Universidad de la República

Facultad de Ciencias Económicas y de Administración

Trabajo final de grado

---

metasurvey

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

Estudiante:
Mauro Loprete

Tutora:
Dra. Natalia da Silva

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

Resumen

metasurvey

por Mauro Loprete

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

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

Tutor: Natalia da Silva

Licenciatura en Estadística

Calificación:

Fecha: 17/12/2024

Tribunal:

Ignacio Álvarez-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 t ambién es de todos ellos.

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

Índice

Resumen **iii**

Agradecimientos **v**

1

1 Introducción **3**
1.1 Motivación . . . 3
1.2 Contexto . . . 4
1.3 Antecedentes e implementaciones similares . . . 5
1.4 Propuesta . . . 6
1.5 Desarrollo del paquete `metasurvey` . . . 7
1.6 Esquema del documento . . . 8

2 Marco conceptual **11**
2.1 Inferencia en muestreo de poblaciones finitas . . . 11
2.1.1 Diseño muestral . . . 11
2.1.2 Probabilidades de inclusión y estimador de Horvitz-Thompson 13
2.1.3 Ponderación basada en el diseño y estimadores más comunes . 13
2.1.4 Medidas de incertidumbre y errores estándar . . . 14
2.2 Desarrollo de paquetes en R . . . 16
2.2.1 ¿Por qué desarrollar un paquete en R? . . . 17
2.2.2 Elementos básicos de un paquete en R . . . 17
2.3 Paradigmas de programación en R . . . 18
2.3.1 Programación funcional . . . 18
2.3.2 Programación orientada a objetos . . . 19
2.3.3 Meta-programación . . . 19

3 Marco teórico **21**
3.1 Investigación reproducible . . . 21
3.1.1 Conceptos clave . . . 23
3.1.2 Workflow reproducible . . . 23
3.2 Investigación reproducible en R . . . 24
3.2.1 Herramientas para el procesamiento de encuestas . . . 24
3.3 Diseño de encuestas y estimación de varianza . . . 25
Resumen de las implementaciones . . . 25
3.4 Desarrollos de paquetes en R . . . 27
Implementación de tests automatizados . . . 27
Documentación . . . 28
Pruebas en diferentes sistemas operativos y versiones de R junto a GitHub Actions . . . 29

4 Desarrollo y metodología **33**
4.1 Estimación de de los errores estándar . . . 33
4.1.1 Métodos de remuestreo . . . 34
4.2 Desarrollo e Implementación . . . 36
4.2.1 Dependencias . . . 36
4.2.2 Ejemplos de la implementación . . . 37
4.2.3 Meta-programación . . . 40

5 Casos de uso **43**
5.1 Encuesta Continua de Hogares . . . 43
5.2 EPH . . . 46

6 Pasos a futuro **49**

7 Bibliografía **51**

Appendices **55**

A Frequently Asked Questions **55**
A.1 How do I change the colors of links? . . . 55

Lista de Figuras

Lista de Tablas

3.1 Comparación de paquetes para análisis de encuestas 26

Lista de códigos

1.1 Instalación de metasurvey 9
3.1 Ejemplo de un test automatizado para verificar el correcto funcionamiento de la función extract_time_pattern. 28
3.2 Extracto de la documentación de la función `load_survey` con las etiquetas necesarias para la generación de la documentación. 29
3.3 Archivo de configuración de GitHub Actions para la ejecución de tests en diferentes sistemas operativos y versiones de R. 31
4.1 Lectura de la encuesta ECH 2022, fijación del ponderador y obtención de recetas. 38
4.2 Definición de la clase `Step` con sus atributos y método `initialize` para inicializar la clase. 39
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. 40
4.4 Desactivar el uso de referencias y utilizar copias de los objetos. 40
4.5 Ejemplo de función para encontrar dependencias de variables en una expresión de un step. Existen otros ejemplos de meta-programación en el paquete aunque son más complejos y no se muestran en este documento. Puede ver ejemplos en la función `compute` y `recode` del paquete `metasurvey` funciones internas que se encargan de aplicar los pasos y recetas a la encuesta. Step.R 41
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. 44
5.2 Obtener los microdatos de la encuesta continua de hogares para el mes de enero de 2023 y ver el diseño de la encuesta. 45

Capítulo 1

Introducción

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

En este trabajo se presenta el paquete `metasurvey`, una herramienta para el procesamiento y obtención de indicadores a partir de encuestas por muestreo. El paquete permite al usuario tener un control total sobre el proceso de transformación de los microdatos a indicadores, permitiendo que el usuario pueda validar y entender su proceso de construcción, además de brindar una forma para la construcción de variables sintéticas, como recodificar variables creando grupos en base a criterios complejos, tratamiento de variables continuas como el ingreso salarial en base a una metodología rigurosa pero siendo fácil de referenciar en la implementación. Es crucial que este proceso sea transparente y entendible para el usuario.

1.1 Motivación

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

En general, el proceso de transformación de los microdatos a indicadores requiere de un conocimiento profundo de la encuesta y en su mayoría no es de conocimiento general. Si bien existen diferentes esfuerzos para facilitar el procesamiento de encuestas, en general estos paquetes tienen limitaciones en cuanto a la flexibilidad y transparencia del proceso de transformación de los microdatos a indicadores de interés. Estas implementaciones son muy sensibles a la estructura y las variables que componen la encuesta, un cambio en la estructura de la encuesta suele implicar una actualización del paquete utilizado para obtener los indicadores en la nueva edición de la encuesta, lo que resulta poco flexible ante cambios en la estructura, que pueden ser frecuentes en la práctica.

