Package 'SimplexRegression'

July 19, 2026

Type Package

Title Simplex Regression Models with Parametric or Fixed Mean Link Functions

Version 0.1.5

Author Maria Eduarda da Cruz Justino [aut, cre] (ORCID: <https://orcid.org/0000-0001-9501-2642>), Francisco Cribari-Neto [ctb, ths] (ORCID: <https://orcid.org/0000-0002-5909-6698>)

Maintainer Maria Eduarda da Cruz Justino <eueduardacruz@gmail.com>

Description Fits and analyzes simplex regression models with either fixed or parametric mean link functions. Implements the simplex probability density function, cumulative distribution function, quantile function, random number generation, and variance evaluation. Offers several fixed and parametric link functions for the mean submodel, tools for residual analysis and diagnostic plotting, hypothesis testing procedures, and influence measures such as Cook's distance and leverage (hat values). Includes the Scout Score (SS) criterion for model selection, enabling comprehensive inference and diagnostic analysis within the simplex regression framework. For more details see Barndorff-Nielsen and Jorgensen (1991) <doi:10.1016/0047-259X(91)90008-P> and Justino and Cribari-Neto (2026) <doi:10.1016/j.apm.2025.116713>.

URL https://github.com/dudajustino/SimplexRegression

BugReports https://github.com/dudajustino/SimplexRegression/issues

Depends R (>= 3.5)

Imports Formula, lmtest, Matrix, moments, parallel, pracma, sandwich

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

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-07-19 18:30:02 UTC

Contents

`AbortionOpposition` *Public Opinion on Abortion Across U.S. States*

Description

This dataset examines the relationship between public opposition to abortion and demographic, religious, and socioeconomic characteristics across the 50 U.S. states and the District of Columbia. The response variable is the proportion of adults who believe abortion should be illegal in all or most cases, bounded in the open interval (0, 1), making the data suitable for simplex regression and beta regression analyses.

Usage

```
data(AbortionOpposition)
```

Format

A data frame with 51 observations and 9 variables:

state Factor. U.S. state or the District of Columbia.

abortion_opp Numeric. Proportion of adults who believe abortion should be illegal in all or most cases, based on the Pew Research Center 2023–24 Religious Landscape Study.

relig_attend Numeric. Percentage of adults who report attending religious services at least once a week, based on the Pew Research Center 2023–24 Religious Landscape Study.

income Numeric. Mean annual household income (USD), 2024.

sex_ratio Numeric. Number of males per 100 females, 2024.

female Numeric. Percentage of the population that is female, 2024.

pop_18_24 Numeric. Percentage of the population aged 18–24 years, obtained directly from the 2024 American Community Survey (ACS) 1-Year Estimates.

bachelors Numeric. Percentage of adults aged 25 and older with a bachelor's degree or higher, 2024.

urban Numeric. Percentage of the population living in urban areas, based on 2020 U.S. Census Bureau urban area criteria (a densely settled core of census blocks meeting minimum housing unit and/or population density thresholds, requiring at least 2,000 housing units or 5,000 people).

Details

The dataset was assembled by the package authors from multiple publicly available sources. The response variable, `abortion_opp`, was obtained from the Pew Research Center 2023–24 Religious Landscape Study, originally published as a percentage (0–100 scale), and rescaled to the (0, 1) interval by dividing by 100.

The explanatory variables were obtained as follows:

