Package ‘rwa’

September 28, 2026

Type Package

Title Perform a Relative Weights Analysis

Version 1.0.1

Description Perform a Relative Weights Analysis (RWA) (a.k.a. Key Drivers Analysis) as per the method described in Tonidandel & LeBreton (2015) <DOI:10.1007/s10869-014-9351-z>, with its original roots in Johnson (2000) <DOI:10.1207/S15327906MBR3501_1>. In essence, RWA decomposes the total variance predicted in a regression model into weights that accurately reflect the proportional contribution of the predictor variables, which addresses the issue of multicollinearity. In typical scenarios, RWA returns similar results to Shapley regression, but with a significant advantage on computational performance.

License GPL-3

Encoding UTF-8

URL https://martinctc.github.io/rwa/, https://github.com/martinctc/rwa

BugReports https://github.com/martinctc/rwa/issues

Depends R (>= 4.1.0)

Imports dplyr, magrittr, stats, tidyr, ggplot2, boot, purrr, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0), rlang, spelling

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Martin Chan [aut, cre]

Maintainer Martin Chan <martinchan53@gmail.com>

Repository CRAN

Date/Publication 2026-09-28 18:10:02 UTC

Contents

plot_rwa 2
remove_all_na_cols 3
rwa 3
rwa_logit 8
rwa_multiregress 9

Index **12**

`plot_rwa` *Plot the rescaled importance values from the output of* `rwa()`

Description

Pass the output of `rwa()` and plot a bar chart of the rescaled importance values. Signs are always calculated and taken into account, which is equivalent to setting the `applysigns` argument to `TRUE` in `rwa()`.

Usage

```
plot_rwa(rwa)
```

Arguments

`rwa`	Direct list output from `rwa()`.

Examples

```
library(ggplot2)
# Use a smaller sample for faster execution
diamonds_small <- diamonds[sample(nrow(diamonds), 1000), ]
diamonds_small %>%
  rwa(outcome = "price",
      predictors = c("depth","carat", "x", "y", "z"),
      applysigns = TRUE) %>%
  plot_rwa()
```

remove_all_na_cols *Remove any columns where all the values are missing*

Description

Pass a data frame and returns a version where all columns made up of entirely missing values are removed.

Usage

```
remove_all_na_cols(df)
```

Arguments

df	Data frame to be passed through.

Details

This is used within `rwa()`.

rwa *Create a Relative Weights Analysis (RWA)*

Description

This function creates a Relative Weights Analysis (RWA) and returns a list of outputs. RWA provides a heuristic method for estimating the relative weight of predictor variables in multiple regression, which involves creating a multiple regression with on a set of transformed predictors which are orthogonal to each other but maximally related to the original set of predictors. `rwa()` is optimised for dplyr pipes and shows positive / negative signs for weights.

Usage

```
rwa(
  df,
  outcome,
  predictors,
  applysigns = FALSE,
  method = "auto",
  sort = TRUE,
  bootstrap = FALSE,
  n_bootstrap = 1000,
  conf_level = 0.95,
  focal = NULL,
  comprehensive = FALSE,
  include_rescaled_ci = FALSE,
```

```
  use = "pairwise.complete.obs",
  weight = NULL
)
```