En este sentido, es importante construir una herramienta que permita al usuarios tener un control total sobre el proceso de transformación de los microdatos a indicadores, ya que esto permite que el usuario pueda validar y entender el proceso de construcción

de indicadores, de una forma totalmente desacoplada dentro de las implementaciones de cada función.

Dentro de este documento se detalla el desarrollo de una herramienta que permita al usuario con simples pasos poder construir indicadores junto a una forma de obtener metodologías brindadas por la comunidad científica pudiendo reproducir los resultados o incluir de forma sencilla la metodología en su propio código. Esta herramienta debe permitir al usuario tener un control total sobre el proceso de transformación de los microdatos a indicadores, permitiendo que el usuario pueda validar y entender el proceso de construcción de indicadores, todo esto estará disponible en forma de paquete en R (R Core Team 2023b).

1.2 Contexto

A lo que refire a la teoría de la inferencia de poblaciones en muestreo de poblaciones finitas, es importante tener en cuenta la incertidumbre y errores asociados a las estimaciones producidas, en general esto no es considerado por los usuarios no expertos en metodología de muestreo. Esto puede llevar a conclusiones erróneas ya que en algunos casos el estimador asociado a la estimación cuenta con una alta variabilidad o fue calculado sin tener en cuenta el diseño muestral correcto.

Antes de continuar, es importante distinguir dentro de la inferencia estadística el enfoque **model-based inference** y **desing-based inference** (Lumley 2011). En el primer enfoque, se asume que la población de interés se puede modelar mediante un modelo probabilístico y se pueden obtener estimaciones de los parámetros del modelo mediante técnicas de inferencia estadística. En el segundo enfoque, se asume que la población de interés es finita y se puede obtener estimaciones de los parámetros de la población mediante técnicas de muestreo.

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

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

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

Una vez considerado el proceso de inferencia también es crucial tener en cuenta el proceso de transformación de los microdatos a indicadores ya que es importante para interpretar los indicadores de manera correcta y realizar comparaciones a lo largo del

tiempo para formalizar la metodología de su construcción. Muchas veces, diferentes usuarios hacen el mismo esfuerzo de construcción de indicadores de manera independiente y sin compartir el código fuente o la metodología de construcción de indicadores, ya que cada uno utiliza su propio estilo de programación o hasta diferentes paquetes estadísticos, en su mayoría propietarios como SPSS, SAS o STATA, donde si bien el usuario puede compartir la sintaxis para su construcción, esta está ligada al software y depende de que el usuario tenga el software instalado con una licencia activa y pueda correr el código.

En los últimos años, el uso de R (R Core Team 2023b) ha crecido exponencialmente en la comunidad científica, en especial en el área de la estadística y la ciencia de datos. R es un lenguaje de programación de código abierto ampliamente utilizado en la comunidad científica para el análisis de datos, estadística y aprendizaje automático, y en general se utiliza el concepto *paquete* para referirse a una colección de funciones, métodos y clases que extienden las funcionalidades de R propuestas por la misma comunidad de usuarios. En este sentido, `metasurvey` busca ser una herramienta relevante para el trabajo con encuestas por muestreo en general ya sea en las ciencias sociales o el uso genérico para otras disciplinas, buscando solucionar las limitaciones anteriormente mencionadas.

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

1.3 Antecedentes e implementaciones similares

Actualmente existen varios esfuerzos para facilitar el procesamiento de encuestas, entre ellos existen principalmente dos tipos de paquetes, aquellos que implementan la metodología de inferencia en muestreo de poblaciones finitas como puede ser el paquete `survey` (Lumley 2024a), `gustave` (Chevalier 2023), `vardpoor` (Breidaks, Liberts, and Ivanova 2020), `svrep` (Schneider 2023), `weights` y aquellos que permiten acceder y manipular encuestas específicas como `ech` (Detomasi 2020), `eph` (Kozlowski et al. 2020), `tidycensus` (Walker and Herman 2024), `casen` (Vargas 2024) entre otros. Sin embargo, estos últimos tienen limitaciones en cuanto a la flexibilidad y transparencia del proceso de transformación de los microdatos a indicadores de interés, como puede ser el indice de pobreza, tasas del mercado laboral, ingreso salarial, etc. En general, sus implementaciones son muy sensibles a la estructura y las variables que componen la encuesta, un cambio en la estructura de la encuesta suele implicar una actualización del paquete utilizado para obtener los indicadores en la nueva edición de la encuesta, lo que resulta poco flexible ante cambios en la estructura, que pueden ser frecuentes en la práctica. Además en las implementaciones actuales, el usuario cuenta con una función de alto nivel que actúa como una caja negra, donde no se permite modificar el código para adaptarlo a sus necesidades o entender cada paso que se realiza para obtener el indicador sin tener que leer el código fuente o la documentación adjunta.

