
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 metaprogramación y técnicas de remuestreo para ofrecer resultados precisos, evaluar la incertidumbre y fomentar la reproducibilidad. A diferencia de otras bibliotecas, metaSurvey combina flexibilidad mediante metaprogramación con las capacidades de procesamiento de encuestas del paquete survey. Los objetivos incluyen proporcionar una herramienta útil, incorporar técnicas de remuestreo para usuarios no expertos, permitir la generación de 'recetas' personalizadas, y fomentar la contribución de la comunidad. Se destaca como alternativa a paquetes propietarios, enfocándose en la transparencia y reproducibilidad para mejorar la confiabilidad de las estimaciones poblacionales.

# *Agradecimientos*

Acá le voy a agradcer a alguien?

# Índice

# Figuras

# Tablas

*Acá va la dedicatoria*

# Descripción del proyecto

Acá poner una descripción del proyecto en sí

# Capítulo 1

# Introducción

En este documento se presenta el desarollo del paquete metaSurvey disponible en R (R Core Team, 2023). El objetivo principal de metaSurvey es permitir que el usuario pueda construir indicadores de manera reproducible y transparente, teniendo el usuario un control total sobre el proceso de transformación de los microdatos a indicadores.

A lo largo del documento se mencionan diferentes conceptos para el desarrollo del paquete, como la meta-programación, conceptos de Inferencia en Poblaciones finitas, esquema de trabajo para desarrollar paquetes en R, etc. Se presentarán ejemplos de cómo utilizar el paquete metaSurvey para construir indicadores de mercado de trabajo a partir de los microdatos de la Encuesta Continua de Hogares (**ECH**) del Instituto Nacional de Estadística de Uruguay (INE) y para mostrar su flexibilidad un ejemplo con la Encuesta Permanente de Hogares (**EPH**) del Instituto Nacional de Estadística y Censos de Argentina (INDEC).

La motivación principal del desarrollo del paquete fue la necesidad de contar con un paquete que permita al usuario tener un control total y transparente sobre el proceso de transformación de los microdatos a indicadores. En general, los paquetes que existen en R para el análisis de encuestas por muestreo son muy sensibles a la estructura y las variables que componen a la encuesta. En general, un cambio en la estructura de la encuesta implica una nueva versión del paquete utilizado para obtener los indicadores, siendo poco flexible a cambios en la estructura de la encuesta que en la practica pueden ser muy cambiantes y generalmente el usuario cuenta con una función de alto nivel que actua como caja negra donde no se permite modificar el código para adaptarlo a sus necesidades o saber cada paso que se realiza para obtener el indicador sin tener que leer el código fuente o la documentación adjunta.

El problema de sensibilidad a la estructura de la encuesta puede verse en el paquete *ech* (Detomasi, 2020) donde en el existen funciones para crear variables de mercado de trabajo, educación o ingresos pero estas funciones dependen de la existencia de ciertas variables en la encuesta donde la estructura puede cambiar de una versión a otra de la encuesta y sin revisar el cuerpo de la función no se sabe el proceso de construcción de variables. Algo similar ocurre con el paquete *eph* (Kozlowski et al., 2020), donde se tienen funciones de alto nivel que no permiten modificar el código para adaptarlo a sus necesidades o saber cada paso que se realiza para obtener el indicador sin tener inspeccionar a fondo el como se construyen las funciones del paquete.

Esta inspección del código fuente como puede ser consultar el repositorio de github del paquete o revisar la definición de la función, puede ser una tarea tediosa y que no garantiza que el usuario pueda entender el proceso de construcción de las variables ya

que puede ser que el código sea 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 por ejemplo el uso de la librería *dplyr* (Wickham et al., 2023) o la librería *tidyr* (Wickham et al., 2024) que son librerías muy populares en R para el manejo de datos pero que el usuario puede no conocer o sea dificil aislar el proceso de construcción de variables y al funcionamiento especifico de la función donde puedan existir código para manejar la forma de presentación, esturcutura del objeto a devolver, etc. Un claro ejemplo de esto puede verse en el paquete *tidycensus* (Walker & Herman, 2024) donde existe una función para obtener datos sobre la migración de la comunidad Estado Unidense pero el usuario no puede y asilar el proceso de recodificación/construcción de variables sobre variables originales y la obtención de datos greográficos [1].