1. relig_attend was obtained from the Pew Research Center 2023–24 Religious Landscape Study.
2. income, sex_ratio, and bachelors were obtained from World Population Review, which itself compiles estimates from the U.S. Census Bureau/American Community Survey. World Population Review is cited here as the immediate data source, following standard practice for reproducibility.
3. female was obtained from the U.S. Census Bureau's American Community Survey (ACS) 1-Year Estimates, Table S0101 (Age and Sex), the same table used for pop_18_24. Note that sex_ratio and female are closely related (both describe the sex composition of each state's population, from different sources) and should generally not be included together as covariates in the same regression model, as doing so may introduce near-perfect collinearity.
4. pop_18_24 corresponds to the percent estimate of the population aged 18–24 years reported directly in Table S0101 (Age and Sex) of the 2024 American Community Survey (ACS) 1-Year Estimates.
5. urban was obtained from World Population Review and is based on 2020 U.S. Census Bureau data, the most recent Census estimate of urbanization by state available at the time this dataset was compiled. Unlike the other explanatory variables, which reflect 2023–24 estimates, urban reflects the 2020 Census urban area delineation (see @format above for definition details).

Source

Pew Research Center (2025). 2023-24 Religious Landscape Study (RLS) Dataset. doi:10.58094/3kwbbf52. Data accessed in 2026.

World Population Review (2024). Per Capita Income by State. https://worldpopulationreview.com/state-rankings/per-capita-income-by-state. Data accessed in 2026.

World Population Review (2024). Sex Ratio by State. https://worldpopulationreview.com/state-rankings/sex-ratio-by-state. Data accessed in 2026.

World Population Review (2024). Educational Attainment by State. https://worldpopulationreview.com/state-rankings/educational-attainment-by-state. Data accessed in 2026.

World Population Review (2024). Most Urban States. https://worldpopulationreview.com/state-rankings/most-urban-states. Data accessed in 2026.

U.S. Census Bureau (2024). Age and Sex. American Community Survey, ACS 1-Year Estimates Subject Tables, Table S0101. https://data.census.gov/table/ACSST1Y2024.S0101?q=age+and+sex+by+state&moe=false. Data accessed in 2026.

Examples

```
# Load the data
data(AbortionOpposition)

# Quick overview
head(AbortionOpposition)
str(AbortionOpposition)

# Summary statistics
summary(AbortionOpposition)
```

```
# Relationship between religious attendance and abortion opposition
plot(abortion_opp ~ relig_attend,
     data = AbortionOpposition,
     pch = 19,
     xlab = "Religious attendance (%)",
     ylab = "Abortion opposition (proportion)")

# Correlation among numeric variables
cor(AbortionOpposition[, c("abortion_opp",
                           "relig_attend",
                           "income",
                           "sex_ratio",
                           "female",
                           "pop_18_24",
                           "bachelors",
                           "urban")])
```

`AISRowing` *Body Composition Data for Australian Rowers*

Description

Hematological, body composition, and anthropometric measurements on 37 elite rowers (22 female, 15 male) at the Australian Institute of Sport (AIS), a subset of a larger dataset collected across multiple sports. The data are useful for investigating sex-based differences in blood and body composition among highly trained athletes.

Usage

```
data(AISRowing)
```

Format

A data frame with 37 observations and 12 variables:

sex Factor. Sex of the athlete (`female` or `male`).

rcc Numeric. Red blood cell count (in 10^{12} per litre).

wcc Numeric. White blood cell count (in 10^{12} per litre).

hc Numeric. Hematocrit (in percent).

hb Numeric. Hemoglobin concentration (in g per decilitre).

ferr Numeric. Plasma ferritin concentration (in ng per millilitre).

bmi Numeric. Body mass index (in kg per metre-squared).

ssf Numeric. Sum of skin folds (in mm).

bfat Numeric. Body fat proportion, originally measured in percent and rescaled to the unit interval (0, 1).

lbm Numeric. Lean body mass (in kg).

ht Numeric. Height (in cm).

wt Numeric. Weight (in kg).

Details

The original measurements were collected in the late 1980s by Richard Telford and Ross Cunningham at the Australian Institute of Sport (AIS), and were later compiled and popularized by Cook and Weisberg (1994). `AISRowing` is the 37-athlete subset corresponding to rowing, out of 202 athletes across all sports in the original dataset. The full dataset (covering all sports, not just rowing) is also discussed in Weisberg (2005, Section 6.4).