Este tipo de problemas puede verse en `ech` (Detomasi 2020), donde existen funciones para crear variables de mercado laboral, educación o ingresos, pero estas funciones dependen de la existencia de ciertas variables en la encuesta, cuya estructura puede cambiar de una versión a otra de la encuesta. Sin revisar el cuerpo de la función, no se conoce el proceso de construcción de variables. Algo similar ocurre con `eph` (Kozlowski et al. 2020), donde se tienen funciones de alto nivel que no permiten modificar el código para adaptarlo a sus necesidades o entender cada paso que se realiza para obtener el indicador sin inspeccionar a fondo cómo se construyen las funciones del paquete. Esta inspección del código fuente, como consultar el repositorio de GitHub del paquete o revisar la definición de la función, puede ser una tarea tediosa y no garantiza que el usuario pueda entender el proceso de construcción de variables. Esto se debe a que el código puede ser muy extenso o que el usuario no tenga el conocimiento suficiente para entender el código o se empleen ciertos frameworks que el usuario no conozca, como el uso de las librerías `dplyr` (Wickham et al. 2023) o `tidyr` (Wickham, Vaughan, and Girlich 2024), muy populares en R para el manejo de datos. También puede ser difícil aislar el proceso de manipulación de la encuesta de la implementación específica de la función para manejar la forma de presentación, estructura del objeto a devolver, etc. Un claro ejemplo de esto puede verse en `tidycensus` (Walker and Herman 2024), donde existe una función para obtener datos sobre la migración de la comunidad estadounidense, y en la misma implementación se encuentran pasos para mejorar la estructura del conjunto de datos a devolver. En este sentido, el usuario no puede aislar el proceso de re-codificación/construcción de variables sobre variables originales y la obtención de datos geográficos y presentación.

1.4 Propuesta

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

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

anteriormente y se presentará el paquete `metasurvey` y su implementación de *recetas* para la construcción de indicadores mediante la meta-programación.

Al trabajar con encuestas por muestreo, es importante tener en cuenta la forma en la que se obtuvieron los datos y su proceso generador para poder realizar inferencias sobre la población de interés. En general, obtener estimaciones puntuales de estadísticos de totales, promedios o proporciones es relativamente sencillo, pero puede ser que se reporte una estimación donde no exista un tamaño de muestra suficiente para obtener una estimación confiable y/o que la variabilidad de la estimación sea alta y no sea recomendable su uso. En este sentido, es importante que el usuario no experto tenga de forma nativa una forma de obtener estimaciones puntuales y sus errores asociados de manera sencilla. Es común utilizar estimaciones puntuales sin tener una medida de incertidumbre o aún peor incluir una estimación del error estándar sin tener en cuenta el diseño muestral correcto, lo que puede llevar a conclusiones erróneas sobre la variabilidad de la estimación. `metasurvey` permite que el usuario pueda obtener estimaciones puntuales y sus errores asociados de forma nativa y con estos resultados hacer recomendaciones sobre la utilidad y confianza de la estimación mediante coeficientes de variación, intervalos de confianza, tamaño de muestra efectivo, entre otros sin tener que ser un experto en metodología de estimación de varianzas y remuestreo. En capítulos posteriores se abordarán ejemplos con los paquetes mencionados anteriormente y se presentará el paquete `metasurvey` y su implementación de estimaciones puntuales y sus errores asociados.

1.5 Desarrollo del paquete `metasurvey`

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

En este sentido, `metasurvey` pretende ser una herramienta relevante para el trabajo con encuestas por muestreo en general ya sea en las ciencias sociales o el uso genérico para otras disciplinas, buscando solucionar las limitaciones anteriormente mencionadas. Todo el proceso de transformación de los microdatos a indicadores se realiza a través de una serie de funciones que permiten al usuario tener un control total y transparente sobre el proceso de transformación de los microdatos a indicadores. Además, `metasurvey` permite que el usuario pueda realizar el proceso de transformación de los microdatos a indicadores de manera reproducible y transparente. El usuario puede compartir el código de una forma entendible, casi como un "recetario de cocina". El procedimiento aplicado a los datos utilizados para obtener los indicadores se realiza mediante lo que denominamos steps y recipes, conformando así una especie de camino transparente para la construcción de indicadores. Esto permite compartir en forma visual un DAG (Directed Acyclic Graph) que permite visualizar el proceso de construcción de indicadores sin tener que abrir un script de R. En complemento al proceso de creación de variables, `metasurvey` permite que el

usuario pueda obtener estimaciones puntuales y sus errores asociados de manera sencilla y brindar recomendaciones sobre la utilidad de la estimación en el caso de que se cuente con una variabilidad alta en la estimación, en base a recomendaciones a su coeficiente de variación o métricas similares.

El enfoque que permite la flexibilidad a la hora de construir los indicadores es la meta-programación. La meta-programación es un paradigma de programación que permite que un programa pueda modificar su estructura interna en tiempo de ejecución. En R, la meta-programación se realiza a través de las funciones `eval`, `parse`, `substitute`, `do.call` y `quote`, que permiten evaluar y parsear código de manera dinámica. En este sentido, `metasurvey` utiliza la meta-programación para permitir que el usuario pueda modificar el código que se utiliza para transformar los microdatos a indicadores, teniendo funciones de alto nivel similares a las que se utilizan en el paquete recipes de la librería `tidymodels` (Kuhn, Wickham, and Hvitfeldt 2024).