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 puede ser recodificar variables creando grupos en base a criterios complejos, tratamiento de variables continuas como puede ser el ingreso salarial en base a metodología rigurosa y es crucial que sea transparente y entendible para el usuario.

En general obtener la información historica de indicadores es un proceso tedioso y que puede ser propenso a errores, en especial si provienen de encuentas donde su estructura y forma de preguntar o codificación puede cambiar en el tiempo siendo un proceso extenso y dificil de entender hasta llegar a la construcción de esta serie de indicadores y 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 esta ligado a la sintaxis y depende de que el usuario tenga el software instalado y pueda correr el código.

Una vez claro el proceso de creación de variables también es importante tener en cuenta que al obtener indicadores se realiza un proceso de estimación de parámetros poblacionales y sus errores asociados, 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 y brindar recomendaciones sobre la utilidad de la estimación en el caso de que se cuente con una variabilidad alta en la estimación ya que en general, obtener la estimación una vez culminado el proceso de preprocesamiento 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, el paquete metaSurvey pretende ser una herramienta relevante para el trabajo con encuestas buscando solucionar estas limitaciones ya que todo el proceso de transformación de los microdatos a indicadores se realiza a través de una

[1] Aquí puede verse el código fuente de la función `get_flow` del paquete tidycensus donde se puede en la linea 151 la recodificación de variables se hace con una tabla `mig_recodes` y al explorar el contenido puede verse como se recodifican las variables además de que la función también tiene código para manejar la forma de presentación de los datos, manipulación de datos geográficos y la estructura del objeto a devolver.

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, ya que el usuario puede compartir el código y los datos utilizados para obtener los indicadores, mediante lo que denominamos steps y recipes, conformando así una especie de "recetario de cocina" para la construcción de indicadores, pudiendo compartir la construcción en forma visual como un DAG (Directed Acyclic Graph) que permite visualizar el proceso de construcción de indicadores sin tener que abrir un script de R. Además, 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.

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 R, la meta-programación se realiza a través de las funciones eval y parse 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 similares a las que se utilizan en el paquete recipes de la librería tidymodels (Kuhn et al., 2024).

En los siguientes capitulos se mencionaran conceptos clave para el desarrollo del paquete, como la meta-programación, conceptos de Inferencia en Poblaciones finitas, esquema de trabajo para desarrollar paquetes en R, etc. A continuación se mencionarán diferentes antececentes y trabajos relacionados con el paquete metaSurvey donde se utiliza la meta-programación y herramientas en las que fue inspirado el paquete. Luego, se formalizaran diferentes conceptos sobre metodología para la estimación de parámetros poblaciones y su varianza y conceptos de meta-programación y como se utilizan en el desarollo del paquete. Para finalizar, se presentarán ejemplos de cómo utilizar el paquete metaSurvey para construir indicadores de mercado de trabajo a partir de los microdatos de la Encuesta Continua de Hogares (**ECH**) y para mostrar su flexibilidad un ejemplo con la Encuesta Permanente de Hogares (**EPH**).

# Capítulo 2

# Marco conceptual

El objetivo principal de este capítulo es presentar los conceptos básicos que se utilizarán a lo largo de este trabajo en especifico la sección de antecedentes y metodología. En primer lugar, se presentará un breve resumen de como 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. 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 presentara un marco básico de inferencia en poblaciones finitas y encuestas por muestreo para luego presentar diferentes métodos de estimación de parámetros poblacionales y sus respectivos errores estándar.

## 2.1. Desarrollo de paquetes en R

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

Para casí cualquier disciplina científica o en la industria se puede encontrar una comunidad de usuarios que desarollan 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.

### 2.1.1. ¿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 algun 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.

