Package 'svytest'

October 30, 2025

Type Package

Title Survey Weight Diagnostic Tests

Version 1.1.0

Description Provides diagnostic tests for assessing the informativeness of survey weights in regression models. Implements difference-in-coefficients tests (Hausman 1978 <doi:10.2307/1913827>; Pfeffermann 1993 <doi:10.2307/1403631>), weight-association tests (DuMouchel and Duncan 1983 <doi:10.2307/2288185>; Pfeffermann and Sverchkov 1999 <https://www.jstor.org/stable/25051118>; Pfeffermann and Sverchkov 2003 <ISBN:9780470845672>; Wu and Fuller 2005 <https://www.jstor.org/stable/27590461>), estimating equations tests (Pfeffermann and Sverchkov 2003 <ISBN:9780470845672>), and non-parametric permutation tests. Includes simulation utilities replicating Wang et al. (2023 <doi:10.1111/insr.12509>) and extensions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports Rcpp, survey

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, MASS, rmarkdown, sampling, dplyr, tidyr, tibble, future.apply, broom, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Corbin Lubianski [aut, cre, cph] (ORCID: <https://orcid.org/0009-0009-0547-3249>)

Maintainer Corbin Lubianski <cnlubianski@yahoo.com>

Repository CRAN

Date/Publication 2025-10-30 20:20:02 UTC

Contents

`diff_in_coef_test` *Difference-in-Coefficients Test for Survey Weights*

Description

Implements the Hausman-Pfeffermann Difference-in-Coefficients test to assess whether survey weights significantly affect regression estimates.

Usage

```
diff_in_coef_test(
  model,
  lower.tail = FALSE,
  var_equal = TRUE,
  robust_type = c("HC0", "HC1", "HC2", "HC3"),
  coef_subset = NULL,
  na.action = stats::na.omit
)

## S3 method for class 'diff_in_coef_test'
print(x, ...)

## S3 method for class 'diff_in_coef_test'
summary(object, ...)

## S3 method for class 'diff_in_coef_test'
tidy(x, ...)

## S3 method for class 'diff_in_coef_test'
glance(x, ...)
```

Arguments

`model`	An object of class `svyglm`.
`lower.tail`	Logical; passed to `pchisq()`.

var_equal	Logical; assume equal residual variance between models. If FALSE, a heteroskedasticity-robust variance estimator is used.
robust_type	Character; type of heteroskedasticity-robust variance estimator to use if var_equal = FALSE. Options are "HC0", "HC1", "HC2", "HC3" as used in 'sandwich' package.
coef_subset	Character vector of coefficient names to include in the test. Defaults to all coefficients.
na.action	Function to handle missing data before fitting the test.
x	An object of class diff_in_coef_test
...	Additional arguments passed to methods
object	An object of class diff_in_coef_test

Details

Let X denote the design matrix and y the response vector. Define the unweighted OLS estimator

$$\hat{\beta}_U = (X^\top X)^{-1} X^\top y,$$

and the survey-weighted estimator

$$\hat{\beta}_W = (X^\top W X)^{-1} X^\top W y,$$

where $W = \mathrm{diag}(w_1, \ldots, w_n)$ is the diagonal matrix of survey weights.

The test statistic is based on the difference

$$d = \hat{\beta}_W - \hat{\beta}_U.$$

Under the null hypothesis that weights are not informative, d has mean zero and variance V_d. The test statistic is

$$T = d^\top V_d^{-1} d,$$

which is asymptotically χ^2_p distributed with p equal to the number of coefficients tested.

If var_equal = TRUE, V_d is estimated assuming equal residual variance across weighted and unweighted models. If var_equal = FALSE, a heteroskedasticity-robust estimator (e.g. HC0–HC3) is used.

This test is a survey-weighted adaptation of the Hausman specification test (Hausman, 1978), as proposed by Pfeffermann (1993).