Arguments

df	Data frame or tibble to be passed through.
outcome	Outcome variable, to be specified as a string or bare input. Must be a numeric variable.
predictors	Predictor variable(s), to be specified as a vector of string(s) or bare input(s). All variables must be numeric.
applysigns	Logical value specifying whether to show an estimate that applies the sign. Defaults to FALSE.
method	String to specify the method of regression to apply. Valid values include: • "auto": automatically detect whether to use multiple regression or logistic regression based on the outcome variable provided. • "multiple": use multiple regression. • "logistic": use logistic regression.
sort	Logical value specifying whether to sort results by rescaled relative weights in descending order. Defaults to TRUE.
bootstrap	Logical value specifying whether to calculate bootstrap confidence intervals. Defaults to FALSE. Currently only supported for multiple regression.
n_bootstrap	Number of bootstrap samples to use when bootstrap = TRUE. Defaults to 1000.
conf_level	Confidence level for bootstrap intervals. Defaults to 0.95.
focal	Focal variable for bootstrap comparisons (optional).
comprehensive	Whether to run comprehensive bootstrap analysis including random variable comparisons and, when focal is supplied, comparisons against that predictor.
include_rescaled_ci	Logical value specifying whether to include confidence intervals for rescaled weights. Defaults to FALSE due to compositional data constraints. Use with caution.
use	Method for handling missing data when computing correlations. Options are: "everything" (remaining missing values propagate, causing a non-finite correlation matrix error if correlations cannot be estimated), "all.obs" (error for remaining missing predictors in unweighted analysis), "complete.obs" (listwise deletion), "na.or.complete" (listwise deletion; no complete cases produces an insufficient-data error rather than an unusable matrix of NAs), "pairwise.complete.obs" (pairwise deletion, default). See cor for more details. Only applicable for multiple regression. Rows with missing outcomes are always removed first, including for "all.obs". When weight is specified, remaining missing weights cause an error for "all.obs" and are removed otherwise. Missing predictors are then removed by listwise deletion for every weighted mode, including "all.obs". Thus weighted correlations always use complete cases, regardless of use; weighted pairwise correlation is not implemented.

`weight` Optional name of a weight variable in the data frame. If provided, a weighted correlation matrix will be computed using the specified weights. Non-missing weights must be numeric, finite, and strictly positive (zero weights are not supported). Missing weights follow the `use` rules. Defaults to `NULL` (unweighted analysis). Only applicable for multiple regression.

Details

`rwa()` produces raw relative weight values (epsilons) as well as rescaled weights (scaled as a percentage of predictable variance) for every predictor in the model. Signs are added to the weights when the `applysigns` argument is set to `TRUE`. See https://www.scotttonidandel.com/rwa-web for the original implementation that inspired this package.

This function is a wrapper around `rwa_multiregress()` and `rwa_logit()`, automatically selecting the appropriate method based on the outcome variable or the `method` argument.

In brief, for the two missing-data and weighting arguments:

- Without `weight`, missing values are handled by `use`, which defaults to pairwise deletion when correlating predictors.
- With `weight`, the analysis always uses complete cases across the outcome, the predictors, and the weight. Weighted pairwise deletion is not implemented, so `use` does not change a weighted result.
- In both cases, rows with a missing outcome are dropped first.

Use weights when the analysis should represent a target population rather than the achieved sample. Comparing weighted with unweighted results is informative: a large difference indicates that sample composition matters.

Multiple-regression estimates require a finite joint correlation matrix. Its smallest eigenvalue must be at least minus `sqrt(.Machine$double.eps) * max(1, max(abs(eigenvalues)))`. The predictor block must have strictly positive computed eigenvalues and the transformation must be solvable. No additional conditioning cutoff is imposed on previously estimable models; highly correlated predictors can still yield sensitive estimates. Calculated R-squared must not exceed one by more than `sqrt(.Machine$double.eps)`; the fit is checked separately because small matrix errors can be amplified by nearly collinear predictors. An exactly fitted outcome (a singular joint matrix) is allowed when the predictor block is positive definite. Invalid matrices, constant variables, and insufficient observations cause informative errors; variables are not dropped and matrices are not silently repaired.

Weighted analysis is observation-weighted RWA. Bootstrap intervals use independent, identically distributed (iid) individual-row resampling, with each row's original weight carried along, not sampling proportional to weights. Rows with missing outcomes are removed before resampling; other missing-data filters are applied within each sample, preserving the outcome-complete sampling frame. A degenerate sample stops the bootstrap with an error: samples are not skipped, retried, or allowed to lose predictors. Clusters, strata, and replicate-weight survey designs are not supported; a weight column alone does not provide general complex-survey variance. See `vignette("weighted-missing-data")` for examples and limitations.

