Análisis de encuestas con srvyr : : **CHEAT SHEET (1 de 2)**

Describir el diseño de la encuesta

Para analizar correctamente datos de encuestas, crea un **objeto de diseño de encuesta.** Este objeto contiene los datos, los pesos y otros metadatos usados en el análisis.

```
design <- as_survey_design(
  .data      = my_data_frame,
  ids        = _, (Requerido) - Variable(s) que identifican las unidades de muestreo (o "conglomerados")
  strata     = _,  (Opcional) - Variable(s) que identifican los estratos de muestreo
  fpc        = _,  (Opcional) - Variable(s) con los tamaños poblacionales o fracciones de muestreo de cada estrato.
  probs      = _,  (Opcional) - Variable(s) con las probabilidades de selección de cada unidad de muestreo
  weights    = _,  (Opcional) - Variable con el peso muestral de cada observación de los datos
)
```

PARA MUESTRAS MULTIETÁPICAS: Una variable por cada etapa de muestreo: p. ej., `c(SCHOOL, STUDENT)`. Solo aplica a **ids**, **strata**, **fpc** y **probs**.

EJEMPLO: conglomerados estratificados

Encuesta escolar: incluye a *todos* los estudiantes de cuatro escuelas muestreadas (dos en cada uno de dos distritos)

Obs	DISTRICT	SCHOOL	SCHOOL_PROB	DISTRICT_SIZE	WEIGHT
1	District 1	School 1	0.40	5	2.5
2	District 1	School 1	0.40	5	2.5
3	District 1	School 2	0.40	5	2.5
4	District 1	School 2	0.40	5	2.5
5	District 2	School 3	0.20	10	5.0
6	District 2	School 3	0.20	10	5.0
7	District 2	School 4	0.20	10	5.0
8	District 2	School 4	0.20	10	5.0

```
design <- as_survey_design(
  .data      = school_survey_data,
  ids        = SCHOOL,
  strata     = DISTRICT,
  fpc        = DISTRICT_SIZE,
  weights    = WEIGHT)
```

EJEMPLO: muestra multietápica

Encuesta escolar: incluye una *muestra* de estudiantes de cuatro escuelas muestreadas (dos en cada uno de dos distritos)

Obs	DISTRICT	SCHOOL	STUDENT	DISTRICT_SIZE	SCHOOL_SIZE	WEIGHT
1	District 1	School 1	Student 1	4	100	100
2	District 1	School 1	Student 2	4	100	100
3	District 1	School 2	Student 3	4	150	150
4	District 1	School 2	Student 4	4	150	150
5	District 2	School 3	Student 5	10	200	500
6	District 2	School 3	Student 6	10	200	500
7	District 2	School 4	Student 7	10	250	625
8	District 2	School 4	Student 8	10	250	625

```
design <- as_survey_design(
  .data      = school_survey_data,
  ids        = c(SCHOOL, STUDENT),
  strata     = c(DISTRICT, SCHOOL),
  fpc        = c(DISTRICT_SIZE, SCHOOL_SIZE),
  weights    = WEIGHT)
```

CONSEJOS PARA EL OBJETO DE DISEÑO

- Si cada observación de los datos es su propia unidad de muestreo, usa `ids = NULL`
- Si analizas solo un subconjunto de los datos, crea primero el objeto de diseño con TODOS los datos y *luego* filtra el objeto de diseño con `filter()`
- Si no se usa el argumento weights, los pesos se crean automáticamente a partir del argumento probs (si existe) o del argumento fpc (si no hay probs)

MANEJO DE "UPM SOLITARIAS"

- Una "UPM solitaria" (lonely PSU) es la única unidad de muestreo de su estrato (o "estrato unitario").
- Esto puede ser un problema al estimar varianzas. Puedes usar la opción "survey.lonely.psu" para resolverlo.
- Al analizar subconjuntos de los datos (es decir, "dominios"), esta opción se puede aplicar a subconjuntos con una sola UPM con: `options(survey.adjust.domain.lonely = TRUE)`

`options(survey.lonely.psu = "fail")`

Muestra un mensaje de error si hay alguna UPM solitaria

`options(survey.lonely.psu = "adjust")`

Supone que la UPM solitaria proviene de un estrato cuya media es igual a la de una UPM promedio.