Value

An object of class "diff_in_coef_test" containing:

statistic	Chi-squared test statistic
parameter	Degrees of freedom
p.value	P-value for the test
betas_unweighted	Unweighted coefficient estimates
betas_weighted	Weighted coefficient estimates
vcov_diff	Estimated variance-covariance matrix of coefficient differences
diff_betas	Vector of coefficient differences
call	Function call

References

Hausman, J. A. (1978). Specification Tests in Econometrics. *Econometrica*, 46(6), 1251-1271. doi:10.2307/1913827

Pfeffermann, D. (1993). The Role of Sampling Weights When Modeling Survey Data. *International Statistical Review*, 61(2), 317-337. doi:10.2307/1403631

See Also

svytestCE for the curated Consumer Expenditure dataset included in this package, which can be used to demonstrate the Difference-in-Coefficients test.

@importFrom survey svyglm

Examples

```
# Load in survey package (required) and load in example data
library(survey)
data(api, package = "survey")

# Create a survey design and fit a weighted regression model
des <- svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat)
fit <- svyglm(api00 ~ ell + meals, design = des)

# Run difference-in-coefficients diagnostic test versions with different variance assumptions
# and reports Chi-Squared statistic, df, and p-value
summary(diff_in_coef_test(fit, var_equal = TRUE))
summary(diff_in_coef_test(fit, var_equal = FALSE, robust_type = "HC3"))
```

`estim_eq_test` *Estimating Equations Test for Informative Sampling (Linear Case)*

Description

Implements the Pfeffermann-Sverchkov estimating equations test for informativeness of survey weights in the linear regression case (Gaussian with identity link). The test compares unweighted estimating equations to adjusted-weight equations using $q_i = w_i / E_s(w_i \mid x_i)$.

Usage

```
estim_eq_test(
  model,
  coef_subset = NULL,
  q_method = c("linear", "log"),
  stabilize = TRUE,
  na.action = stats::na.omit
)
```

```
## S3 method for class 'estim_eq_test'
print(x, ...)

## S3 method for class 'estim_eq_test'
summary(object, ...)

## S3 method for class 'estim_eq_test'
tidy(x, ...)

## S3 method for class 'estim_eq_test'
glance(x, ...)
```

Arguments

model	An object of class svyglm with family = gaussian(identity).
coef_subset	Optional character vector of coefficient names to include in the test. Defaults to all coefficients in the model matrix.
q_method	Method for estimating $E_s(w \mid x)$: "linear" (default, OLS regression of w on X) or "log" (regress log(w) on X, then exponentiate).
stabilize	Logical; if TRUE (default) clips extreme q values.
na.action	Function to handle missing data.
x	An object of class estim_eq_test
...	Additional arguments passed to methods
object	An object of class estim_eq_test

Details

For linear regression, the per-observation score is $u_i = x_i(y_i - x_i^\top \hat{\beta}_{\text{unw}})$ at the unweighted OLS estimate. The test statistic is based on $R_i = (1 - q_i)u_i$, where $q_i = w_i / E_s(w_i \mid x_i)$. The Hotelling F statistic is $F = \frac{n-p}{p} \bar{R}^\top S^{-1} \bar{R}$, with df1 = p, df2 = n - p.

Value

An object of class "estim_eq_test" containing:

statistic	Hotelling F statistic
p.value	p-value under F distribution
df1	Numerator df (# tested coefficients)
df2	Denominator df (n - p)
Rbar	Mean estimating-equation contrast vector
S	Sample covariance of R
terms	Names of tested coefficients
n	Sample size
call	Matched call
method	Description string

References

Pfeffermann, D., & Sverchkov, M. Y. (2003). Fitting generalized linear models under informative sampling. In R. L. Chambers & C. J. Skinner (Eds.), *Analysis of Survey Data* (Ch. 12). Wiley.

See Also

diff_in_coef_test, wa_test, perm_test

Examples

```
# Load in survey package (required) and load in example data
library(survey)
data("svytestCE", package = "svytest")

# Create a survey design and fit a weighted regression model
des <- svydesign(ids = ~1, weights = ~FINLWT21, data = svytestCE)
fit <- svyglm(TOTEXPCQ ~ ROOMSQ + BATHRMQ + BEDROOMQ + FAM_SIZE + AGE, design = des)

# Run estimating equations diagnostic test; reports F statistic, df's, and p-value
results <- estim_eq_test(fit, q_method = "linear")
print(results)
```

---

perm_test *Permutation test for weight informativeness in survey regression*

Description

Non-parametric test that permutes survey weights (optionally within blocks) to generate the null distribution of a chosen statistic. Supports fast closed-form WLS (linear case) via C++ and a pure R engine.

