio: pd/pea	Q1	8.4%	3.2%	0.3160	Utilizar con precaucion
Tasa de Desempleo	survey::svyratio: pd/pea	Q2	8.6%	3.0%	0.2973	Utilizar con precaucion
Tasa de Desempleo	survey::svyratio: pd/pea	Q3	8.1%	2.8%	0.2941	Utilizar con precaucion
Tasa de Desempleo	survey::svyratio: pd/pea	Q4	8.2%	2.8%	0.2894	Utilizar con precaucion
Tasa de Actividad	survey::svyratio: pea/pet	Q1	62.7%	2.4%	0.0320	Excelente
Tasa de Actividad	survey::svyratio: pea/pet	Q2	63.2%	2.1%	0.0275	Excelente
Tasa de Actividad	survey::svyratio: pea/pet	Q3	63.6%	2.1%	0.0273	Excelente
Tasa de Actividad	survey::svyratio: pea/pet	Q4	63.9%	2.1%	0.0275	Excelente
Tasa de Ocupación	survey::svyratio: po/pet	Q1	57.5%	2.4%	0.0348	Excelente
Tasa de Ocupación	survey::svyratio: po/pet	Q2	57.8%	2.1%	0.0307	Excelente
Tasa de Ocupación	survey::svyratio: po/pet	Q3	58.5%	2.0%	0.0289	Excelente
Tasa de Ocupación	survey::svyratio: po/pet	Q4	58.6%	2.0%	0.0281	Excelente

Fuente: Elaboración propia en base a ECH 2023

Estimación por trimestre

Como se puede ver en el código 5.4 se pueden realizar estimaciones a diferentes niveles temporales y con distintos tipos de agregación, en este caso se realizan estimaciones a nivel trimestral a partir de las estimaciones mensuales, el resultado de la estimación se puede ver en la Tabla 5.1.

Estimación con

También es posible realizar estimaciones a nivel anual, para esto se debe de considerar las medias de las estimaciones mensuales para cada trimestre y luego realizar la estimación anual.

La nueva metodología puede dar confusiones sobre que microdatos utilizar ya que para estimaciones anuales se consideran los microdatos de implantación y las referidas a mercado de trabajo se consideran los microdatos de seguimiento esto se debe a que el hogar luego de ser seleccionado en la implantación el formulario se realiza por vía telefónica y se centra unicamente a variables referidas al situación de empleo del hogar. Para realizar estimaciones por ejemplo de cantidad de personas que finalizaron un ciclo educativo se debe de considerar los microdatos de implantación a nivel anual.

Podemos ver que se obtiene un objeto donde se tiene las estimaciones de las tasas de mercado de trabajo a nivel mensual, trimestral y anual. En este caso se utilizan las replicas bootstrap para calcular los errores estándar de las estimaciones y se incluye una recomendación sobre la confiabilidad de las estimaciones en base al coeficiente de variación de las estimaciones.

5.2 Encuesta Permanente de Hogares de Argentina

Para demostrar la versatilidad de `metasurvey`, se presenta un segundo caso de uso con la Encuesta Permanente de Hogares (EPH) de Argentina. La EPH es una encuesta continua de hogares que se realiza en el país desde 1985 y es la principal fuente de

Cuadro 5.2: Tasas de mercado de trabajo a nivel anual a partir de las estimaciones mensuales de la ECH 2023 con sus respectivos errores estándar y coeficientes de variación considerando las replicas bootstrap.

Indicadores de Mercado de Trabajo
Estimaciones anuales a partir de datos mensuales

tasa	Indicador	Periodo	Valor	Error Estándar	CV	Evaluación
Tasa de Desempleo	survey::svyratio: pd/pea	2023	8.3%	3.5%	0.2992	Utilizar con precaucion
Tasa de Actividad	survey::svyratio: pea/pet	2023	63.4%	2.6%	0.0286	Excelente
Tasa de Ocupación	survey::svyratio: po/pet	2023	58.1%	2.5%	0.0306	Excelente

Fuente: Elaboración propia en base a ECH 2023

información sobre el mercado laboral argentino. Los microdatos de la EPH están disponibles en el portal del INDEC.