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* (*Roxygen R6 Guide · Mlr-Org/Mlr3 Wiki*, n.d.) para generar la documentación de las funciones.
  - `DESCRIPTION`: Archivo que contiene la descripción del paquete, incluyendo el nombre, versión, descripción, autor, entre otros.
  - `NAMESPACE`: Archivo que contiene la información sobre las funciones que se exportan y las dependencias del paquete.
  - `LICENSE`: Archivo que contiene la licencia bajo la cual se distribuye el paquete.
  - `README.md`: Archivo que contiene información general sobre el paquete.
- **Documentación**: La documentación de las funciones es un componente esencial de un paquete en R, ya que permite que los usuarios puedan entender el funcionamiento de las funciones que se encuentran en el paquete. La documentación de las funciones se realiza utilizando el sistema de documentación de R, que se basa en el uso de comentarios en el código fuente de las funciones.
- **Pruebas**: Es importante que el paquete tenga pruebas que permitan verificar que las funciones se comportan de la manera esperada. Las pruebas se realizan utilizando el paquete *testthat* (Wickham, 2011) que permite realizar pruebas unitarias.
- **Control de versiones**: Es importante que el paquete tenga un sistema de control de versiones que permita llevar un registro de los cambios que se realizan en el paquete. El sistema de control de versiones más utilizado en la comunidad de R es `git`.
- **Licencia**: Es importante que el paquete tenga una licencia que permita a los usuarios utilizar, modificar y distribuir el paquete. La licencia más utilizada en la comunidad de R es la licencia MIT.

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

El proceso de chequeo fue automatizado con github actions, por lo que cada vez que se realiza un cambio en el repositorio, se ejecutan los chequeos de CRAN y se notifica si el paquete cumple con los requisitos para ser publicado en caso de que no cumpla

con los requisitos se notifica el error y no puede ser incluido en la rama principal del repositorio hasta que se corrija el error.

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

## 2.2. Paradigmas de programación en R

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

### 2.2.1. Programación funcional

La programación funcional es un paradigma de programación que se basa en el uso de funciones para resolver problemas. En R, las funciones son objetos de primera clase, lo que significa que se pueden utilizar como argumentos de otras funciones, se pueden asignar a variables, entre otros (Wickham, 2019, pp. 204–281). 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.

### 2.2.2. Programación orientada a objetos

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

- **Clases y objetos**: En R, las clases son plantillas que definen la estructura y el comportamiento de los objetos y los objetos son instancias de clases. En R, las

clases se definen utilizando la función `setClass` y los objetos se crean utilizando la función `new`.

- **Atributos y métodos**: En R, los atributos son variables que almacenan información sobre el estado de un objeto y los métodos son funciones que permiten modificar el estado de un objeto. En R, los atributos se definen utilizando la función `setClass` y los métodos se definen utilizando la función `setMethod`.

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

De forma similar se modelan las clases `Step`, `Recipe` y `Workflow` elementos cruciales en el ecosistema de metaSurvey donde se definen los pasos de preprocesamiento, recetas de preprocesamiento y flujos de trabajo respectivamente. En este caso particular se utiliza el paquete *R6* (Chang, 2022)que permite definir clases de manera sencilla y eficiente además de permitir la herencia de clases y la definición de métodos y atributos de manera sencilla.

### 2.2.3. Meta-programación

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

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

## 2.3. Inferencia en muestreo de poblaciones finitas

### 2.3.1. Diseño de muestreo

## Capítulo 3

# Antecedentes

## Capítulo 4

# Metodología

# Capítulo 5

# Resultados

Acá va la viñeta Use recipes

## 5.1. ECH

```
library(magrittr)

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

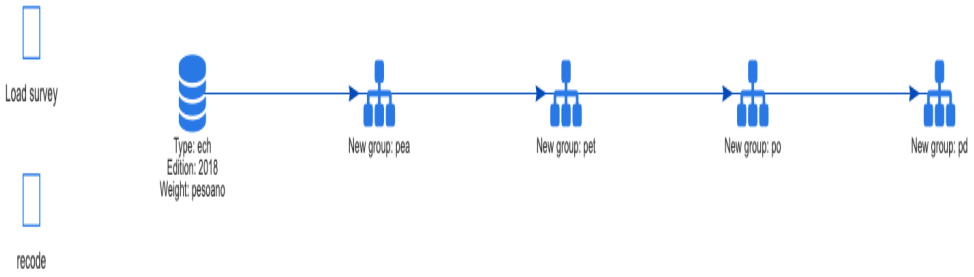
```
Engine: data.table
```

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

ech_meta_steps = ech_meta %>%
  metaSurvey::step_recode(
    "pea",
    pobpcoac %in% 2:5 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "pet",
    pobpcoac != 1 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "po",
    pobpcoac == 2 ~ 1,
    .default = 0
  ) %>%
  metaSurvey::step_recode(
    "pd",
    pobpcoac %in% 3:5 ~ 1,
    .default = 0
```

```
    )
```