Usage

```
perm_test(
  model,
  stat = c("pred_mean", "coef_mahal"),
  B = 1000,
  coef_subset = NULL,
  block = NULL,
  normalize = TRUE,
  engine = c("C++", "R"),
  custom_fun = NULL,
  na.action = stats::na.omit
)

## S3 method for class 'perm_test'
print(x, ...)
```

```
## S3 method for class 'perm_test'
summary(object, ...)

## S3 method for class 'perm_test'
tidy(x, ...)

## S3 method for class 'perm_test'
glance(x, ...)
```

Arguments

model	An object of class svyglm (currently supports Gaussian family best).
stat	Statistic to use. Options: • "pred_mean": Compares the mean predicted outcome under weighted vs. unweighted regression. Simple, interpretable, and directly tied to differences in fitted population means. Sensitive to shifts in overall prediction levels caused by informative weights. • "coef_mahal": Computes the Mahalanobis distance between the weighted and unweighted coefficient vectors, using the unweighted precision matrix ($X'X$) as the metric. Captures joint shifts in regression coefficients, not just mean predictions. More powerful when informativeness manifests as changes in slopes or multiple coefficients simultaneously.
B	Number of permutations (e.g., 1000).
coef_subset	Optional character vector of coefficient names to include.
block	Optional factor for blockwise permutations (e.g., strata), permute within levels.
normalize	Logical; if TRUE (default), normalize weights to have mean 1.
engine	"C++" for fast WLS or "R" for pure R loop.
custom_fun	Optional function(model, X, y, wts) -> scalar statistic (overrides stat).
na.action	Function to handle missing data.
x	An object of class perm_test
...	Additional arguments passed to methods
object	An object of class perm_test

Details

This procedure implements a non-parametric randomization test for the informativeness of survey weights. The null hypothesis is that, conditional on the covariates X, the survey weights w are *non-informative* with respect to the outcome y. Under this null, permuting the weights across observations should not change the distribution of any statistic that measures the effect of weighting.

The algorithm is:

1. Fit the unweighted regression
$$\hat{\beta}_U = (X^\top X)^{-1} X^\top y$$

and the weighted regression

$$\hat{\beta}_W = (X^\top W X)^{-1} X^\top W y,$$

where $W = \mathrm{diag}(w)$.

2. Compute the observed test statistic T_{obs}:
   - For `"pred_mean"`: the difference in mean predicted outcomes between weighted and unweighted fits.
   - For `"coef_mahal"`: the Mahalanobis distance

     $$T = (\hat{\beta}_W - \hat{\beta}_U)^\top (X^\top X)(\hat{\beta}_W - \hat{\beta}_U),$$

     using the unweighted precision matrix as the metric.
   - For a user-supplied `custom_fun`, any scalar function of (X, y, w).
3. Generate the null distribution by permuting the weights:

   $$w^{*(b)} = P_b w, \quad b = 1, \ldots, B,$$

   where each P_b is a permutation matrix. If a `block` factor is supplied, permutations are restricted within block levels.
4. Recompute the test statistic $T^{*(b)}$ for each permuted weight vector. The empirical distribution of $T^{*(b)}$ represents the null distribution under non-informative weights.
5. The two-sided permutation p-value is

   $$p = \frac{1 + \sum_{b=1}^{B} I\{|T^{*(b)} - T_0| \geq |T_{\mathrm{obs}} - T_0|\}}{B + 1},$$

   where T_0 is the baseline statistic under equal weights.

Intuitively, if the weights are informative, the observed statistic will lie in the tails of the permutation distribution, leading to a small p-value. If the weights are non-informative, shuffling them destroys any spurious association with the outcome, and the observed statistic is typical of the permutation distribution.