Value

`rwa()` returns a list of outputs, as follows:

- `predictors`: character vector of names of the predictor variables used.
- `rsquare`: the rsquare value of the regression model (multiple regression only).
- `result`: the final output of the importance metrics (sorted by Rescaled.RelWeight in descending order by default).
  - The `Rescaled.RelWeight` column sums up to 100.
  - The `Sign` column indicates whether a predictor is positively or negatively correlated with the outcome.
  - When bootstrap = TRUE, includes confidence interval columns for raw weights.
  - When bootstrap = TRUE, `Random.Diff.CI.Lower`/`Random.Diff.CI.Upper` give the interval for the difference between each predictor's weight and the weight of a randomly generated variable, and `Raw.Significant` is `TRUE` when `Random.Diff.CI.Lower` is above zero. Significance is assessed this way, and not from the interval around the weight itself, because raw relative weights are non-negative: an unrelated predictor still receives a small positive weight, so an interval around it would almost always exclude zero. Comparing each weight to that of a randomly generated variable is the approach suggested by Tonidandel, LeBreton and Johnson (2009; doi:10.1037/a0017735) as a way to judge whether a weight exceeds what chance alone would produce. The `Raw.Significant` cutoff applied here is directional: a predictor is significant only when it exceeds the random variable (`Random.Diff.CI.Lower > 0`), because an interval lying entirely below zero indicates the predictor performed worse than noise, which is not evidence of importance in either direction.
  - Rescaled weight CIs are available via include_rescaled_ci = TRUE but not recommended for inference.
- `n`: complete-case observation count for the selected analysis variables (and weight, if supplied). Unweighted pairwise correlations may use more observations than this conservative count.
- `n_weighted`: weighted results only; sum of original weights after all analysis filters. This is a population-size estimate only for appropriately calibrated weights and the retained population scope.
- `n_effective`: weighted results only; Kish's unequal-weighting effective sample size, `(sum(w)^2) / sum(w^2)`, calculated using scaled weights for numerical stability. This diagnostic ignores clustering, stratification, and weight/outcome relationships; it is not model degrees of freedom or the exact precision of RWA.
- `bootstrap`: bootstrap results (only present when bootstrap = TRUE), containing:
  - `ci_results`: confidence intervals for weights
  - `boot_object`: raw bootstrap object for advanced analysis
  - `n_bootstrap`: number of bootstrap samples used
- `lambda`: lambda matrix from the RWA calculation.
- `RXX`: Correlation matrix of all the predictor variables against each other. Not available for logistic regression.
- `RXY`: Correlation values of the predictor variables against the outcome variable. Not available for logistic regression.

See Also

plot_rwa() for plotting results, rwa_multiregress() and rwa_logit() for the underlying implementations.

Examples

```
library(ggplot2)
# Basic RWA (results sorted by default)
rwa(diamonds, "price", c("depth", "carat"))

# RWA without sorting (preserves original predictor order)
rwa(diamonds, "price", c("depth", "carat"), sort = FALSE)

# Plot results using plot_rwa()
diamonds |>
  rwa("price", c("depth", "carat", "x", "y")) |>
  plot_rwa()


# For faster examples, use a subset of data for bootstrap
diamonds_small <- diamonds[sample(nrow(diamonds), 1000), ]

# RWA with different missing data handling
# Use complete.obs for listwise deletion
rwa(diamonds_small, "price", c("depth", "carat"), use = "complete.obs")

# RWA with weights
diamonds_small$sample_weight <- runif(nrow(diamonds_small), 0.5, 2)
rwa(diamonds_small, "price", c("depth", "carat"), weight = "sample_weight")

# RWA with bootstrap confidence intervals (raw weights only)
rwa(diamonds_small, "price", c("depth", "carat"),
    bootstrap = TRUE, n_bootstrap = 100)

# Include rescaled weight CIs (use with caution for inference)
rwa(diamonds_small, "price", c("depth", "carat"),
    bootstrap = TRUE, include_rescaled_ci = TRUE, n_bootstrap = 100)

# Comprehensive bootstrap analysis with focal variable
result <- rwa(diamonds_small, "price", c("depth", "carat", "table"),
              bootstrap = TRUE, comprehensive = TRUE, focal = "carat",
              n_bootstrap = 100)
# View confidence intervals
result$bootstrap$ci_results


# Based on logistic regression (auto-detected from binary outcome)
diamonds$IsIdeal <- as.numeric(diamonds$cut == "Ideal")
rwa(diamonds, "IsIdeal", c("depth", "carat"))
```