1.6 Esquema del documento

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

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

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

Para poder continuar con el documento, se recomienda instalar metasurvey en su versión de desarrollo, para ello se puede ejecutar el siguiente Código 1.1

Código 1.1 Instalación de metasurvey

```
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 instalo la versión de desarrollo de metasurvey")
}
```

Capítulo 2

Marco conceptual

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

El objetivo principal de este capítulo es presentar los conceptos básicos que se utilizarán a lo largo de este trabajo, en específico en el Capítulo 3. En primer lugar se presentara un marco básico de inferencia en muestreo de poblaciones finitas para luego presentar diferentes métodos de estimación de parámetros poblacionales y sus respectivos errores estándar. Se hará una primera introducción a diseños sencillos y se propondrán diferentes estimadores y se hará mención a su diferencia con los diseños complejos, situación común en Encuestas Socioeconómicas. Luego, se presentarán los conceptos básicos de la programación funcional y orientada a objetos en R para luego enfocarnos en la meta-programación. Finalmente, se presentará un breve resumen de cómo crear un paquete en R, los componentes mínimos para su publicación en **CRAN** repositorio donde se encuentran disponibles versiones estables de diferentes paquetes de R, y las herramientas que se pueden utilizar para su desarrollo.

2.1 Inferencia en muestreo de poblaciones finitas

Como fue mencionado anteriormente las encuestas por muestreo son la principal fuente de información para la construcción de indicadores socio-demográficos y económicos, en este sentido, es importante tener en cuenta un marco teórico para realizar estas inferencias. Es sumamente sencillo obtener estimaciones puntuales de un determinado estadístico aunque es importante considerar la variabilidad de los estimadores, tanto para poder realizar un proceso de inferencia completo así como también para poder cuantificar la confiabilidad de la estimación.

A continuación, se definen los conceptos básicos de inferencia en muestreo de poblaciones finitas como son el diseño muestral, probabilidades de inclusión basadas en el diseño, estimadores de Horvitz-Thompson **HT**, ponderación, medidas de incertidumbre y errores estándar basados en (Särndal, Swensson, and Wretman 2003).

2.1.1 Diseño muestral

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

es necesario abordar el problema de otra forma y asumir ciertas hipótesis para poder construir probabilidades de inclusión tanto de primer orden como segundo orden el cual sera abordado en la sección 2.1.2

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

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

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

- **Diseño Aleatorio Simple (SI)**

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

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

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

- **Diseño Bernoulli (BE)**

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

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

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

- **Diseño Estratificado (ST)**

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

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

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

2.1.2 Probabilidades de inclusión y estimador de Horvitz-Thompson

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

- **Probabilidad de inclusión de primer orden**

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

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

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

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

Este estimador es propuesto por Horvitz y Thompson en 1952 y es un estimador insesgado en el diseño, en el sentido de que $E(\hat{t}_y) = t$ y es eficiente en el sentido de que $Var(\hat{t}_y)$ es el menor posible entre los estimadores insesgados. Este estimador es muy utilizado en la práctica y es la base para la construcción de otros estadísticos,como medias, proporciones, varianzas, entre otros. Para mas detalles sobre las propiedades de Horvitz-Thompson (**HT**) se puede consultar en (Särndal, Swensson, and Wretman 2003) y (Horvitz and Thompson 1952).

2.1.3 Ponderación basada en el diseño y estimadores más comunes

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

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

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

Estimador de un promedio

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

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

Estimador de una proporción

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

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

Estimador de una razón

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

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

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

Inferencia sobre el tamaño de la población

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

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

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

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

2.1.4 Medidas de incertidumbre y errores estándar

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

2.1.4.1 Momentos muéstrales y estimadores de varianza

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

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

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

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

Después de simplificar, obtenemos:

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

Donde definimos las siguientes identidades para simplificar cálculos:

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

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

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

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

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

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

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

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

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

Es importante considerar que en esta sección se presenta un caso ideal donde la muestra es obtenida de un listado **perfecto** de la población objetivo denominado **marco de muestreo**. En la práctica, el marco de muestreo es imperfecto y se debe considerar la no respuesta, la cobertura y la falta de actualización del marco de muestreo. En general para la publicación de microdatos se publican ciertos ponderadores que no son precisamente los ponderadores originales definidos en la sección anterior sino que son sometidos a un proceso de **calibración** donde se intenta ajustar a ciertas variables de control y mejorar problemas causados por la no respuesta. Al realizar el proceso de calibración los ponderadores calibrados son lo mas cercano posible a los ponderadores originales, de forma que si los ponderadores originales son insesgados, los ponderadores calibrados serán próximos a ser insesgados.

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

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

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

2.2 Desarrollo de paquetes en R

R es un lenguaje de código abierto y además cuenta con una gran comunidad de usuarios, en diferentes áreas de investigación, esto ha permitido que se desarrollen una gran cantidad de paquetes que permiten realizar diferentes tareas de análisis de datos, visualización, bioinformática, aprendizaje automático y ramas afines a la estadística. Dentro de la comunidad existen diferentes organizaciones que se encargan de mantener la calidad de los paquetes y de asegurar que los paquetes cumplan con ciertos estándares de calidad, una de estas organizaciones es el **Comprehensive R Archive Network** (**CRAN**), que es un repositorio de paquetes de R que contiene versiones estables de los paquetes de R, bioconductor, que es un repositorio de paquetes de R que contiene paquetes para el análisis de datos biológicos, y rOpenSci que Para casi cualquier disciplina científica o en la industria se puede encontrar una comunidad de usuarios que desarrollan paquetes en R, en este sentido, el desarrollo de paquetes en R es una tarea que se ha vuelto muy común entre los usuarios de R y

es muy sencillo de realizar. A continuación, se presentan los conceptos básicos para el desarrollo de paquetes en R.

2.2.1 ¿Por qué desarrollar un paquete en R?

