

Package: complexr (via r-universe)

July 6, 2026

Title 'Shiny' Interface for Complex Survey Data Analysis

Version 1.0.1

Description Provides a 'Shiny'-based interactive interface for the analysis of complex survey data. The package supports data import from multiple file formats, construction and diagnosis of complex survey designs with stratification, clustering, and sampling weights, as well as estimation of means, totals, proportions, associated standard errors, confidence intervals, and design effects. The multilingual interface facilitates the analysis and visualization of survey results without requiring specialized programming knowledge.

License GPL (>= 3)

URL <https://github.com/stalynGuerrero/complexr>

BugReports <https://github.com/stalynGuerrero/complexr/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr, tibble, tidyr, readr, readxl, haven, rlang, purrr, shiny, tools, stats, utils, srvyr, survey, magrittr, ggplot2, jsonlite, writexl, DT, tidyselect

Suggests convey, knitr, rmarkdown, quarto, printr, testthat (>= 3.0.0), spelling

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Language en-US

NeedsCompilation no

Author Stalyn Guerrero Gómez [aut, cre]

Maintainer Stalyn Guerrero Gómez <guerrero stalyn@gmail.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-07-06 14:39:38 UTC

RemoteUrl <https://github.com/cran/complexr>

RemoteRef HEAD

RemoteSha 5ecb6e24014473ad9be353d61e63ecbcff9443eb

Contents

app_server	2
app_ui	3
as_survey_design_tbl	3
describe_survey_design	6
estimate_survey	7
format_results_table	12
generate_example_data	13
il8n_load	15
il8n_t	16
mutate_survey_data	16
plot_results_bar	17
read_survey_data	18
run_app	19
validate_simulation	20

Index **21**

app_server	<i>Shiny server for complexr</i>
------------	----------------------------------

Description

Shiny server for complexr

Usage

```
app_server(input, output, session)
```

Arguments

input	Shiny input
output	Shiny output
session	Shiny session

app_ui	<i>Shiny UI for complexr</i>
--------	------------------------------

Description

Shiny UI for complexr

Usage

```
app_ui()
```

Value

A Shiny UI object.

as_survey_design_tbl	<i>Create a complex survey design object</i>
----------------------	--

Description

Constructs a `tbl_svy` design object from survey microdata by encoding the sampling structure required for design-based inference: adjusted weights w_k , stratification variable h , primary sampling units (PSUs) α , and an optional finite population correction (FPC).

Usage

```
as_survey_design_tbl(
  data,
  weight,
  strata = NULL,
  cluster = NULL,
  fpc = NULL,
  nest = TRUE,
  check_psu = TRUE
)
```

Arguments

<code>data</code>	A tibble containing survey microdata with one row per sampled unit k .
<code>weight</code>	Character string. Name of the column holding the adjusted sampling weights w_k . Must be strictly positive and free of NA values.
<code>strata</code>	Optional character string. Name of the column identifying the stratum $h \in \{1, \dots, H\}$ to which each unit belongs. When supplied, variance is estimated within strata.

cluster	Optional character string. Name of the column identifying the PSU $\alpha \in \{1, \dots, \alpha_h\}$ within stratum h . When NULL, a single-stage design is assumed ($\alpha_h = 1$ for all h).
fpc	Optional character string. Name of the column holding the finite population correction value for each unit, used to reduce variance estimates when the sampling fraction n/N is non-negligible.
nest	Logical. Whether PSU labels are nested within strata, i.e. the same PSU code may appear in different strata. Default is TRUE. Set to FALSE only when PSU identifiers are globally unique.
check_psu	Logical. If TRUE (default), emits a warning when PSUs appear in more than one stratum and nest = FALSE, which may indicate a labelling inconsistency.

Details

Let $U = \{1, \dots, N\}$ be the finite population and $s \subset U$ the sample. For each unit $k \in s$, the **first-order inclusion probability** is $\pi_k = \Pr(k \in s) > 0$ and the **basic design weight** is $d_k = 1/\pi_k$. The **adjusted weight** w_k stored in `weight` incorporates any non-response or calibration corrections applied after data collection.