`options(survey.lonely.psu = "average")`

Supone que la UPM solitaria aporta a la varianza lo mismo que un estrato promedio.

`options(survey.lonely.psu = "remove")`

Ignora la UPM solitaria al estimar varianzas.

Bases de datos

Con datos grandes, puedes guardarlos en una tabla de una base de datos y cargarlos en R solo cuando haga falta. Para saber más, mira la viñeta de srvyr **"Databases in srvyr"**.

1. Crea una conexión a la base de datos con el paquete DBI.
2. Usa dplyr::tbl() para elegir una tabla de la base de datos
3. Usa la tabla en .data de as_survey_design() o as_survey_rep()

```
library(DBI)
library(dplyr)

db_conn <- dbConnect([dbdriver],…)

tbl(db_conn, "TABLE_NAME") |>
  as_survey_design(…)
```

Pesos replicados

Como alternativa al objeto de diseño de encuesta, puedes crear un **objeto de diseño con réplicas** con los **pesos replicados** de los datos.

```
design <- as_survey_rep(
  .data      = my_data_frame,
  weights    = _,
  repweights = _,
  type       = _,
  mse        = _
)
```

ARGUMENTOS

- **weights:** Variable con los pesos de la muestra completa
- **repweights**: Variables con los pesos replicados. Funciones auxiliares útiles para listarlas:
  - **num_range("REP_WGT", 1:80)**
  - **starts_with("REP_WGT")**
- **type:** Método con el que se crearon los pesos replicados (p. ej., "bootstrap", "JK1" o "BRR"). Para más flexibilidad, usa type = "other" y los argumentos "scale" y "rscales" (ver "Factores de escala" abajo).
- **mse:** Con TRUE, la varianza usa cuadrados en torno a la estimación de la muestra completa; con FALSE, en torno a la media de las réplicas
- **degf**: (Opcional) Indica los grados de libertad

CÓDIGO DE EJEMPLO

```
design <- as_survey_rep(
  .data      = my_data_frame,
  weights    = FULL_SAMPLE_WGT,
  repweights = num_range("REP_WGT", 1:50),
  type       = "bootstrap",
  mse        = TRUE
)
```

FACTORES DE ESCALA

Todos los tipos de replicación usan la misma fórmula de varianza, con factores de escala **scale** y **rscales**, que se calculan automáticamente salvo con **type = "other" o "JKn"**; en ese caso se pueden usar **estos argumentos:**

- **scale:** Factor de escala global
- **rscales:** Factores de escala de cada réplica

$$\mathrm{V}(\hat{\theta}) = A \sum_{r=1}^{R} a_r \left(\hat{\theta}_r - \hat{\theta}\right)^2$$

scale · rscales

si **mse=FALSE**, se reemplaza por el promedio: $\frac{1}{R}\sum_{r=1}^{R} \hat{\theta}_r$

CC by SA Greg Freedman Ellis y Ben Schneider • Más información en **http://gdfe.co/srvyr/** • versión del paquete 1.3.0 • Actualizado: 2025-01 • Traducido por Anthony Romero-Cerdán • anthonyromero.netlify.app

Análisis de encuestas con srvyr : : **CHEAT SHEET (2 de 2)**



Tras crear un **objeto de diseño de encuesta** con srvyr, puedes manipular los datos y calcular resúmenes con los verbos de dplyr, como **filter()** o **summarize()**. srvyr ofrece funciones especializadas de resumen estadístico para calcular estimaciones ponderadas junto con errores estándar e intervalos de confianza.

Manipular datos



filter(.data, …, .preserve = FALSE**)**
Extrae las filas que cumplen criterios lógicos.

design |> filter(AGE >= 16, AGE <= 65)



mutate(.data, …**)**
Calcula nueva(s) columna(s)**.**

design |> mutate(BMI = WGT/HEIGHT^2)

Funciones de resumen

Aplica funciones de resumen a las columnas para crear una tabla nueva de estadísticos de resumen. Las funciones de resumen reciben vectores y devuelven un valor (ver reverso).



summarize(.data, …**)**
Calcula una tabla de resúmenes.

design |> summarise(mean_age = survey_mean(AGE))



survey_count(x, …, sort = FALSE, name = "n"**)**
Conteos ponderados y errores estándar de cada grupo definido por las variables en …

survey_count(design, cyl)

Agrupar casos

Usa **group_by(.data, …)** para crear una copia "agrupada" de una tabla según las columnas en ...

dplyr y srvyr manipulan cada "grupo" por separado y combinan los resultados.