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

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

2.2.2 Elementos básicos de un paquete en R

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

- **Directorio**: Un paquete en R debe estar contenido en un directorio que contenga al menos los siguientes archivos y directorios:
  - `R/`: Directorio que contiene los archivos con las funciones que se desean incluir en el paquete.
  - `man/`: Directorio que contiene los archivos con la documentación de las funciones que se encuentran en el directorio `R/`. En general se utiliza *Roxygen2* (Wickham, Danenberg, et al. 2024) para generar la documentación de las funciones.
  - `DESCRIPTION`: Archivo que contiene la descripción del paquete, incluyendo el nombre, versión, descripción, autor, entre otros.
  - `NAMESPACE`: Archivo que contiene la información sobre las funciones que se exportan y las dependencias del paquete.
  - `LICENSE`: Archivo que contiene la licencia bajo la cual se distribuye el paquete.
  - `README.md`: Archivo que contiene información general sobre el paquete.
- **Documentación**: La documentación de las funciones es un componente esencial de un paquete en R, ya que permite que los usuarios puedan entender el funcionamiento de las funciones que se encuentran en el paquete. La documentación de las funciones se realiza utilizando el sistema de documentación de R, que se basa en el uso de comentarios en el código fuente de las funciones.
- **Pruebas**: Es importante que el paquete tenga pruebas que permitan verificar que las funciones se comportan de la manera esperada. Las pruebas se realizan utilizando el paquete *testthat* (Wickham 2011a) que permite realizar pruebas unitarias.

- **Control de versiones**: Es importante que el paquete tenga un sistema de control de versiones que permita llevar un registro de los cambios que se realizan en el paquete. El sistema de control de versiones más utilizado en la comunidad de R es `git`.
- **Licencia**: Es importante que el paquete tenga una licencia que permita a los usuarios utilizar, modificar y distribuir el paquete. La licencia más utilizada en la comunidad de R es la licencia MIT.

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

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

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

2.3 Paradigmas de programación en R

R es un lenguaje de programación que permite realizar programación funcional y orientada a objetos (Chambers 2014), lo que permite que los usuarios puedan utilizar diferentes paradigmas de programación para resolver problemas. A continuación, se presentan los conceptos básicos de la programación funcional y orientada a objetos en R.

2.3.1 Programación funcional

La programación funcional es un paradigma de programación que se basa en el uso de funciones para resolver problemas. En R, las funciones son objetos de primera clase, lo que significa que se pueden utilizar como argumentos de otras funciones, se pueden asignar a variables, entre otros (Wickham 2019, 204–81). A continuación, se presentan los conceptos básicos de la programación funcional en R.

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

ejemplo de una función pura en R es la función `sqrt` que toma como argumento un número y retorna la raíz cuadrada de ese número.

Este paradigma de programación es muy útil para realizar análisis de datos, ya que permite que los usuarios puedan utilizar funciones para realizar operaciones sobre los datos de manera sencilla y eficiente, dentro de metasurvey no existe una presencia fuerte de programación funcional, sin embargo, se utilizan algunas funciones de orden superior para realizar operaciones sobre los datos.

2.3.2 Programación orientada a objetos

La programación orientada a objetos es un paradigma de programación que se basa en el uso de objetos para resolver problemas. En R, los objetos son instancias de clases que tienen atributos y métodos (Wickham 2019, 285–370; Mailund 2017). A continuación, se presentan los conceptos básicos de la programación orientada a objetos en R.

- **Clases y objetos**: En R, las clases son plantillas que definen la estructura y el comportamiento de los objetos y los objetos son instancias de clases. En R, las clases 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 una creación de una clase especifica llamada `Survey` que permite además de almacenar los datos de la encuesta añadir atributos y métodos que permiten realizar operaciones sobre los datos de manera sencilla y eficiente.

De forma similar se modelan las clases `Step`, `Recipe` y `Survey`, 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.

2.3.3 Meta-programación

La meta-programación es un paradigma de programación que se basa en el uso de código para manipular código (Wickham 2019, 373–500; Thomas Mailund 2017) . En R, la meta-programación se realiza utilizando el sistema de meta-programación de R que se basa en el uso de expresiones, llamadas y funciones. A continuación, se presentan los conceptos básicos de la meta-programación en R.

- **Expresiones**: En R, las expresiones son objetos que representan código y se crean utilizando la función `quote`. Un ejemplo de una expresión en R es la expresión `quote(x + y)` que representa el código `x + y`.

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

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

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

Capítulo 3

Marco teórico

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

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

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

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

3.1 Investigación reproducible

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

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

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

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

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

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

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

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

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

3.1.1 Conceptos clave

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

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

3.1.2 Workflow reproducible

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

En el ámbito del aprendizaje automático existe un gran esfuerzo para poder desgranar y documentar los modelos conocido como **Model Cards** (Mitchell et al. 2019)

donde se hace un detalle de los algoritmos utilizados, las métricas de evaluación, los datos utilizados y su procesamiento, siendo esto el análogo a los `Steps` y `Recipes` de `metasurvey`. Este concepto se ha extendido siendo un estándar en la industria y siendo adoptado por diferentes organizaciones como **Google** y **Hugging Face**.