Source

Telford, R. D. and Cunningham, R. B. (1991). Sex, sport, and body-size dependency of hematology in highly trained athletes. *Medicine & Science in Sports & Exercise*, **23**(7), 788–794.

References

Cook, R. D. and Weisberg, S. (1994). *An Introduction to Regression Graphics*. John Wiley & Sons, New York.

Weisberg, S. (2005). *Applied Linear Regression*, 3rd edition. New York: Wiley, Section 6.4.

Examples

```
# Load the data
data(AISRowing)

# Quick overview
head(AISRowing)
str(AISRowing)

# Summary statistics
summary(AISRowing)

# Sex-based comparison
aggregate(bfat ~ sex, data = AISRowing, summary)
boxplot(bfat ~ sex, data = AISRowing,
        main = "Body Fat Proportion by Sex",
        ylab = "Body fat (proportion)")
```

`Biomass` *Biomass Allocation in Two Grass Species Under Different Nitrate Supply*

Description

This dataset examines biomass allocation patterns in plants, specifically the proportional distribution of biomass to different plant organs (stems, leaves, and roots). The data come from an experiment manipulating nitrate supply in fast-growing and slow-growing grass species.

The response variables (stem, leaves, roots) are proportions bounded in the (0, 1) interval, making them suitable for simplex regression analysis.

Usage

```
data(Biomass)
```

Format

A data frame with 500 observations and 13 variables:

group Factor. Combined species-by-nitrate-treatment code (`DfH`, `DfL`, `HlH`, `HlL`), corresponding to the combination of `species` and `trt` where:

- `DfH` = D. flexuosa (slow-growing), high nitrate
- `DfL` = D. flexuosa (slow-growing), low nitrate
- `HlH` = H. lanatus (fast-growing), high nitrate
- `HlL` = H. lanatus (fast-growing), low nitrate

species Factor. Species scientific name (`D. flexuosa` or `H. lanatus`).

trt Factor. Nitrate treatment level (`high` or `low`).

day Numeric. Experimental day (0 to 49).

pl_num Numeric. Plant number (individual plant identifier, 6-8 replicates per treatment).

ldm_mg Numeric. Leaf dry mass (in mg).

sdm_mg Numeric. Stem dry mass (in mg).

rdm_mg Numeric. Root dry mass (in mg).

tdm_mg Numeric. Total dry mass (in mg).

lmf Numeric. Leaf mass fraction, proportion of biomass allocated to leaves, bounded in the open interval (0, 1).

smf Numeric. Stem mass fraction, proportion of biomass allocated to stems, bounded in the open interval (0, 1).

rmf Numeric. Root mass fraction, proportion of biomass allocated to roots, bounded in the open interval (0, 1).

ln_tdm Numeric. Natural log of total dry mass (log-transformed for allometric analysis).

Source

bobdouma (2019). *bobdouma/proportions_beta_Dirichlet: v.01*. doi:10.5281/zenodo.3234670.

References

bobdouma (2019). *bobdouma/proportions_beta_Dirichlet: v.01*. doi:10.5281/zenodo.3234670.

Poorter, H.; van de Vijver, C. A. D. M.; Boot, R. G. A. and Lambers, H. (1995). Growth and carbon economy of a fast-growing and a slow-growing grass species as dependent on nitrate supply. *Plant and Soil*, **171**, 217–227. doi:10.1007/BF00010275

Poorter, H. and Sack, L. (2012). Pitfalls and possibilities in the analysis of biomass allocation patterns in plants. *Frontiers in Plant Science*, **3**, 259. doi:10.3389/fpls.2012.00259

Examples

```
# Load the data
data(Biomass)

# Quick overview
head(Biomass)
str(Biomass)

# Check that proportions sum to 1 (within rounding error)
summary(rowSums(Biomass[, c("lmf", "smf", "rmf")]))

# Simple plot of root mass fraction by treatment
boxplot(rmf ~ trt * species, data = Biomass,
        main = "Root Mass Fraction by Species and Nitrate Treatment",
        las = 2)
```

`cooks.distance.simplexregression` *Cook's Distance for Simplex Regression Models*

Description

Computes approximate Cook's distances for simplex regression models with parametric or fixed mean link function.