Podemos hacer algo similar a lo que hicimos con la ECH, en este caso cargamos los microdatos de la EPH 2022 trimestre 3 y creamos las variables necesarias para el cálculo de las tasas de mercado de trabajo. Estos `steps` fueron basados en base al formulario de la encuesta y la documentación disponible.

5.2.1 Visualización de las recetas

Algo interesante que se puede hacer con `metasurvey` es visualizar las recetas que se utilizan para la construcción de las variables, esto puede ser útil para entender como se construyen las variables y que variables dependen de otras. En este caso se muestra la visualización para la creación de la variable `POBPCOAC` en la ECH 2019, si bien el INE ya incorpora esta variable en los microdatos, se puede ver como se construye la variable en base a otras variables del formulario.

Si bien se puede ingresar a los atributos de la receta para ver las variables que dependen de la misma, se puede visualizar de una forma más amigable con la función `view_graph` como se puede ver en la Figura 5.2.

5.2.2 Compartiendo Recetas entre Usuarios

En el ejemplo anterior se descargaron recetas ya definidas pero también se pueden compartir recetas entre usuarios, esto puede ser útil para compartir recetas que se han construido en base a la documentación de la encuesta o para compartir recetas que se han construido en base a la experiencia de los usuarios como se ve en el código 5.8.

5.3 Más sobre `metasurvey`

Para la construcción del paquete se ha realizado mucho trabajo en el desarrollo de cada una de las funciones, se ha buscado que el paquete sea lo más eficiente posible y que sea fácil de usar, se han realizado pruebas de rendimiento y se han buscado formas de optimizar el código para que sea lo más rápido posible.

En la Figura 5.3 se puede ver las dependencias de las funciones dentro de `metasurvey`, se puede ver que las funciones están muy bien organizadas y que se pueden utilizar de forma independiente, esto permite que el paquete sea muy flexible y que se puedan utilizar las funciones de forma independiente o en conjunto.

FIGURA 5.2: Flujo de construcción de la variable POBPCOAC en la ECH 2019 y variables de mercado de trabajo que dependen de la misma.

Desarrollar el paquete ha sido un proceso largo y complejo, pero ha sido muy gratificante ver cómo ha evolucionado puede ser utilizado por diferentes usuarios para diferentes tipos de encuestas, se espera que el paquete siga evolucionando y que se sigan incorporando nuevas funcionalidades y mejoras. Si bien el paquete ha sido desarrollado en base a las encuestas de hogares de Uruguay y Argentina, se espera que pueda ser utilizado para otras encuestas de hogares en América Latina y en otros países.

En futuras versiones se espera que el usuario pueda buscar y crear recetas de una interfaz gráfica, un registro de usuarios, monitoreo y analítica de uso, entre otras funcionalidades.

El paquete es muy flexible y la forma de que sea más útil es que los usuarios compartan sus recetas y que se puedan compartir entre usuarios, esto permitirá que el paquete sea más útil y que se puedan construir recetas más complejas y que se puedan compartir entre usuarios.