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

3.2 Investigación reproducible en R

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

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

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

3.2.1 Herramientas para el procesamiento de encuestas

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

Para el procesamiento de encuestas por muestreo, existe una serie de paquetes que permiten implementar la metodología de encuestas por muestreo como puede ser el caso de `survey` (Lumley 2024a) que permite el análisis de encuestas complejas, `srvyr` (Ellis and Schneider 2023) aunque estos son utilizados en el proceso final o de inferencia y no en el proceso de la construcción y limpieza de los datos como si lo hace `ech` (Detomasi 2020) que tiene diferentes funciones para la ECH y permite al usuario crear variables referidas a Vivienda, Educación, Mercado de Trabajo, Ingresos

y Pobreza algo similar con `eph` (Kozlowski et al. 2020) que permite la descarga de datos de la EPH y la creación de variables para analizar la pobreza y el mercado de trabajo.

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

3.3 Diseño de encuestas y estimación de varianza

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

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

Cada metodología depende de cada diseño y variables a estimar, por esto es que existen diferentes metodologías y paquetes como `gustave` (Chevalier 2023) , `vardpoor` (Breidaks, Liberts, and Ivanova 2020), `svrep` (Schneider 2023) y `samplingVarEst` (Emilio Lopez Escobar, Zamudio, and Rosas 2023), aunque existen similitudes entre implementaciones y métodos es difícil encontrar una implementación que permita la estimación de varianzas de manera sencilla y que permita la reutilización de los pasos de limpieza y transformación de los datos, esto puede ser complicado para usuarios que no tienen experiencia en el procesamiento de encuestas por muestreo y que buscan una herramienta que les permita realizar este tipo de análisis de manera sencilla y visual.

Resumen de las implementaciones

En la Tabla 3.1 a continuación se presenta un resumen de las implementaciones de los paquetes mencionados anteriormente:

Como se puede observar en la Tabla 3.1, `metasurvey` busca llenar el vacío de la reutilización de los pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo, aprovechando como dependencia `survey` y `svrep` para

TABLE 3.1: 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`.	2024-10-05T17:06:52Z
gustave	Cuenta con una función principal `qvar` que no utiliza el diseño muestral como argumento sino que se deben de ingresar las probabilidades de inclusión y no queda del todo claro los estadísticos que se pueden obtener. Además, no permite la reutilización de los pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo. Si bien tiene diferentes implementaciones de aproximación de varianzas como la de Deville y Tillé, no es la forma de estimación que utiliza `metasurvey` por defecto aunque puede ser una alternativa para futuras implementaciones.	2024-01-15T16:56:39Z
vardpoor	Este paquete utiliza la estimación de varianzas por el método de último conglomerado, otras métricas como el efecto diseño, linealización del estimador de razón, coeficiente de Gini y diferentes estadísticos enfocados en la desigualdad. Si bien es una implementación interesante, su sintaxis es particular y ajena a la del paquete `survey`, tiene muy pocas dependencias pero puede ser implementado como motor de estimación de varianzas en futuras implementaciones.	2022-02-17T14:50:27Z
convey	Este paquete sí utiliza el diseño muestral como argumento e implementa algunos estadísticos que se incluyen en `vardpoor`, sin embargo, no permite la reutilización de pasos de limpieza y transformación de los datos de manera sencilla y visualizar el flujo de trabajo. Sin embargo, al tener una implementación del diseño muestral, se sincroniza muy bien con `metasurvey`. Los estadísticos pueden ser utilizados de forma nativa en `metasurvey`.	2024-10-17T13:19:17Z
svrep	Extensión del paquete `survey` para trabajar con réplicas sobre los pesos muestrales, permite la estimación de varianzas por el método de Bootstrap, Jackknife y replicación de balanceo. La integración con el diseño del tipo `bootstrap` y `jackknife`. Este paquete es utilizado como dependencia **clave** para ocupar los principales estimadores poblacionales y diseño muestral y es utilizado en `metasurvey` para la estimación de varianzas.	2024-04-26T18:40:02Z

Datos extraídos desde GitHub a través de la API.

la estimación de varianzas y la generación de indicadores sociales, permitiendo la generación de reportes de manera sencilla y visualizar el flujo de trabajo de manera gráfica.

3.4 Desarrollos de paquetes en R

Actualmente R es uno de los lenguajes de programación más utilizados en la comunidad científica y parte de la industria cuando se realizan análisis de datos, esto se debe a la gran comunidad de desarrolladores y la cantidad de paquetes que se encuentran disponibles en CRAN y GitHub. En la actualidad, existen diferentes organizaciones que promueven la ciencia abierta y la reproducibilidad en la investigación científica, proporcionando herramientas y guías para promover la ciencia abierta mediante R.

Sin embargo 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 2011b) 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.