Usage

```
## S3 method for class 'simplexregression'
cooks.distance(model, type = c("pearson", "weighted"), ...)
```

Arguments

`model`	An object of class `"simplexregression"`.
`type`	Character string indicating the type of residual used in the influence measure: `"pearson"` (default) or `"weighted"`.
`...`	Currently not used.

Details

Cook's distance measures the influence of each observation on the estimated regression coefficients. It combines the leverage of an observation (see `hatvalues.simplexregression`) with the magnitude of its residual. Observations with high Cook's distance may have a disproportionate effect on the fitted model.

Two approximate versions are available, depending on `type`:

- `"pearson"`: the conventional approximate Cook's distance, based on the Pearson residual, analogous to the measure used in beta regression (Ferrari and Cribari-Neto, 2004).
- `"weighted"`: the approximate Cook's distance based on the weighted residual, as proposed specifically for the simplex regression model by Espinheira and Silva (2026).

Value

A numeric vector of Cook's distance values.

References

Cook, R. D. (1977). Detection of influential observation in linear regression. *Technometrics*, **19**(1), 15–18. doi:10.2307/1268249

Espinheira, P. L. and Silva, A. O. (2026). Prediction in the nonlinear simplex model. *International Journal of Data Science and Analytics*, **22**, 161. doi:10.1007/s41060026011149

Ferrari, S. L. P. and Cribari-Neto, F. (2004). Beta regression for modeling rates and proportions. *Journal of Applied Statistics*, **31**(7), 799–815. doi:10.1080/0266476042000214501

See Also

`hatvalues.simplexregression`, `gleverage.simplexregression`, `residuals.simplexregression`.

`dev.unit.simplex` *Unit Deviance Function of the Simplex Distribution*

Description

Computes the unit deviance (scaled deviance component) for the simplex distribution.

Usage

```
dev.unit.simplex(y, mu)
```

Arguments

`y`	Numeric scalar or vector of observed values ($0 < y < 1$). If a vector, must be the same length as `mu` or recyclable.
`mu`	Numeric scalar or vector of mean values ($0 < \mu < 1$). If a vector, must be the same length as `y` or recyclable.

Details

The unit deviance for the simplex distribution is defined as:

$$d(y, \mu) = \frac{(y - \mu)^2}{y(1 - y)[\mu(1 - \mu)]^2}.$$

This function is used internally in maximum likelihood estimation and model diagnostics for simplex regression.

Value

A numeric scalar or vector of unit deviance values.

References

Jørgensen, B. (1997). *The Theory of Dispersion Models*. Chapman and Hall, London.

Song, P. X.-K. and Tan, M. (2000). Marginal models for longitudinal continuous proportional data. *Biometrics*, **56**(2), 496–502. doi:10.1111/j.0006341X.2000.00496.x

See Also

variance.simplex, dsimplex.

Examples

```
# Single value
dev.unit.simplex(y = 0.6, mu = 0.5)

# Vector of values
y_vec <- c(0.2, 0.5, 0.8)
mu_vec <- c(0.3, 0.5, 0.7)
dev.unit.simplex(y = y_vec, mu = mu_vec)

# Perfect fit returns zero deviance
dev.unit.simplex(y = 0.5, mu = 0.5)
```

---

diag.distances *Distance-Based Influence Diagnostics for Simplex Regression*

Description

Computes leave–one–out influence measures based on distributional distances (Wasserstein with $p_W = 1$ and $p_W = 2$ or Hellinger) for simplex regression models, as proposed by Justino and Cribari-Neto (2026).