Select by id

**Type**



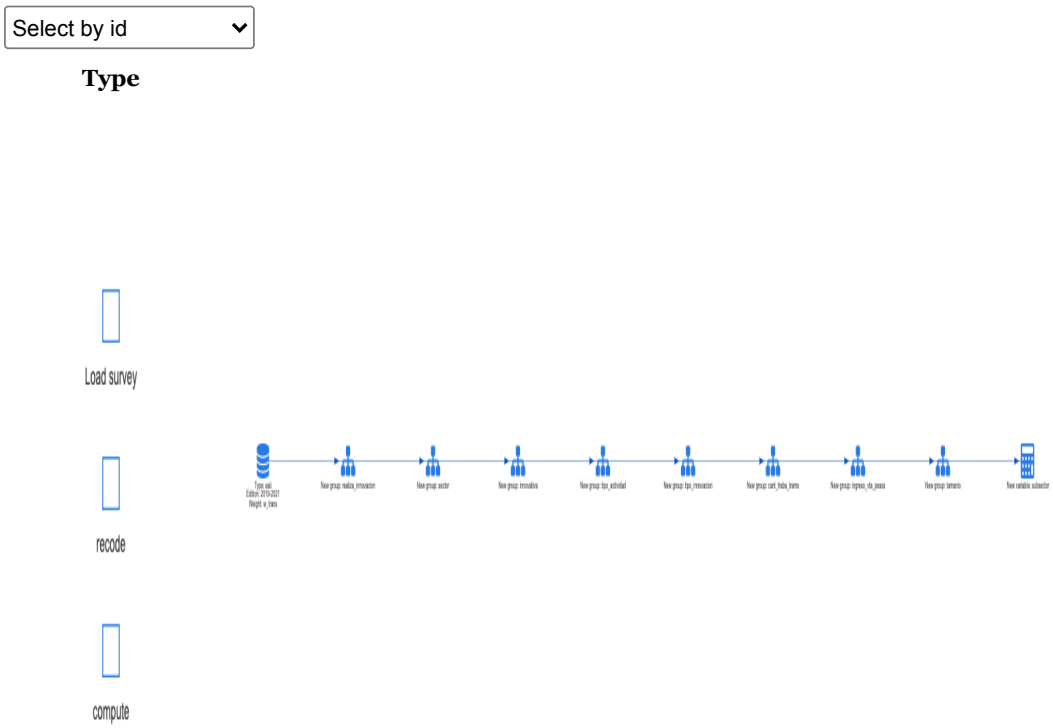

## 5.2. EAII

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

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

new_svy = svy_example %>%
    metaSurvey::step_recode(
        new_var = "realiza_innovacion",
        B1_1_1 == 1 ~ 1,
        B1_2_1 == 1 ~ 1,
        B1_3_1 == 1 ~ 1,
        B1_4_1 == 1 ~ 1,
        B1_5_1 == 1 ~ 1,
```