Value

An object of class `"perm_test"` with fields:

`stat_obs`	Observed statistic with actual weights
`stat_null`	Baseline statistic under equal weights (for centering)
`perm_stats`	Vector of permutation statistics
`p.value`	Permutation p-value (two-sided, centered at baseline)
`effect`	Observed minus median of permutation stats
`stat`	Statistic name
`B`	Number of permutations
`call`	Matched call
`method`	Description string

Examples

```
# Load in survey package (required) and load in example data
library(survey)
data(api, package = "survey")

# Create a survey design and fit a weighted regression model
des <- svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat)
fit <- svyglm(api00 ~ ell + meals, design = des)

# Run permutation diagnostic test; reports permutation statistics with p-value
results <- perm_test(fit, stat = "pred_mean", B = 1000, engine = "R")
print(results)
```

---

plot.perm_test *Plot method for permutation test objects*

Description

Produces a histogram of the permutation distribution with a vertical line indicating the observed statistic.

Usage

```
## S3 method for class 'perm_test'
plot(x, bins = 30, col = "lightgray", line_col = "red", ...)
```

Arguments

x	An object of class "perm_test".
bins	Number of histogram bins (default 30).
col	Color for histogram bars (default "lightgray").
line_col	Color for observed statistic line (default "red").
...	Additional arguments passed to hist().

Value

A base R side effect plot. Function returns NULL invisibly.

run_all_diagnostic_tests *Run All Diagnostic Tests for Informative Weights*

Description

This function runs all implemented diagnostic tests: - wa_test() with types "DD", "PS1", "PS1q", "PS2", "PS2q", "WF" - diff_in_coef_test() - estim_eq_test() - perm_test() with stats "pred_mean" and "coef_mahal"

Usage

```
run_all_diagnostic_tests(model, alpha = 0.05, B = 1000)

## S3 method for class 'run_all_diagnostic_tests'
print(x, ...)
```

Arguments

model	A fitted svyglm object.
alpha	Critical value for rejection (default 0.05).
B	Number of permutations for permutation tests (default 1000).
x	An object of class run_all_diagnostic_tests
...	Additional arguments passed to methods

Value

A list with:

results	Data frame of test names, statistics, p-values, reject indicator
recommendation	Character string with suggested action
raw	List of raw test outputs, including permutation test objects

See Also

wa_test, diff_in_coef_test, estim_eq_test, perm_test

Examples

```
# Load in survey package (required) and load in example data
library(survey)
data(api, package = "survey")

# Create a survey design and fit a weighted regression model
des <- svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat)
fit <- svyglm(api00 ~ ell + meals, design = des)
```

```
# Run all diagnostic tests and return a list of statistics, including a recommendation
results <- run_all_diagnostic_tests(fit)
print(results)
```

---

svytestCE *Subset of 2015 Consumer Expenditure (CE) Dataset*

Description

A curated subset of rows and columns from the Consumer Expenditure (CE) dataset that is provided in the [rpms](https://CRAN.R-project.org/package=rpms) package by Daniell Toth. This example dataset is designed for demonstration purposes within this package. Please reframe from using this dataset for inferential purposes.

Usage

```
svytestCE
```

Format

A data frame with _n_ rows and _m_ variables:

NEWID Consumer unit identifying variable, constructed using the first seven digits of a unique identifier.

CID Cluster Identifier for all clusters (constructed using PSU, REGION, STATE, and POPSIZE).

QINTRVMO Month for which the data were collected.

FINLWT21 Final sample weight used to make population inferences.

STATE State FIPS code indicating the location of the consumer unit.

REGION Region code: 1 = Northeast, 2 = Midwest, 3 = South, 4 = West.

BLS_URBN Indicator of urban (1) versus rural (2) residence status.

POPSIZE Population size class of the PSU, ranging from 1 (largest) to 5 (smallest).

CUTENURE Housing tenure: 1 = Owned with mortgage; 2 = Owned without mortgage; 3 = Owned (mortgage not reported); 4 = Rented; 5 = Occupied without cash rent; 6 = Student housing.

ROOMSQ Number of rooms (including finished living areas but excluding bathrooms).

BATHRMQ Number of bathrooms in the consumer unit.

BEDROOMQ Number of bedrooms in the consumer unit.