Usage

```
diag.distances(
  model,
  data,
  type = c("W1", "W2", "H"),
  plot = FALSE,
  verbose = TRUE,
  ncores = 1,
  label.pos = 3,
  plot.type = NULL,
  ...
)
```

Arguments

`model`	An object of class `"simplexregression"`.
`data`	The data frame used to fit `model`.
`type`	Character string or integer specifying the distance measure: `"W1"` (default, Wasserstein with $p_W = 1$), `"W2"` (Wasserstein with $p_W = 2$), `"H"` (Hellinger).
`plot`	Logical; if `FALSE` (default), returns the numeric vector of distances. If `TRUE`, produces an index plot with the ad hoc threshold and flagged-observation labels.
`verbose`	Logical; if `TRUE` (default), prints progress during leave–one–out refitting. Ignored when `ncores` > 1, since output from parallel workers does not reach the main console.
`ncores`	Positive integer specifying the number of CPU cores to use for the leave–one–out loop. Default is 1 (sequential). Values greater than 1 activate parallel computation via `parLapply`. If `ncores` exceeds `parallel::detectCores() - 1`, it is clamped to that value with a warning to avoid overloading the system. A safe explicit choice is `parallel::detectCores() - 1`.
`label.pos`	Position(s) for outlier labels in the plot. Can be a single value (applied to all labels) or a vector. Values: 1 = below, 2 = left, 3 = above, 4 = right.
`plot.type`	Character string controlling the plot style when `plot = TRUE`. If `NULL` (default), uses `"h"` for $n \leq 150$ and `"p"` for $n > 150$ (automatic). Passed to the `type` argument of `plot()`.
`...`	Additional graphical parameters passed to `plot()`.

Details

Let $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}^{\mathbf{2}}$ denote the vectors of maximum likelihood estimates obtained by fitting the model to the complete dataset. For each $i = 1, \ldots, n$, the model is refit after omitting observation i, yielding the leave–one–out estimate vectors $\hat{\boldsymbol{\mu}}^{(i)}$ and $\hat{\boldsymbol{\sigma}}^{\mathbf{2}(i)}$. The fitted simplex density (see `dsimplex`) for observation j is, $f(y; \hat{\mu}_j, \hat{\sigma}_j^2)$ under the full data and $f(y; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)})$ under the fit with observation i deleted.

Hellinger distance. For observation j, under deletion of observation i,

$$H_j^{(i)} = H\big(f(y; \hat{\mu}_j, \hat{\sigma}_j^2),\ f(y; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)})\big) = \sqrt{1 - \rho_j^{(i)}},$$

$$\rho_j^{(i)} = \int_0^1 \sqrt{f(y; \hat{\mu}_j, \hat{\sigma}_j^2)\, f(y; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)})}\, dy,$$

where $\rho_j^{(i)}$ is Bhattacharyya's coefficient, evaluated via numerical integration (`quadgk`) from the **pracma** package. The overall influence measure for the deletion of observation i is the aggregate distance $I_i^H = \sum_{j=1}^n H_j^{(i)}$.

Wasserstein distance. Let $F(y; \hat{\mu}_j, \hat{\sigma}_j^2)$ and $F(y; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)})$ denote the simplex cumulative distribution functions (see `psimplex`) fitted for observation j with the full data and with observation i deleted, respectively. For $p_W = 1$,

$$W_{1,j}^{(i)} = \int_0^1 \left| F(y; \hat{\mu}_j, \hat{\sigma}_j^2) - F(y; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)}) \right| dy,$$

computed by numerically integrating the absolute difference between the fitted simplex CDFs. For general $p_W \geq 1$, with $F^{-1}(u; \hat{\mu}_j, \hat{\sigma}_j^2)$ and $F^{-1}(u; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)})$ denoting the corresponding fitted simplex quantile functions (see `qsimplex`),

$$W_{p_W,j}^{(i)} = \left(\int_0^1 \left| F^{-1}(u; \hat{\mu}_j, \hat{\sigma}_j^2) - F^{-1}(u; \hat{\mu}_j^{(i)}, \hat{\sigma}_j^{2(i)}) \right|^{p_W} du \right)^{1/p_W},$$

computed by numerically integrating the absolute difference between the fitted simplex quantile functions raised to the power p_W. As in the Hellinger case, the total influence measure for observation i is $I_i^W = \sum_{j=1}^n W_{p_W,j}^{(i)}$.