The function wraps `survey::svydesign()` and converts the result to a **svyr** object via `svyr::as_survey()`.

Supported design configurations

Configuration	Arguments supplied
Simple random sampling (SRS)	weight only
Stratified	weight + strata
Single-stage cluster	weight + cluster
Stratified multistage	weight + strata + cluster
Any of the above with FPC	add fpc

Design weights and inclusion probabilities

For a stratified multistage design with H strata and α_h PSUs in stratum h , the Horvitz–Thompson (HT) estimator of the population total is:

$$\hat{Y}_{HT} = \sum_{h=1}^H \sum_{\alpha=1}^{\alpha_h} \sum_{k=1}^{n_{h\alpha}} \omega_{h\alpha k} y_{h\alpha k},$$

where $\omega_{h\alpha k}$ are the adjusted weights of unit k in PSU α of stratum h . The resulting `tbl_svy` object encodes this structure so that all subsequent calls to `estimate_survey` use the correct design-based variance estimator $\hat{V}_p(\hat{Y}_{HT})$.

Lonely PSU handling

When a stratum h contains only one PSU ($\alpha_h = 1$), the within-stratum variance cannot be estimated by Taylor linearization. The function automatically sets `options(survey.lonely.psu = "adjust")` so that variance is approximated by centering the PSU total at the stratum mean, following the conservative recommendation of Cochran (1977).

Finite population correction

When the sampling fraction $f_h = n_h/N_h$ is non-negligible (typically $> 5\%$), supply `fpc` to reduce variance estimates by the factor $(1 - f_h)$.

Value

A `tbl_svy` object (class from **srvyr**) compatible with **srvyr** and **survey** functions. The object carries a "design_vars" attribute that records weight, strata, cluster, fpc and nest for downstream diagnostics.

References

Horvitz, D. G. & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260), 663–685. doi:10.1080/01621459.1952.10483446

Cochran, W. G. (1977). *Sampling Techniques* (3rd ed.). Wiley.

Sarndal, C.-E., Swensson, B. & Wretman, J. (1992). *Model Assisted Survey Sampling*. Springer. doi:10.1007/9781461243786

Lumley, T. (2010). *Complex Surveys: A Guide to Analysis Using R*. Wiley. doi:10.1002/9780470580066

See Also

[describe_survey_design](#), [estimate_survey](#), [svydesign](#), [as_survey](#)

Examples

```
data <- generate_example_data(n_upm = 30)

# Stratified multistage design: weight = w_k, strata = h, cluster = alpha
design <- as_survey_design_tbl(
  data,
  weight = "weight",
  strata = "strata",
  cluster = "upm"
)

# Cluster design without stratification (H = 1)
design2 <- as_survey_design_tbl(
  data,
  weight = "weight",
  cluster = "upm"
)

# Weights-only design (SRS approximation, d_k = w_k)
design3 <- as_survey_design_tbl(
  data,
  weight = "weight"
)
```

 describe_survey_design

Describe a complex survey design

Description

Returns a one-row tibble of design diagnostics: sample size n , number of strata H , total number of PSUs $\sum_h \alpha_h$, and summary statistics of the adjusted weights w_k including their coefficient of variation $CV(w) = s_w/\bar{w}$, which is an indicator of weight variability and its effect on variance inflation.

Usage

```
describe_survey_design(design)
```

Arguments

`design` A `tbl_svy` object created with [as_survey_design_tbl](#).

Value

A tibble with one row and the following columns:

`n_obs` Total sample size $n = |s|$.

`n_strata` Number of strata H (NA if no stratification variable was supplied).

`n_clusters` Total number of PSUs $\sum_{h=1}^H \alpha_h$ (NA if no cluster variable was supplied).

`weight_min` $\min_{k \in s} w_k$.

`weight_max` $\max_{k \in s} w_k$.

`weight_mean` Unweighted mean of the adjusted weights $\bar{w} = \hat{N}_w/n$, where $\hat{N}_w = \sum_{k \in s} w_k$.

`weight_cv` Coefficient of variation of the weights $CV(w) = s_w/\bar{w}$. Large values (> 0.5) indicate high weight variability, which can substantially inflate the design effect \widehat{DEFF} .