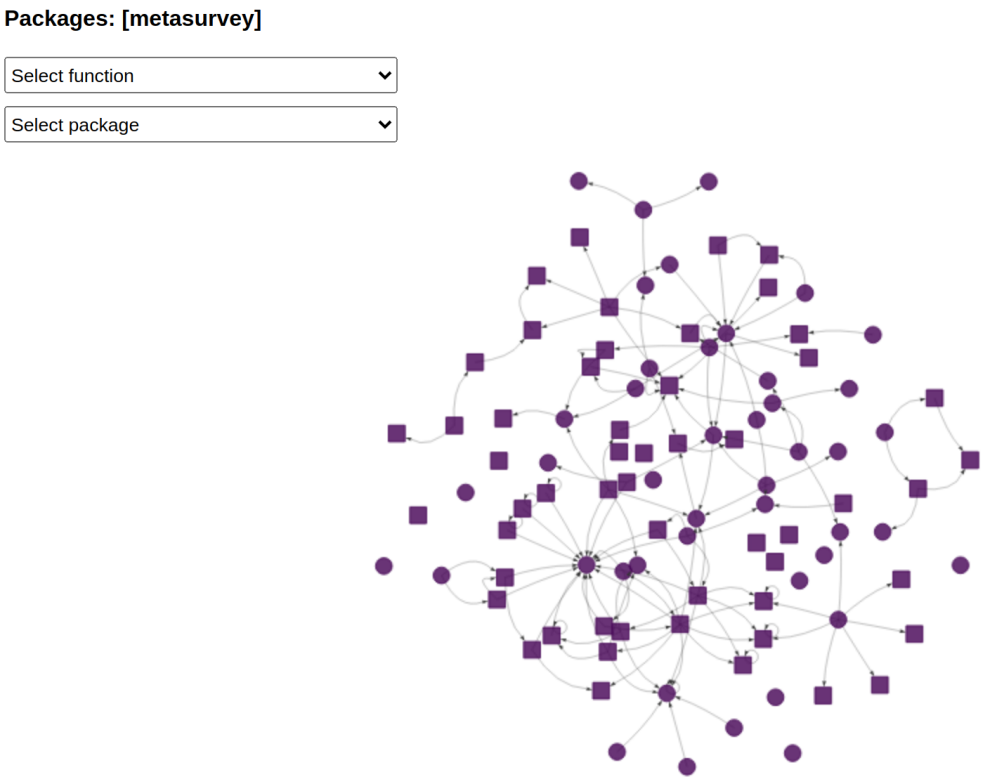


FIGURA 5.3: Funciones dentro de `metasurvey` y sus dependencias. Los nodos representan las funciones y las aristas las dependencias entre ellas.

5.4 Resumen

Los casos de uso presentados demuestran la flexibilidad y potencia de `metasurvey` para el procesamiento de diferentes encuestas de hogares:

- En el caso de la ECH, se mostró cómo manejar eficientemente el nuevo diseño de panel rotativo y los pesos muestrales replicados bootstrap para generar estimaciones precisas de indicadores del mercado laboral.
- Con la EPH, se evidenció la capacidad del paquete para adaptarse a diferentes estructuras de datos y metodologías de encuestas, manteniendo una interfaz consistente.
- La visualización de recetas mediante `view_graph()` facilita la comprensión de las dependencias entre variables y la transparencia en la construcción de indicadores.
- La posibilidad de compartir recetas entre usuarios permite aprovechar el conocimiento colectivo y acelerar el proceso de análisis de encuestas.

Estas características hacen de `metasurvey` una herramienta valiosa para investigadores y analistas que trabajan con encuestas de hogares en América Latina, proporcionando un marco unificado y eficiente para el procesamiento y análisis de datos.

Código 5.1 Carga de la encuesta continua de hogares en 2023, se carga la implantación y el seguimiento de la encuesta, se especifica el tipo de encuesta, el peso de la implantación y el peso del seguimiento, en este caso se utilizan pesos replicados bootstrap para el seguimiento de la encuesta.

```
library(metasurvey)
library(magrittr)

# Actualizar la ruta de los archivos de datos
path_dir <- file.path("/home/runner/work/document_tfg/document_tfg/chapters/example-d

ech_2023 <- load_panel_survey(
    path_implantation = file.path(
        path_dir,
        "ECH_implantacion_2023.csv"
    ),
    path_follow_up = file.path(
        path_dir,
        "seguimiento"
    ),
    svy_type = "ECH_2023",
    svy_weight_implantation = add_weight(
        annual = "W_ANO"
    ),
    svy_weight_follow_up = add_weight(
        monthly = add_replicate(
            "W",
            replicate_path = file.path(
                path_dir,
                c(
                    "Pesos replicados Bootstrap mensuales enero_junio 2023",
                    "Pesos replicados Bootstrap mensuales julio_diciembre 2023"
                ),
                c(
                    "Pesos replicados mensuales enero_junio 2023",
                    "Pesos replicados mensuales Julio_diciembre 2023"
                )
            ),
            replicate_id = c("ID" = "ID"),
            replicate_pattern = "wr[0-9]+",
            replicate_type = "bootstrap"
        )
    )
)
```