VEHQ Number of owned vehicles.

FAM_TYPE Household type based on the relationship of members to the reference person; for example, 1 = Married Couple only, 2 = Married Couple with children (oldest < 6 years), 3 = Married Couple with children (oldest 6-17 years), etc.

FAM_SIZE Number of members in the consumer unit (family size).

PERSLT18 Count of persons less than 18 years old in the consumer unit.

PERSOT64 Count of persons older than 64 years in the consumer unit.

NO_EARNR Number of earners in the consumer unit.

AGE Age of the primary earner.

EDUCA Education level of the primary earner, coded as 1 = None, 2 = 1st-8th Grade, 3 = Some high school, 4 = High school, 5 = Some college, 6 = AA degree, 7 = Bachelor's degree, 8 = Advanced degree.

SEX Gender of the primary earner (F = Female, M = Male).

MARITAL Marital status of the primary earner (1 = Married, 2 = Widowed, 3 = Divorced, 4 = Separated, 5 = Never Married).

MEMBRACE Race of the primary earner (e.g., 1 = White, 2 = Black, 3 = Native American, 4 = Asian, 5 = Pacific Islander, 6 = Multi-race).

HORIGIN Indicator of Hispanic, Latino, or Spanish origin (Y for yes, N for no).

ARM_FORC Indicator if the primary earner is a member of the armed forces (Y/N).

IN_COLL Current college enrollment status for the primary earner (Full for full time, Part for part time, No for not enrolled).

EARNTYPE Type of employment for the primary earner: 1 = Full time all year, 2 = Part time all year, 3 = Full time part-year, 4 = Part time part-year.

OCCUCODE Occupational code representing the primary job of the earner.

INCOMEY Type of employment: coded as 1 = Employee of a private company, 2 = Federal government employee, 3 = State government employee, 4 = Local government employee, 5 = Self-employed, 6 = Working without pay in a family business.

FINCBTAX Amount of consumer unit income before taxes in the past 12 months.

SALARYX Wage or salary income received in the past 12 months, before deductions.

SOCRRX Income received from Social Security and Railroad Retirement in the past 12 months.

TOTEXPCQ Total expenditures reported for the current quarter.

TOTXEST Total taxes paid (estimated) in the current period.

EHOUSNGC Total expenditures for housing in the current quarter.

HEALTHCQ Expenditures for health care during the current quarter.

FOODCQ Expenditures on food during the current quarter.

Details

This example dataset is a subset extracted from the complete CE dataset used by the **rpms** package. It is intended to illustrate how to work with survey data in the context of recursive partitioning. The original CE data contain 68,415 observations on 47 variables; this example contains a smaller selection for ease of demonstration. The curated subset of the dataset removed several columns were removed for mostly missing data, redundant data, or not relevant to the examples. Rows were filtered for strictly-positive salary, expenditure, and tax variable values. Weights were not recalibrated following the changes.

Note

For more information on the methodology and details behind the original dataset, please see: Toth, D. (2021). *rpms: Recursive Partitioning for Modeling Survey Data* (v0.5.1). CRAN. https://CRAN.R-project.org/package=rpms.

Source

rpms package on CRAN

See Also

rpms for an overview of the functions provided in the original package.

---

wa_test *Weight-Association Tests for Survey Weights*

Description

Implements several weight-association tests that examine whether survey weights are informative about the response variable after conditioning on covariates. Variants include DuMouchel-Duncan (DD), Pfeffermann-Sverchkov (PS1 and PS2, with optional quadratic terms or user-supplied auxiliary designs), and Wu-Fuller (WF).