See Also

[as_survey_design_tbl](#), [estimate_survey](#)

Examples

```
data <- generate_example_data(30)
design <- as_survey_design_tbl(
  data,
  weight = "weight",
  strata = "strata",
  cluster = "upm"
)
describe_survey_design(design)
```

Description

Computes design-based point estimates and their associated uncertainty measures for complex survey data. The function operates on objects of class `tbl_svy` and implements five estimators grounded in the Horvitz–Thompson (HT) framework: weighted mean \bar{y}_w , population total \hat{Y}_{HT} , proportion \hat{p}_k , ratio \hat{R} , and quantile \hat{q}_p .

Usage

```
estimate_survey(
  design,
  variable = NULL,
  estimator = c("mean", "total", "prop", "ratio", "quantile"),
  by = NULL,
  na_rm = TRUE,
  conf_level = 0.95,
  numerator = NULL,
  denominator = NULL,
  ratio_num_level = NULL,
  ratio_den_level = NULL,
  probs = c(0.25, 0.5, 0.75)
)
```

Arguments

<code>design</code>	A <code>tbl_svy</code> object created with <code>as_survey_design_tbl()</code> , encoding weights w_k , strata h and primary sampling units α .
<code>variable</code>	Character string with the name of the target variable y_k . Required for "mean", "total", "prop" and "quantile". Ignored when estimator = "ratio".
<code>estimator</code>	Character string specifying the estimator. One of "mean", "total", "prop", "ratio", or "quantile".
<code>by</code>	Optional character vector naming domain variables that define subpopulations $U_d \subset U$.
<code>na_rm</code>	Logical. Whether to remove NA values before estimation. Default is TRUE.
<code>conf_level</code>	Numeric in $(0, 1)$. Confidence level for interval estimation. Default is 0.95.
<code>numerator</code>	Character string with the name of the numerator variable y_k in ratio estimation.
<code>denominator</code>	Character string with the name of the denominator variable x_k in ratio estimation.
<code>ratio_num_level</code>	Character. Category of numerator used as \hat{N}_{num} when the numerator variable is categorical.

ratio_den_level

Character. Category of denominator used as \hat{N}_{den} when the denominator variable is categorical.

probs

Numeric vector of probabilities in $(0, 1)$ for quantile estimation.

Details

Let $U = \{1, \dots, N\}$ be the finite population and $s \subset U$ the selected sample. For each unit $k \in s$, $d_k = 1/\pi_k$ is the **basic design weight** (inverse of the first-order inclusion probability π_k), and w_k is the **adjusted weight** after non-response or calibration corrections.

All estimators account for stratification, clustering and unequal weights. Variance is estimated by Taylor linearization via the **survey** and **srvyr** packages, and the design effect $\widehat{\text{DEFF}} = \hat{V}_p(\hat{\theta})/\hat{V}_{\text{SRS}}(\hat{\theta})$ is reported for mean, total and proportion estimators. Domain estimation (argument by) is supported for all estimators; variance is computed over the full sample s to avoid subsetting bias.

Mean

The weighted mean is the HT ratio estimator:

$$\bar{y}_w = \frac{\hat{Y}_w}{\hat{N}_w} = \frac{\sum_{k \in s} w_k y_k}{\sum_{k \in s} w_k}$$

where $\hat{N}_w = \sum_{k \in s} w_k$ is the estimated population size. Its variance is estimated by Taylor linearization (Sarndal et al., 1992):

$$\hat{V}_p(\bar{y}_w) = \frac{1}{\hat{N}_w^2} \hat{V}_p(\hat{Y}_w).$$

Total

The HT estimator of the population total is (Horvitz & Thompson, 1952):

$$\hat{Y}_{HT} = \sum_{k \in s} d_k y_k.$$

With adjusted weights and a stratified multistage design (H strata, α_h PSUs per stratum h , $n_{h\alpha}$ observations per PSU):

$$\hat{Y}_w = \sum_{h=1}^H \sum_{\alpha=1}^{\alpha_h} \sum_{k=1}^{n_{h\alpha}} \omega_{h\alpha k} y_{h\alpha k}, \quad \hat{V}_p(\hat{Y}_w) = \sum_{h=1}^H \hat{V}_{p,h}(\hat{Y}_{w,h}).$$