Código 3.1 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"
      )
    )
  }
)
```

Al tener una serie de test automatizados como el presentado en el Código 3.1 se realice antes de realizar cambios en el código fuente pueda el desarrollador ejecutar los tests y verificar que no existan problemas ante un cambio o una refactorización del código fuente. 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 3.2, 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 3.2 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 3.3, 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 3.3 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

Desarrollo y metodología

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

En este capítulo se divide en dos partes, la primera parte se centra en la metodología de los métodos de estimación de varianzas y errores estándar, donde se describen los diferentes métodos de remuestreo y se explican las ventajas de los pesos replicados. La segunda parte se centra en el desarrollo e implementación de las diferentes partes del paquete y la meta-programación. Los ejemplos de código se muestran con código real del paquete `metasurvey` y 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 de la misma.

4.1 Estimación de de los errores estándar

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, entre otros (Särndal, Swensson, and Wretman 2003).

En este trabajo vamos a centrarnos 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 órden son muy complejas de calcular por lo que una estimación directa no es muy factible además de que 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 ultimo conglomerado, donde se asume que la variabilidad proviene unicamente 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.

4.1.1 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 contamos 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 extrapola esta variabilidad para obtener una estimación que se asemeje a la que se obtendría si tuviéramos múltiples muestras independientes.

4.1.1.1 Réplicas de Mitad de Muestra

Para entender mejor este método, consideremos un diseño estratificado en el que se seleccionan dos unidades por estrato. Si dividimos 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 tenemos dos mitades, podemos 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.

4.1.1.2 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:

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

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

4.1.1.3 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). Esto hace que la varianza deba considerar la correlación entre unidades seleccionadas en cada etapa.

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

- En un diseño **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.

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

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.

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

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

4.1.1.4 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 tipos de estadísticas, 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` proporcionan 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 replicas 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.

4.2 Desarrollo e Implementación

En esta sección se describen las diferentes partes del paquete `metasurvey` y su implementación, incluyendo la estructura del paquete, las funciones principales y la forma en que se implementan los métodos de estimación de varianzas y errores estándar. El repositorio de `metasurvey` está disponible en GitHub y sigue la estructura estándar de un paquete de R, tal como se menciona en lasección 2.2.

4.2.1 Dependencias

Un aspecto destacado del paquete `metasurvey` es su uso limitado de dependencias externas, cada una seleccionada por su propósito específico. Por ejemplo, `survey` (Lumley 2024b) se utiliza para el procesamiento de encuestas, `data.table` (Barrett et al. 2024) facilita la manipulación eficiente de datos, `R6` (Chang 2022) permite la programación orientada a objetos, y `glue` (Hester and Bryan 2024) contribuye a la interpolación de cadenas, mejorando la legibilidad del código al crear o modificar fragmentos de texto. Además, `jsonlite` (Ooms 2014) se encarga de la lectura y escritura de archivos JSON, lo cual resulta esencial para compartir las configuraciones obtenidas a través de la API de `metasurvey`, que emplea una base de datos NoSQL, mientras que `httr` (Wickham 2023) gestiona las peticiones HTTP.

También se incluyen algunas dependencias adicionales para mejorar la experiencia del usuario, como `visNetwork` (V., Contributors, and Thieurmel 2022) para la visualización de recetas y pasos, y `crayon` para la impresión de mensajes en la consola con colores.

La optimización de las dependencias es un aspecto importante en el desarrollo de paquetes, ya que influye en la eficiencia y la velocidad de carga. Por lo tanto, se ha

procurado mantener un equilibrio entre la funcionalidad y la eficiencia, evitando la inclusión de paquetes innecesarios que puedan ralentizar el rendimiento del paquete.

Existe una versión previa a `metasurvey` llamada srvyuRu donde las dependencias eran muy amplias. El uso de `dplyr` (Wickham et al. 2023) y `tidyverse` (Wickham et al. 2019) hacía que el paquete fuera muy pesado y lento, junto al paquete `rlang` (Henry and Wickham 2024), que si bien es muy útil para la implementación de la metaprogramación, incrementaba las dependencias de manera innecesaria.

Dentro de `srvyuRu`, también se utilizaba `shiny` (Chang 2022) para la implementación de una interfaz gráfica. Sin embargo, se decidió no incluir esta funcionalidad en `metasurvey` ya que se consideró que no era esencial y que podía ser implementada en un paquete independiente.

La versión anterior fue motivada por la necesidad de contar con una herramienta que automatizara el proceso de recodificación de variables y cálculo de indicadores, para compatibilizar los indicadores de la EAI y ECH para el portal PRISMA en la sección de Innovación y Género respectivamente. Sin embargo, la implementación de la interfaz gráfica no fue exitosa, ya que permitía al usuario crear recetas y pasos de forma gráfica, lo cual llevaba a una complejidad innecesaria y a un paquete muy pesado debido a la inclusión de dependencias innecesarias.

4.2.2 Ejemplos de la implementación

4.2.2.1 Carga de una encuesta

El paquete puede dividirse en dos partes principales: la carga y procesamiento de encuestas, y la estimación de errores estándar. Dentro de lo que es la carga y procesamiento de encuestas, se incluyen funciones para cargar encuestas en diferentes formatos, como 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 se encarga de cargar la encuesta y realizar las operaciones básicas mencionadas anteriormente. Dentro de ella podemos ver que es simplemente un wrapper de diferentes paquetes para cargar la encuesta ya sea `read.spss` del paquete `foregin` (R Core Team 2023a) 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, 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 Chapter 5 se menciona como utilizar replicas brindadas por la institución que publica los microdatos y estimadores de cambios netos compuestos.

Una vez definida la carga de datos dentro de la misma implementación de 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 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 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.

4.2.2.2 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 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.

Código 4.2 Definición de la clase `Step` con sus atributos y método `initialize` para inicializar la clase.