Usage

```
wa_test(
  model,
  type = c("DD", "PS1", "PS1q", "PS2", "PS2q", "WF"),
  coef_subset = NULL,
  aux_design = NULL,
  na.action = stats::na.omit
)

## S3 method for class 'wa_test'
print(x, ...)

## S3 method for class 'wa_test'
summary(object, ...)

## S3 method for class 'wa_test'
tidy(x, ...)

## S3 method for class 'wa_test'
glance(x, ...)
```

Arguments

model	An object of class svyglm.
type	Character string specifying the test type: "DD", "PS1", "PS1q", "PS2", "PS2q", "WF".
coef_subset	Optional character vector of coefficient names to include in the test. Defaults to all coefficients.
aux_design	Optional matrix or function to generate auxiliary regressors for PS1/PS2 tests. If a function, it should take X and y and return a matrix of extra columns to include.
na.action	Function to handle missing data before testing.
x	An object of class wa_test
...	Additional arguments passed to methods
object	An object of class wa_test

Details

Let y denote the response, X the design matrix of covariates, and w the survey weights. The null hypothesis in all cases is that the weights are *non-informative* given X, i.e. they do not provide additional information about y beyond the covariates.

The following test variants are implemented:

- **DuMouchel–Duncan (DD)**: After fitting the unweighted regression

$$\hat{\beta} = (X^\top X)^{-1} X^\top y,$$

  compute residuals $e = y - X\hat{\beta}$. The DD test regresses e on the weights w:

$$e = \gamma_0 + \gamma_1 w + u.$$

  A significant γ_1 indicates association between weights and residuals, hence informativeness.

- **Pfeffermann–Sverchkov PS1**: Augments the outcome regression with functions of the weights as auxiliary regressors:

$$y = X\beta + f(w)\theta + \varepsilon.$$

  Under the null, $\theta = 0$. Quadratic terms (w^2) can be included ("PS1q"), or the user may supply a custom auxiliary design matrix $f(w)$.

- **Pfeffermann–Sverchkov PS2**: First regress the weights on the covariates,

$$w = X\alpha + \eta,$$

  and obtain fitted values $\hat{w}$. Then augment the outcome regression with $\hat{w}$ (and optionally $\hat{w}^2$ for "PS2q"):

$$y = X\beta + g(\hat{w})\theta + \varepsilon.$$

  Again, $\theta = 0$ under the null.

- **Wu–Fuller (WF)**: Compares weighted and unweighted regression fits. Let $\hat{\beta}_W$ and $\hat{\beta}_U$ denote the weighted and unweighted estimators. The test statistic is based on

$$T = (\hat{\beta}_W - \hat{\beta}_U)^\top \widehat{\mathrm{Var}}^{-1} (\hat{\beta}_W - \hat{\beta}_U)$$

  and follows an approximate F distribution. A large value indicates that weights materially affect the regression.

In all cases, the reported statistic is an F-test with numerator degrees of freedom equal to the number of auxiliary regressors added, and denominator degrees of freedom equal to the residual degrees of freedom from the augmented regression.

Value

An object of class `"wa_test"` containing:

`statistic`	F-test statistic
`parameter`	Degrees of freedom (numerator, denominator)
`p.value`	P-value for the test
`method`	Name of the test performed
`call`	Function call

References

DuMouchel, W. H., & Duncan, G. J. (1983). Using sample survey weights in multiple regression analyses of stratified samples. *Journal of the American Statistical Association*, 78(383), 535-543.

Pfeffermann, D., & Sverchkov, M. (1999). Parametric and semi-parametric estimation of regression models fitted to survey data. *Sankhya: The Indian Journal of Statistics, Series B*, 61(1), 166-186.

Pfeffermann, D., & Sverchkov, M. (2003). Fitting generalized linear models under informative sampling. In R. L. Chambers & C. J. Skinner (Eds.), *Analysis of Survey Data* (pp. 175-196). Wiley.

Wu, Y., & Fuller, W. A. (2005). Preliminary testing procedures for regression with survey samples. In *Proceedings of the Joint Statistical Meetings, Survey Research Methods Section* (pp. 3683-3688). American Statistical Association.

See Also

`diff_in_coef_test` for the Hausman-Pfeffermann difference-in-coefficients test, and `svytestCE` for the example dataset included in this package.

Examples

```
# Load in survey package (required) and load in example data
library(survey)
data(api, package = "survey")

# Create a survey design and fit a weighted regression model
des <- svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat)
```

```
fit <- svyglm(api00 ~ ell + meals, design = des)

# Run weight-association diagnostic test; reports F-stat, df's, and p-value
results <- wa_test(fit, type = "DD")
print(results)
```

Index