Proportion

For a binary variable $y_k \in \{0, 1\}$, the estimated proportion is:

$$\hat{p} = \frac{\hat{N}_1}{\hat{N}_w} = \frac{\sum_h \sum_{\alpha} \sum_k \omega_{h\alpha k} I(y_k = 1)}{\sum_h \sum_{\alpha} \sum_k \omega_{h\alpha k}}.$$

For a categorical variable with categories \mathcal{K} , the proportion of category c is $\hat{p}_c = \hat{N}_c / \hat{N}_w$, where \hat{N}_c is the weighted count of units with $y_k = c$. Variance is approximated by Taylor linearization (Heeringa et al., 2017):

$$\hat{V}_p(\hat{p}) \approx \frac{\hat{V}_p(\hat{N}_1) + \hat{p}^2 \hat{V}_p(\hat{N}_w) - 2\hat{p} \widehat{\text{cov}}(\hat{N}_1, \hat{N}_w)}{\hat{N}_w^2}.$$

Ratio

The ratio estimator of two population totals is (Cochran, 1977):

$$\hat{R} = \frac{\hat{Y}_w}{\hat{X}_w} = \frac{\sum_{k \in s} w_k y_k}{\sum_{k \in s} w_k x_k}.$$

Variance is estimated by first-order Taylor linearization:

$$\hat{V}_p(\hat{R}) \approx \frac{1}{\hat{X}_w^2} \hat{V}_p(\hat{Y}_w - \hat{R} \hat{X}_w).$$

When variables are categorical, indicator variables are constructed for the specified levels before calling survey::svyratio(). The deff column is NA for ratio estimates.

Quantile

Quantiles are derived from the weighted empirical CDF (Woodruff, 1952):

$$\hat{F}_w(t) = \frac{\sum_{k \in s} w_k I(y_k \leq t)}{\sum_{k \in s} w_k}, \quad \hat{q}_p = \inf\{t : \hat{F}_w(t) \geq p\}.$$

Confidence intervals use the Woodruff linearization method, which maps the problem to the cumulative proportion scale:

$$IC_p[\hat{q}_p] = \left\{ t : \hat{F}_w(t) \in \left[p \pm t_{1-\alpha/2, gl} ee(\hat{F}_w(t)) \right] \right\}.$$

The deff column is NA for quantile estimates.

Design effect

For estimators that support it, the output column deff contains (Kish, 1965):

$$\widehat{\text{DEFF}} = \frac{\hat{V}_p(\hat{\theta})}{\hat{V}_{\text{SRS}}(\hat{\theta})}.$$

Domain estimation

When by is supplied, estimation is performed within each subpopulation U_d defined by the combination of domain variable levels. The domain mean estimator is:

$$\bar{y}_{w,d} = \frac{\sum_{k \in s} w_k y_k I(k \in U_d)}{\sum_{k \in s} w_k I(k \in U_d)} = \frac{\hat{Y}_{w,d}}{\hat{N}_{w,d}}.$$

Variance is computed over the full sample s , preserving the design structure and avoiding subsetting bias (Lumley, 2010).

Confidence intervals

Intervals are constructed under asymptotic normality as:

$$\hat{\theta} \pm z_{1-\alpha/2} ee(\hat{\theta}), \quad ee(\hat{\theta}) = \sqrt{\hat{V}_p(\hat{\theta})}.$$

Value

A tibble with the following columns, plus one column per domain variable (if by is supplied):

variable Name of the target variable y_k .

estimator Type of estimator ("mean", "total", etc.).

estimate Point estimate $\hat{\theta}$.

se Standard error $ee(\hat{\theta}) = \sqrt{\hat{V}_p(\hat{\theta})}$.

cv Coefficient of variation $CV = ee(\hat{\theta})/\hat{\theta}$.

deff Design effect \widehat{DEF} (mean, total, proportion only; NA for ratio and quantile).

lci Lower confidence bound.

uci Upper confidence bound.

quality Precision label derived from the CV: "Very high precision" ($CV < 5\%$), "High precision" (5%–10%), "Acceptable precision" (10%–20%), "Use with caution" (20%–30%), or "Low precision" ($\geq 30\%$).