CONSEJO: las operaciones agrupadas funcionan con `summarize()`, `mutate()` y `filter()`



```
design |>
  group_by( REGION ) |>
  summarize(
    mean_age = survey_mean( AGE )
  )
```

Operaciones por columnas

Usa funciones de **dplyr** como `across()` para aplicar resúmenes a varias columnas.

Ejemplo: design |> summarise(across(where(is.numeric), survey_mean))

Más información sobre las funciones por columnas de **dplyr** aquí:
https://dplyr.tidyverse.org/articles/colwise.html

Funciones de resumen estadístico

Estas funciones se usan ***dentro*** de las funciones de dplyr **summarize()** y **mutate()** y devuelven una estimación y su error estándar (con el sufijo "_se")

Código de ejemplo

```
design |>
  group_by(region) |>
  summarize(pop_size    = survey_total(),
            mean_age = survey_mean(AGE))
```

Resultado de ejemplo

region	pop_size	pop_size_se	mean_age	mean_age_se
North	15,342,875	25,410	46	1.2
South	5,861,942	37,902	44	1.1

survey_total(x**)**

Estima el total poblacional, ya sea de una variable numérica **x** o, si no se indica **x**, del grupo actual.

survey_mean(x, …**)**

Estima la media poblacional.

survey_mean(x, proportion = TRUE, prop_method = "logit"**)**

Estima una proporción si **x** es una variable binaria con valores 0 y 1 o una variable lógica.

x puede ser una expresión como `AGE > 25`

Usa el argumento **prop_method** para elegir un método especializado para calcular intervalos de confianza.

survey_median(x**)**

Estima la mediana poblacional.

survey_quantile(x, quantiles = 0.5**)**

Estima cuantiles poblacionales, como la mediana o los cuartiles.

survey_sd(x**)**

Estima la desviación estándar poblacional de una variable.

survey_var(x**)**

Estima la varianza poblacional de una variable.

survey_ratio(numerator, denominator**)**

Estima la razón entre las medias poblacionales de dos variables.

survey_corr(x, y**)**

Estima la correlación poblacional entre dos variables.

ERRORES ESTÁNDAR E INTERVALOS DE CONFIANZA

Toda función de resumen estadístico incluye un argumento **vartype** con el que se puede pedir una o más de las siguientes medidas de variación muestral.

- **"var"**: La varianza muestral
- **"se"**: El error estándar (incluido por defecto)
- **"cv"**: El coeficiente de variación
- **"ci"**: Un intervalo de confianza; el nivel de confianza se controla con el argumento **level**

Código de ejemplo

```
design |>
  summarize(
    age = survey_mean(AGE, vartype = c("se", "ci"), level = 0.95)
  )
```

Proporciones y porcentajes

En datos agrupados, puedes estimar la proporción de la población total de cada grupo con **survey_prop()**.

survey_prop(…, prop_method = "logit"**)**

Por defecto, con *varias* variables de agrupación, **x** e **y**, se obtienen proporciones de **y** dentro de las categorías de **x**.

La función **interact()** permite obtener proporciones para combinaciones de variables.

Proporciones anidadas: proporción de una variable *dentro* de otra

```
design |>
  group_by(region, agree) |>
  summarize(prop = survey_prop())
```

region	agree	prop	prop_se
North	Yes	0.9	0.01
North	No	0.1	0.01
South	Yes	0.5	0.05
South	No	0.5	0.05

Proporciones cruzadas: proporciones de *combinaciones* de variables

```
design |>
  group_by(interact(region, agree))
|>
  summarize(prop = survey_prop())
```

region	agree	prop	prop_se
North	Yes	0.675	0.01
North	No	0.075	0.01
South	Yes	0.125	0.05
South	No	0.125	0.05

CC by SA Greg Freedman Ellis y Ben Schneider • Más información en **http://gdfe.co/srvyr/** • versión del paquete 1.3.0 • Actualizado: 2025-01 • Traducido por Anthony Romero-Cerdán • anthonyromero.netlify.app