```
        B1_6_1 == 1 ~ 1,
        B1_7_1 == 1 ~ 1,
        B1_8_1 == 1 ~ 1,
        B1_9_1 == 1 ~ 1,
        .default = 0
    ) %>%
    metaSurvey::step_recode(
        new_var = "sector",
        data.table::between(Division, 10, 33) ~ "Industria",
        data.table::between(Division, 34, 99) ~ "Servicios",
        Division == "C1" ~ "Industria",
        Division == "C2" ~ "Servicios",
        Division == "E1" ~ "Servicios"
    ) %>%
    metaSurvey::step_recode(
        new_var = "innovativa",
        E1_1_1 == 1 ~ 1,
        E1_2_1 == 1 ~ 1,
        .default = 0
    ) %>%
    metaSurvey::step_recode(
        new_var = "tipo_actividad",
        B1_1_1 == 1 ~ "I + D Interna",
        B1_2_1 == 1 ~ "I + D Externa",
        B1_3_1 == 1 ~ "Bienes de Capital",
        B1_4_1 == 1 ~ "Software",
        B1_5_1 == 1 ~ "Propiedad Intelectual",
        B1_6_1 == 1 ~ "Ingeniería",
        B1_7_1 == 1 ~ "Capacitación",
        B1_8_1 == 1 ~ "Marketing",
        B1_9_1 == 1 ~ "Gestión",
        .default = "Otra"
    ) %>%
    metaSurvey::step_recode(
        new_var = "tipo_innovacion",
        E1_1_1 == 1 ~ "Producto",
        E1_2_1 == 1 ~ "Proceso",
        .default = "Otra"
    ) %>%
    metaSurvey::step_recode(
        new_var = "cant_traba_tramo",
        data.table::between(IG_4_1_3, 0, 4) ~ "1",
        data.table::between(IG_4_1_3, 5, 19) ~ "2",
        data.table::between(IG_4_1_3, 20, 99) ~ "3",
        IG_4_1_3 > 99 ~ "4"
    ) %>%
    metaSurvey::step_recode(
        new_var = "ingreso_vta_pesos",
        data.table::between(IG_5_1_1_3, 0, 9942787) ~ "1",
        data.table::between(IG_5_1_1_3, 9942788, 49713934) ~ "2", # nolint
```

```
        data.table::between(IG_5_1_1_3, 49713935, 372854507) ~ "3", # nolint
        IG_5_1_1_3 > 372854507 ~ "4"
    ) %>%
    metaSurvey::step_recode(
        new_var = "tamanio",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "1" ~ "Pequenias",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "2" ~ "Pequenias",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "1" ~ "Pequenias",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "2" ~ "Pequenias",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "3" ~ "Medianas",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "2" ~ "Medianas",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "1" ~ "Medianas",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "3" ~ "Medianas",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "3" ~ "Medianas",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "4" ~ "Grandes",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "3" ~ "Grandes",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "2" ~ "Grandes",
        cant_traba_tramo == "4" & ingreso_vta_pesos == "1" ~ "Grandes",
        cant_traba_tramo == "1" & ingreso_vta_pesos == "4" ~ "Grandes",
        cant_traba_tramo == "2" & ingreso_vta_pesos == "4" ~ "Grandes",
        cant_traba_tramo == "3" & ingreso_vta_pesos == "4" ~ "Grandes"
    ) %>%
    metaSurvey::step_compute(
        subsector = Division
    )

metaSurvey::get_metadata(new_svy)
```

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



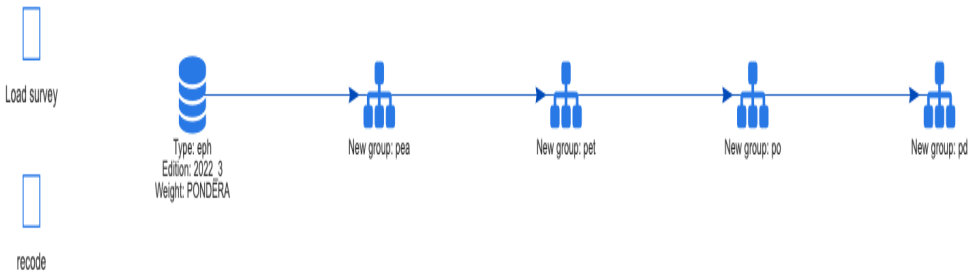

## 5.3. EPH

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

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

Select by id

**Type**



## Capítulo 6

# Infraestructura

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

## Capítulo 7

# Resultados

# Soy un apaendice

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

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

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

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

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

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

*Roxygen R6 guide · mlr-org/mlr3 wiki*. (n.d.). https://github.com/mlr-org/mlr3/wiki/Roxygen-R6-Guide

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

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

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

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

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

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