References

- Horvitz, D. G. & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260), 663–685. doi:10.1080/01621459.1952.10483446
- Cochran, W. G. (1977). *Sampling Techniques* (3rd ed.). Wiley.
- Woodruff, R. S. (1952). Confidence intervals for medians and other position measures. *Journal of the American Statistical Association*, 47(260), 635–646. doi:10.1080/01621459.1952.10483443
- Kish, L. (1965). *Survey Sampling*. Wiley.
- Sarndal, C.-E., Swensson, B. & Wretman, J. (1992). *Model Assisted Survey Sampling*. Springer. doi:10.1007/9781461243786
- Lumley, T. (2010). *Complex Surveys: A Guide to Analysis Using R*. Wiley. doi:10.1002/9780470580066
- Heeringa, S. G., West, B. T. & Berglund, P. A. (2017). *Applied Survey Data Analysis* (2nd ed.). CRC Press. doi:10.1201/9781315153278

See Also

[survey_mean](#), [survey_total](#), [svyratio](#), [svyquantile](#), [as_survey_design_tbl](#)

Examples

```

# -----
# Generate example data and build a stratified multistage
# design: weight = w_k, strata = h, cluster = alpha
# -----
data <- generate_example_data(n_upm = 30, seed = 123)

design <- srvyr::as_survey_design(
  data,
  ids    = upm,
  strata = strata,
  weights = weight
)

# -----
# Mean --  $\bar{y}_w$  with DEFF and 95 % CI
# -----
estimate_survey(
  design,
  variable = "ingreso_pc",
  estimator = "mean"
)

# Domain mean  $\bar{y}_{w,d}$  by region
estimate_survey(
  design,
  variable = "ingreso_pc",
  estimator = "mean",
  by       = "region"
)

# -----
# Total --  $\hat{Y}_{HT}$  with DEFF
# -----
estimate_survey(
  design,
  variable = "ingreso_pc",
  estimator = "total"
)

# -----
# Proportion --  $\hat{p} = \hat{N}_1 / \hat{N}_w$ 
# -----
estimate_survey(
  design,
  variable = "pobre",
  estimator = "prop"
)

# Multinomial proportion  $\hat{p}_c = \hat{N}_c / \hat{N}_w$ 
estimate_survey(
  design,

```

```

    variable = "empleo",
    estimator = "prop"
  )

# -----
# Ratio --  $\hat{R} = \hat{Y}_w / \hat{X}_w$ 
# -----
estimate_survey(
  design,
  estimator = "ratio",
  numerator = "ingreso_pc",
  denominator = "gasto_pc"
)

# Ratio with domains
estimate_survey(
  design,
  estimator = "ratio",
  numerator = "ingreso_pc",
  denominator = "gasto_pc",
  by = "region"
)

# Categorical / Categorical:  $\hat{N}_{\text{Formal}} / \hat{N}_{\text{Informal}}$ 
estimate_survey(
  design,
  estimator = "ratio",
  numerator = "empleo",
  denominator = "empleo",
  ratio_num_level = "Formal",
  ratio_den_level = "Informal"
)

# -----
# Quantile --  $\hat{q}_p$  via Woodruff linearization
# -----
estimate_survey(
  design,
  variable = "ingreso_pc",
  estimator = "quantile",
  probs = c(0.25, 0.5, 0.75)
)

```

format_results_table *Format survey estimation results*

Description

Formats survey estimation outputs by ensuring required metrics are present (CV, confidence intervals) and applying rounding.

Usage

```
format_results_table(res, digits = 2)
```

Arguments

res	A data.frame or tibble with estimation results.
digits	Integer. Number of decimal places.

Value

A tibble with formatted results.