Código 5.2 Se puede ver las variables que dependen de la receta de ingreso compatibilizado para la ECH 2022.

```
ingreso_compatibilizado <- get_recipe(
    svy_type = "ech",
    svy_edition = "2022",
    topic = "ingreso"
)
ingreso_compatibilizado$depends_on
```

Código 5.3 Creación de variables para el cálculo de tasas de mercado de trabajo en los microdatos de seguimiento de la ECH 2023.

```
ech_2023 <- ech_2023 %>%
    step_recode(
        "pea",
        POBPCOAC %in% 2:5 ~ 1,
        .default = 0,
        comment = "Población Económicamente Activa",
        .level = "follow_up"
    ) %>%
    step_recode(
        "pet",
        e27 >= 14 ~ 1,
        .default = 0,
        comment = "Población Empleada",
        .level = "follow_up"
    ) %>%
    step_recode(
        "po",
        POBPCOAC == 2 ~ 1,
        .default = 0,
        comment = "Población Ocupada",
        .level = "follow_up"
    ) %>%
    step_recode(
        "pd",
        POBPCOAC %in% 3:5 ~ 1,
        .default = 0,
        comment = "Población Desocupada",
        .level = "follow_up"
    )

ech_2023_bake <- bake_steps(ech_2023)
```

Código 5.4 Estimación de tasas de mercado de trabajo a nivel trimestral a partir de las estimaciones mensuales.

```
mercado_trabajo_mensual <-
    workflow(
        survey = extract_surveys(
            ech_2023_bake,
            quarterly = 1:4
        ),
        survey::svyratio(
            ~pea,
            denominator = ~pet
        ),
        survey::svyratio(
            ~po,
            denominator = ~pet
        ),
        survey::svyratio(
            ~pd,
            denominator = ~pea
        ),
        estimation_type = "quarterly:monthly",
        rho = 0.5,
        R = 5 / 6
    )
```

Código 5.5 Estimación de tasas de mercado de trabajo a nivel anual a partir de las estimaciones mensuales.

```
mercado_trabajo_anual <- workflow(
    survey = extract_surveys(
        ech_2023_bake,
        annual = 2023
    ),
    survey::svyratio(
        ~pea,
        denominator = ~pet
    ),
    survey::svyratio(
        ~po,
        denominator = ~pet
    ),
    survey::svyratio(
        ~pd,
        denominator = ~pea
    ),
    estimation_type = "annual:monthly"
)
```

Código 5.6 Carga de la encuesta permanente de hogares en el tercer trimestre de 2022, se crean las variables necesarias para el cálculo de tasas de mercado de trabajo.

```
eph2022_3 <- metasurvey::load_survey(
    path = metasurvey::load_survey_example(
        "eph",
        "eph2022_3"
    ),
    svy_type = "eph",
    svy_edition = "eph_202302",
    svy_weight = add_weight(
        monthly = "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
    )
```

Código 5.7 Carga de la encuesta y se obtienen las recetas disponibles referidas al tópico de mercado de trabajo.

```
ech_2019 <- load_survey(
    path = "https://metasurvey-example-data.s3.us-east-2.amazonaws.com/ech/ech_2019.csv",
    svy_type = "ech",
    svy_edition = "2019",
    svy_weight = add_weight(
        annual = "pesoano"
    ),
    recipes = get_recipe(
        svy_type = "ECH",
        svy_edition = 2019,
        topic = "Mercado de trabajo"
    )
)
ech_2019 <- bake_recipes(ech_2019)
```

Código 5.8 Compartir receta de clasificación de la población por condición de actividad en la ECH 2019.

```
receta <- steps_to_recipe(
    name = "Población por condición de actividad",
    user = "INE-Uruguay",
    svy = ech_2019,
    description = "Clasificación de la población por condición de actividad",
    steps = ech_2019$steps,
    topic = "Mercado de trabajo"
)

publish_recipe(receta)
#> $insertedId
#> [1] "67be01564f09ed65a224295b"
```

Capítulo 6

Pasos a futuro

El desarrollo de `metasurvey` marca un hito significativo en la creación de herramientas reproducibles y transparentes para el análisis de encuestas por muestreo. Sin embargo, más allá de los avances alcanzados, este trabajo abre múltiples caminos de exploración, mejora y expansión. En este capítulo, reflexionamos sobre los logros conseguidos y planteamos una hoja de ruta ambiciosa pero alcanzable para el futuro.