`rwa_logit` *Create a Relative Weights Analysis with logistic regression*

Description

This function performs Relative Weights Analysis (RWA) for binary outcome variables using logistic regression. RWA provides a method for estimating the relative importance of predictor variables by transforming them into orthogonal variables while preserving their relationship to the outcome. This implementation follows Johnson (2000) for logistic regression.

Usage

```
rwa_logit(df, outcome, predictors, applysigns = FALSE)
```

Arguments

`df`	Data frame or tibble to be passed through.
`outcome`	Outcome variable, to be specified as a string or bare input. Must be a numeric variable.
`predictors`	Predictor variable(s), to be specified as a vector of string(s) or bare input(s). All variables must be numeric.
`applysigns`	Logical value specifying whether to show an estimate that applies the sign. Defaults to `FALSE`.

Value

`rwa_logit()` returns a list of outputs, as follows:

- `predictors`: character vector of names of the predictor variables used.
- `rsquare`: the pseudo R-squared value (sum of epsilon weights) for the logistic regression model.
- `result`: the final output of the importance metrics.
  - The `Rescaled.RelWeight` column sums up to 100.
  - The `Sign` column indicates whether a predictor is positively or negatively associated with the outcome.
- `n`: indicates the number of observations used in the analysis.
- `lambda`: the Lambda transformation matrix from the analysis.

Examples

```
# Create a binary outcome variable
mtcars_binary <- mtcars
mtcars_binary$high_mpg <- ifelse(mtcars$mpg > median(mtcars$mpg), 1, 0)

# Basic logistic RWA
result <- rwa_logit(
```

```
  df = mtcars_binary,
  outcome = "high_mpg",
  predictors = c("cyl", "disp", "hp", "wt")
)

# View the relative importance results
result$result

# With sign information
result_signed <- rwa_logit(
  df = mtcars_binary,
  outcome = "high_mpg",
  predictors = c("cyl", "disp", "hp", "wt"),
  applysigns = TRUE
)
result_signed$result
```

`rwa_multiregress` *Create a Relative Weights Analysis (RWA)*

Description

This function creates a Relative Weights Analysis (RWA) and returns a list of outputs. RWA provides a heuristic method for estimating the relative weight of predictor variables in multiple regression, which involves creating a multiple regression with on a set of transformed predictors which are orthogonal to each other but maximally related to the original set of predictors. `rwa_multiregress()` is optimised for dplyr pipes and shows positive / negative signs for weights.

Usage

```
rwa_multiregress(
  df,
  outcome,
  predictors,
  applysigns = FALSE,
  use = "pairwise.complete.obs",
  weight = NULL
)
```

Arguments

`df`	Data frame or tibble to be passed through.
`outcome`	Outcome variable, to be specified as a string or bare input. Must be a numeric variable.
`predictors`	Predictor variable(s), to be specified as a vector of string(s) or bare input(s). All variables must be numeric.

`applysigns` Logical value specifying whether to show an estimate that applies the sign. Defaults to `FALSE`.