Examples

```
res <- data.frame(estimate = c(0.45, 0.30), se = c(0.02, 0.015))
format_results_table(res)
```

generate_example_data *Generate simulated complex survey data with hierarchical structure*

Description

Generates a synthetic dataset representing a complex survey design with a three-level hierarchical structure: primary sampling units (PSUs), households, and individuals. The function ensures internal consistency of weights, cluster sizes, and derived variables, making it suitable for testing survey estimators, calibration methods, and small area estimation workflows.

Usage

```
generate_example_data(n_upm = 200, seed = 123)
```

Arguments

n_upm	Integer. Number of primary sampling units (PSUs).
seed	Integer. Random seed for reproducibility.

Details**Sampling structure**

The simulated data follows a hierarchical design:

1. **PSU (UPM)**: each PSU belongs to a stratum and contains a random number of households (between 5 and 15).
2. **Households**: each household has a sampling weight and a fixed area (urban/rural).
3. **Individuals**: observations are generated at the individual level.

Weights

Sampling weights are defined at the household level and are constant for all individuals within the same household:

$$w_{hi} = w_h$$

Income generation

Household income is generated with hierarchical random effects:

$$Y_h \sim \text{Gamma}(\alpha, \beta \cdot \exp(u_j + v_h))$$

where:

- u_j is the PSU-level effect
- v_h is the household-level effect

Per capita income is then defined as:

$$y_{hi} = \frac{Y_h}{N_h}$$

where N_h is the household size.

Education and income correlation

Education is assigned at the individual level conditional on age. A household-level education effect is derived from the maximum education level within the household and used to adjust household income:

$$Y_h^* = Y_h \cdot f(E_h)$$

where $f(E_h)$ is a multiplicative factor depending on household education.

Consistency constraints

The simulation enforces:

- All individuals in a household share the same area
- Individuals younger than 5 years have no education (NA)
- Per capita income is constant within households
- Each PSU contains at most 15 households

Variables

The dataset includes:

- Design variables: strata, upm, hogar_id, persona_id, weight
- Domains: region, sexo, area
- Demographics: edad, educacion, empleo
- Welfare indicators: ingreso_pc, gasto_pc, pobre
- Auxiliary variable with missingness: ingreso2

Value

A tibble at the individual level with hierarchical identifiers and survey variables.

Examples

```
data <- generate_example_data(n_upm = 50)
validate_simulation(data)
# -----
# Example: build survey design
# -----
# design <- as_survey_design_tbl(
#   data,
#   weight = "weight",
#   strata = "strata",
#   cluster = "upm"
# )
```

i18n_load

Load a language dictionary from JSON

Description

Reads a JSON translation file from the package's `inst/` directory and returns it as a nested list suitable for use with [i18n_t](#).

Usage

```
i18n_load(lang, path = "app/www/i18n")
```

Arguments

lang	Character. Language code, e.g. "en" (English) or "es" (Spanish).
path	Character. Path inside <code>inst/</code> to the directory that contains the JSON translation files. Defaults to "app/www/i18n".

Value

A named list with the translation dictionary.

See Also

[i18n_t](#)

Examples

```
dict <- i18n_load("en")
i18n_t(dict, "mod_datos.title")
```

i18n_t	<i>Translate a key from a language dictionary</i>
--------	---

Description

Resolves a dot-separated key path in a nested translation list returned by `i18n_load`. Returns the key itself when the path is not found, so the interface degrades gracefully on missing translations.

Usage

```
i18n_t(dict, key)
```

Arguments

dict	List. A translation dictionary returned by <code>i18n_load()</code> .
key	Character. A dot-separated path to the translation string, e.g. <code>"mod_datos.title"</code> .

Value

A character string with the translated text, or key when the path does not exist.

See Also

[i18n_load](#)

Examples

```
dict <- i18n_load("en")
i18n_t(dict, "mod_datos.title")
i18n_t(dict, "nonexistent.key") # returns "nonexistent.key"
```

mutate_survey_data	<i>Mutate survey data using declarative definitions</i>
--------------------	---

Description

Adds or transforms variables in survey microdata using a named list of formulas. Each definition must be a one-sided formula of the form `~ expression`, evaluated within the data context.