6.1 Logros alcanzados y su relevancia

Durante este proceso, `metasurvey` ha consolidado su posición como una herramienta versátil y poderosa para investigadores y analistas de datos. Su diseño modular y su enfoque en la reproducibilidad han sido claves para facilitar el análisis de datos complejos en un entorno accesible. Este capítulo busca contextualizar estos logros dentro de una visión más amplia de las necesidades actuales y emergentes en el campo de las encuestas por muestreo.

6.2 Optimización del rendimiento y escalabilidad

6.2.1 Hacia un procesamiento paralelo eficiente

Uno de los desafíos principales en el manejo de encuestas es la escalabilidad. En particular, el procesamiento de grandes conjuntos de datos o de análisis longitudinales representa una carga significativa para los sistemas. La implementación de procesamiento paralelo nativo se erige como una prioridad. Al habilitar cálculos distribuidos para operaciones intensivas como la generación de réplicas bootstrap, no solo se reducirán los tiempos de ejecución, sino que se abrirá la puerta a análisis más complejos y en tiempo real.

Además, exploraremos estrategias de reducción de memoria mediante la implementación de estructuras de datos más ligeras y algoritmos optimizados, adaptados a los escenarios más comunes de uso.

6.3 Expansión de funcionalidades estadísticas

6.3.1 Integración de nuevos métodos analíticos

La robustez de una herramienta radica en su capacidad de adaptarse a nuevos paradigmas y métodos. En este sentido, la integración de metodologías avanzadas para el cálculo de varianzas y estimaciones en diseños complejos es un área de desarrollo clave.

Por ejemplo, los métodos de (Deville and Tillé, 2005) proporcionan un marco teórico sólido que puede fortalecer las capacidades de `metasurvey`.

Adicionalmente, el soporte para modelos lineales representará un paso adelante en el tratamiento de diseños muestrales complejos, permitiendo a los usuarios responder preguntas de investigación más sofisticadas.

6.4 Fomentar una comunidad activa de usuarios

La sostenibilidad de `metasurvey` como herramienta de referencia depende en gran medida de la creación de una comunidad activa y comprometida. Para ello, se prevé:

- **Plataformas de colaboración**: Establecimiento de un repositorio público interactivo donde los usuarios puedan compartir scripts, soluciones y preguntas frecuentes.
- **Eventos de capacitación**: Organización de talleres y webinars orientados tanto a principiantes como a usuarios avanzados, cubriendo casos prácticos y metodologías avanzadas.
- **Reconocimiento a contribuciones**: Introducción de un sistema de menciones en las publicaciones científicas del proyecto para quienes contribuyan de manera significativa al desarrollo de `metasurvey`.

6.5 Conclusión

`metasurvey` no solo es una herramienta; es un paso hacia la democratización del análisis de encuestas. A medida que avanzamos hacia un futuro más interconectado, el compromiso con la innovación, la reproducibilidad y la colaboración será el motor que impulse su evolución continua. Este proyecto, aunque ambicioso, refleja la pasión y el compromiso de una comunidad dedicada a transformar la investigación por muestreo.

En resumen, los próximos pasos delineados aquí representan no solo una oportunidad de mejora técnica, sino también una invitación a co-crear un futuro donde el análisis de encuestas sea accesible, inclusivo y poderoso.

Biblografía

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," 2024, 2024. https://airflow.apache.org/docs/latest/.

Barrett, Tyson, Matt Dowle, Arun Srinivasan, Jan Gorecki, Michael Chirico, Toby Hocking, and Benjamin Schwendinger, 2024. *Data.table: Extension of 'Data.frame'*. https://CRAN.R-project.org/package=data.table.

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.

Codecov: Code Coverage Insights, 2024. https://about.codecov.io.

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

Detomasi, Gabriela Mathieu & Richard, 2020. "Ech: Caja de Herramientas Para Procesar La Encuesta Continua de Hogares," 2020. 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.