```
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(name, edition, survey_type, type, new_var, exprs, call, svy_bef
      self$name <- name
      self$edition <- edition
      self$survey_type <- survey_type
      self$type <- type
      self$new_var <- new_var
      self$exprs <- exprs
      self$call <- call
      self$svy_before <- svy_before
      self$default_engine <- default_engine
      self$depends_on <- depends_on
      self$comments <- comments
      self$bake <- bake
    }
  )
)
```

El principal uso de `R6` y no de `S3` o `S4` es que `R6` permite la creación de objetos con estado, 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 mas sencilla lo que permitió implementar el proceso perezoso de los pasos y recetas donde se aplican los pasos y recetas solo cuando se necesita y no de forma inmediata así como también el uso de referencias y no de copias de los objetos optimizando el uso de memoria. Se puede ver mas información de esto último en la sección 4.2.2.3.

4.2.2.3 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 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)
```

4.2.2.4 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, lo cual permite que las operaciones se realicen de forma más rápida y eficiente. 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)
```

4.2.3 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.

En el código 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

Código 4.5 Ejemplo de función para encontrar dependencias de variables en una expresión de un step. Existen otros ejemplos de meta-programación en el paquete aunque son más complejos y no se muestran en este documento. Puede ver ejemplos en la función `compute` y `recode` del paquete `metasurvey` funciones internas que se encargan de aplicar los pasos y recetas a la encuesta. Step.R

```
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))
}
```

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.

Capítulo 5

Casos de uso

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

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

5.1 Encuesta Continua de Hogares

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

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

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

- Mencionar más sobre los paneles que parten desde 2021
- Mencionar las bases compatibilizadas del IECON

A continuación se presentan algunos ejemplos de como poder replicar los resultados de las tasas de mercado de trabajo tanto a nivel mensual, trimestral y anual utilizando la encuesta continua de hogares, utilizando los microdatos de la encuesta en 2023.

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.

```
path_dir <- here::here("example-data", "ech", "ech_2023")
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"
    )
  )
)
```

El objeto `ech_2023` tiene dentro los microdatos de implantación y seguimiento en conjunto con las replicas boostrap, para asociar a cada conjunto de microdatos se extrae el patrón de tiempo de los nombres de los archivos de las replicas bootstrap y se asocia a cada conjunto de microdatos. Es importante notar que se puede extraer los microdatos para un mes en particular y poder ver el diseño que se crea utilizando el paquete `survey`:.

Código 5.2 Obtener los microdatos de la encuesta continua de hogares para el mes de enero de 2023 y ver el diseño de la encuesta.

```
ech_01 <- get_follow_up(ech_2023, index = 1)[[1]]
ech_01$design
#> $monthly
#> Call: svrepdesign.default(id = psu, weights = as.formula(paste("~",
#>     x$weight)), data = data_aux, repweights = x$replicate_pattern,
#>     type = x$replicate_type)
#> Survey bootstrap with 1000 replicates.
```

También pueden agruparse los microdatos de seguimiento en un trimestre en particular:

```
extract_surveys(
  ech_2023,
  quarterly = 1
)
#> Type: ECH_2023
#> Periodicity: Periodicity of pool quarterly each survey monthly
#> Steps: None
#> Recipes: None
#> Group: Q1
```

El agrupar los microdatos también puede utilizarse para obtener indicadores de medias móviles como se recomienda cuando se realizan estimaciones en diferentes dominios geográficos o grupos de edad.

Para construir las variables de interés el usuario puede descargar las recetas desde la API de metasurvey y aplicarlas a los microdatos de la encuesta continua de hogares o puede crear sus propias recetas y aplicarlas a los microdatos. A continuación se presentan algunas recetas que se pueden utilizar para calcular las tasas de mercado de trabajo a nivel mensual, trimestral y anual.

```
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"
  )
```

Dentro

- Poner estimación a nivel mensual, trimestral y anual
- Incluir algún dominio de estimación
- Descargar recetas desde la API de metasurvey
- Multiples años

5.2 EPH

- Mencionar la flexibilidad de metasurvey

```
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
  )
```

```
metasurvey::view_graph(eph2022_3)
```

Capítulo 6

Pasos a futuro

Capítulo 7

Bibliografía

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

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

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

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.

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

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

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

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

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

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

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

Escobar, Emilio L., and Yves G. Berger. 2013. "A New Replicate Variance Estimator for Unequal Probability Sampling Without Replacement." *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* 41 (3): 508–24. https://www.jstor.org/stable/43186201.

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

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

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.

Henry, Lionel, and Hadley Wickham. 2024. *Rlang: Functions for Base Types and Core r and 'Tidyverse' Features*. https://rlang.r-lib.org.

Hester, Jim, and Jennifer Bryan. 2024. *Glue: Interpreted String Literals*. https://CRAN.R-project.org/package=glue.

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

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

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

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

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

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

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

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

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

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

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

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

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

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019.

"Model Cards for Model Reporting." In, 220–29. https://doi.org/10.1145/3287560.3287596.

Ooms, Jeroen. 2014. "The Jsonlite Package: A Practical and Consistent Mapping Between JSON Data and r Objects." *arXiv:1403.2805 [Stat.CO]*. https://arxiv.org/abs/1403.2805.

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

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

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

R Core Team. 2023a. *Foreign: Read Data Stored by 'Minitab', 's', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...* https://CRAN.R-project.org/package=foreign.

———. 2023b. *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, Carl-Erik, Bengt Swensson, and Jan Wretman. 2003. *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." 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.

V., Almende B., Contributors, and Benoit Thieurmel. 2022. *visNetwork: Network Visualization Using 'Vis.js' Library*. https://CRAN.R-project.org/package=visNetwork.

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

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

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

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

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the Tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

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

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

Apéndice A

Frequently Asked Questions

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

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

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

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

{`\hypersetup{hidelinks}`}.

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

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