Usage

```
mutate_survey_data(data, definitions, overwrite = TRUE)
```

Arguments

data	A tibble with survey microdata.
definitions	Named list of formulas defining new variables.
overwrite	Logical. If FALSE, prevents overwriting existing variables.

Value

A tibble with new variables added or modified.

Examples

```
data <- tibble::tibble(  
  ingreso = c(100, 200),  
  miembros = c(2, 4)  
)  
  
mutate_survey_data(  
  data,  
  list(  
    ingreso_pc = ~ ingreso / miembros  
  )  
)
```

plot_results_bar *Plot survey estimates as bar chart*

Description

Plot survey estimates as bar chart

Usage

```
plot_results_bar(results)
```

Arguments

results	Tibble returned by estimate_survey().
---------	---------------------------------------

Value

A ggplot object.

Examples

```
res <- data.frame(
  region = c("North", "South", "East"),
  estimate = c(0.45, 0.30, 0.55),
  lci = c(0.40, 0.25, 0.50),
  uci = c(0.50, 0.35, 0.60),
  estimator = "prop"
)
plot_results_bar(res)
```

read_survey_data	<i>Read survey microdata from multiple file formats</i>
------------------	---

Description

Reads microdata commonly used in official statistics and survey analysis from a local file path. The function returns a tibble and attaches source metadata as attributes, facilitating traceability and reproducibility.

Usage

```
read_survey_data(
  path,
  col_types = NULL,
  guess_max = 10000,
  encoding = "UTF-8",
  sheet = 1
)
```

Arguments

path	Character. File path to the dataset.
col_types	Optional. Passed to <code>readr::read_csv()</code> / <code>read_tsv()</code> .
guess_max	Integer. Maximum rows used for type guessing (CSV/TSV only).
encoding	Character. File encoding (CSV/TSV only). Default "UTF-8".
sheet	Integer or character. Sheet to read for Excel files. Default 1.

Details

Supported formats:

- .csv — comma-separated values (`readr`)
- .tsv, .txt — tab-separated values (`readr`)
- .xlsx — Excel 2007+ (`readxl`)
- .xls — Excel 97-2003 (`readxl`)

- .sav — SPSS (haven)
- .por — SPSS portable (haven)
- .dta — Stata (haven)
- .sas7bdat — SAS data (haven)
- .xpt — SAS transport (haven)
- .rds — R serialized object (base R)

Value

A tibble with attributes:

- source_path: normalized file path
- source_format: file extension
- n_rows: number of rows
- n_cols: number of columns

Examples

```
csv_path <- system.file("extdata", "simulated_survey_data.csv", package = "complexr")
df <- read_survey_data(csv_path)
attr(df, "source_format")
```

```
rds_path <- system.file("extdata", "simulated_survey_data.rds", package = "complexr")
df2 <- read_survey_data(rds_path)
attr(df2, "source_format")
```

run_app

Run complexr application

Description

Launches the Shiny app bundled within the package in the default browser.

Usage

```
run_app()
```

Value

Called for its side effect: opens the Shiny application. Returns invisibly when the app is stopped.

Examples

```
if (interactive()) {
  run_app()
}
```

validate_simulation *Validate simulated survey data*

Description

Performs a set of structural and consistency checks on a dataset generated by `generate_example_data()`. The function verifies household-level invariants (area, weights, per capita income), demographic rules, and PSU size constraints.

Usage

```
validate_simulation(data, strict = FALSE)
```

Arguments

`data` A tibble or data.frame at individual level.
`strict` Logical. If TRUE, stops execution when any validation fails.

Value

A tibble with validation results (one row per check).

Examples

```
data <- generate_example_data(n_upm = 20, seed = 123)  
validate_simulation(data)
```

Index

app_server, 2
app_ui, 3
as_survey, 5
as_survey_design_tbl, 3, 6, 10

describe_survey_design, 5, 6

estimate_survey, 4–6, 7

format_results_table, 12

generate_example_data, 13

i18n_load, 15, 16
i18n_t, 15, 16

mutate_survey_data, 16

plot_results_bar, 17

read_survey_data, 18
run_app, 19

survey_mean, 10
survey_total, 10
svydesign, 5
svyquantile, 10
svyratio, 10

validate_simulation, 20