Driessen, Vincent, 2010. "A Successful Git Branching Model." https://nvie.com/posts/a-successful-git-branching-model/.

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.

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

Freedman Ellis, Greg, and Ben Schneider, 2024. *Srvyr: 'Dplyr'-Like Syntax for Summary Statistics of Survey Data*. https://CRAN.R-project.org/package=srvyr.

Gentle, James E., 2009. *Computational Statistics*. Statistics and Computing. New York, NY: Springer. https://link.springer.com/10.1007/978-0-387-98144-4.

Git, 2025. *Git: Fast Version Control System*. https://git-scm.com/.

GitHub Actions: Automate Your Workflow, 2024. GitHub. https://github.com/features/actions.

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.

Instituto de Economía, Universidad de la República. 2020. "Encuesta Continua de Hogares Compatibilizada 1981-2018." http://doi.org/10.47426/ECH.INE.

Instituto Nacional de Estadística, 2021. "Metodología de La Encuesta Continua de Hogares Instituto Nacional de Estadística," 2021. https://www.ine.gub.uy.

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, 2004. "Analysis of Complex Survey Samples." *Journal of Statistical Software* 9 (, 2004): 1–19. https://doi.org/10.18637/jss.v009.i08.

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

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.

Open Source Initiative, 1988. "The MIT License (MIT)." https://opensource.org/licenses/MIT.

Prabhu, Anirudh, and Peter Fox, 2020. "Reproducible Workflow," 2020. 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. *Foreign: Read Data Stored by 'Minitab', 's', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...* https://CRAN.R-project.org/package=foreign.

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

"R Packages (2e)." n.d. https://r-pkgs.org/.

rOpenSci, Brooke Anderson, Scott Chamberlain, Laura DeCicco, Julia Gustavsen, Anna Krystalli, Mauro Lepore, et al., 2024. "rOpenSci Packages: Development, Maintenance, and Peer Review." Zenodo. https://doi.org/10.5281/zenodo.10797633.

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, 1992. *Model Assisted Survey Sampling*. Springer Science & Business Media.

Schauberger, Philipp, and Alexander Walker. 2024. *Openxlsx: Read, Write and Edit Xlsx Files*. https://CRAN.R-project.org/package=openxlsx.

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

Sottile, Anthony, and Contributors, 2024. *Pre-Commit: A Framework for Managing and Maintaining Multi-Language Pre-Commit Hooks*. https://pre-commit.com.

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.

Valliant, Richard, Alan Dorfman, and Richard Royall, 2000. "Finite Population Sampling and Inference," 2000.

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

———, 2023. *Httr: Tools for Working with URLs and HTTP*. https://CRAN.R-project.org/package=httr.

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

———, 2011. "Testthat: Get Started with Testing." *The R Journal* 3 (, 2011): 5–10. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Wickham, Hadley, Jennifer Bryan, Malcolm Barrett, and Andy Teucher, 2024. *Usethis: Automate Package and Project Setup*. https://CRAN.R-project.org/package=usethis.

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

Wickham, Hadley, Jay Hesselberth, Maëlle Salmon, Olivier Roy, and Salim Brüggemann, 2024. *Pkgdown: Make Static HTML Documentation for a Package*. https://CRAN.R-project.org/package=pkgdown.

Wickham, Hadley, Jim Hester, Winston Chang, and Jennifer Bryan, 2022. *Devtools: Tools to Make Developing r Packages Easier*. https://CRAN.R-project.org/package=devtools.

Wickham, Hadley, Evan Miller, and Danny Smith, 2023. *Haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files*. https://CRAN.R-project.org/package=haven.

Apéndice A

Apendice

Código A.1 Instalación del paquete

```
branch <- "develop"

is_available <- "metasurvey" %in% rownames(
 available.packages(
    repos = "https://cloud.r-project.org/"
    )
  )

  if (is_available) {
    install.packages("metasurvey")
  } else {
    remotes::install_github(
      "metasurveyr/metasurvey",
      ref = branch,
      force = TRUE
    )
    message("Se instaló la versión de desarrollo de metasurvey")
  }
```