`use` Method for handling missing data when computing correlations. Options are: "everything" (remaining missing values propagate, causing a non-finite correlation matrix error if correlations cannot be estimated), "all.obs" (error for remaining missing predictors in unweighted analysis), "complete.obs" (listwise deletion), "na.or.complete" (listwise deletion; no complete cases produces an insufficient-data error rather than an unusable matrix of NAs), "pairwise.complete.obs" (pairwise deletion, default). See `cor` for more details. Only applicable for multiple regression. Rows with missing outcomes are always removed first, including for "all.obs". When `weight` is specified, remaining missing weights cause an error for "all.obs" and are removed otherwise. Missing predictors are then removed by listwise deletion for every weighted mode, including "all.obs". Thus weighted correlations always use complete cases, regardless of `use`; weighted pairwise correlation is not implemented.

`weight` Optional name of a weight variable in the data frame. If provided, a weighted correlation matrix will be computed using the specified weights. Non-missing weights must be numeric, finite, and strictly positive (zero weights are not supported). Missing weights follow the `use` rules. Defaults to `NULL` (unweighted analysis). Only applicable for multiple regression.

Details

`rwa_multiregress()` produces raw relative weight values (epsilons) as well as rescaled weights (scaled as a percentage of predictable variance) for every predictor in the model. Signs are added to the weights when the `applysigns` argument is set to `TRUE`. See https://www.scotttonidandel.com/rwa-web for the original implementation that inspired this package.

This is observation-weighted RWA when `weight` is provided, not a complex-survey variance estimator. See `rwa()` for joint-matrix and predictor positive-definiteness tolerances, bootstrap sampling assumptions, and diagnostics. See `vignette("weighted-missing-data")` for worked examples.

Value

`rwa_multiregress()` returns a list of outputs, as follows:

- `predictors`: character vector of names of the predictor variables used.
- `rsquare`: the rsquare value of the regression model.
- `result`: the final output of the importance metrics.
  - The `Rescaled.RelWeight` column sums up to 100.
  - The `Sign` column indicates whether a predictor is positively or negatively correlated with the outcome.
- `n`: complete-case observation count for the selected variables and weight, if supplied. Unweighted pairwise correlations may use more observations.
- `n_weighted`: weighted results only; sum of original weights after outcome, missing-weight, and predictor-completeness filters. Population-size meaning requires appropriately calibrated weights and retained population scope.

- `n_effective`: weighted results only; Kish's unequal-weighting effective sample size, `(sum(w)^2) / sum(w^2)`, evaluated using scaled weights. This is not model degrees of freedom or exact RWA precision and ignores clustering, stratification, and weight/outcome relationships.
- `lambda`: the transformation matrix that maps the original correlated predictors to orthogonal variables while preserving their relationship to the outcome. Used internally to compute relative weights.
- `RXX`: Correlation matrix of all the predictor variables against each other.
- `RXY`: Correlation values of the predictor variables against the outcome variable.

Examples

```
# Basic multiple regression RWA
result <- rwa_multiregress(
  df = mtcars,
  outcome = "mpg",
  predictors = c("cyl", "disp", "hp", "wt")
)

# View the relative importance results
result$result

# With sign information
result_signed <- rwa_multiregress(
  df = mtcars,
  outcome = "mpg",
  predictors = c("cyl", "disp", "hp", "wt"),
  applysigns = TRUE
)
result_signed$result

# Using listwise deletion for missing data
rwa_multiregress(
  df = mtcars,
  outcome = "mpg",
  predictors = c("cyl", "disp"),
  use = "complete.obs"
)

# With observation weights
mtcars_weighted <- mtcars
mtcars_weighted$w <- runif(nrow(mtcars), 0.5, 2)
rwa_multiregress(
  df = mtcars_weighted,
  outcome = "mpg",
  predictors = c("cyl", "disp"),
  weight = "w"
)
```

Index

cor, *4*, *10*

plot_rwa, 2
plot_rwa(), *7*

remove_all_na_cols, 3
rwa, 3
rwa(), *10*
rwa_logit, 8
rwa_logit(), *7*
rwa_multiregress, 9
rwa_multiregress(), *7*