Because neither the simplex density nor its quantile function admits a closed-form expression, every distance computed by this function relies on numerical integration. For $n > 500$ this may be slow; consider using a subset or a faster integration method.

Ad hoc threshold for identifying influential observations uses an asymmetric interquartile range (IQR) adjusted for skewness, as proposed by Justino and Cribari-Neto (2026). Let $Q(p)$ denote the empirical p-th quantile of the distances I_i, the threshold is

$$\text{threshold} = Q(0.75) + (1 + a)(Q(0.75) - Q(0.25))$$

where a is the sample skewness of the distances. Observations with I_i above this threshold are flagged as potentially influential.

For the full derivation and rationale behind these measures, see Justino and Cribari-Neto (2026).

Parallel computation: when `ncores > 1`, the leave–one–out loop is distributed across workers using `parLapply` from the **parallel** package (included in base R). Each worker receives the necessary objects and loads the **SimplexRegression** package. The random-number stream is initialized with a fixed seed via `clusterSetRNGStream` to ensure reproducibility across runs. Progress messages (`verbose`) are suppressed in parallel mode because worker output does not reach the main console.

Value

If `plot = FALSE`, a list containing:

`distances` Numeric vector of length n with the leave–one–out distances.

`threshold` Named numeric scalar with the ad hoc upper threshold.

`outliers` Data frame of flagged observations (index and distance value). An empty data frame if no observations are flagged.

`type` Distance type used (full label).

`n` Number of observations.

If `plot = TRUE`, the same list is returned invisibly.

References

Justino, M. E. C. and Cribari-Neto, F. (2026). Influence diagnostics in beta regression via Hellinger and Wasserstein distances. *Statistical Papers*, **67**(3), 50. doi:10.1007/s00362026018230

See Also

`diag.im`, `local.influence`, `gleverage`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)


# Sequential (default) — Wasserstein W1
dd <- diag.distances(fit, data = ReadingSkills, type = "W1")

# Index plot with flagged observations
diag.distances(fit, data = ReadingSkills, type = "W1", plot = TRUE)

# Hellinger distance
diag.distances(fit, data = ReadingSkills, type = "H", plot = TRUE)
```

---

diag.im *Sample Influence Measures for Simplex Regression Models*

Description

Computes leave–one–out sample influence measures $s_{3,i}$ and $s_{5,i}$ for simplex regression models, based on the information–matrix–based criteria measures proposed by Cribari-Neto, Vasconcellos and Santana-e-Silva (2025).

Usage

```
diag.im(
  model,
  data,
  type = c("s3", "s5"),
  interval = c("I1", "I2"),
  parameter = c("theta", "beta", "gamma"),
  plot = FALSE,
  verbose = TRUE,
  ncores = 1,
  label.pos = 3,
  plot.type = NULL,
  ...
)
```

Arguments

model	An object of class "simplexregression".
data	The data frame used to fit model.

`type`	Character vector specifying measure(s): `"s3"`, `"s5"`, or both (default).
`interval`	Character string specifying the outlier detection threshold: `"I1"` (default, moderate) or `"I2"` (strict).
`parameter`	Character string indicating the parameter block: `"theta"` (default, all parameters), `"beta"` (mean submodel), or `"gamma"` (dispersion submodel).
`plot`	Logical; if `TRUE`, produces index plots of $s_{3,i}$ and $s_{5,i}$ with threshold lines and flagged-observation labels. Default is `FALSE`.
`verbose`	Logical; if `TRUE` (default), prints progress during leave–one–out refitting. Ignored (set to `FALSE`) when `ncores > 1`, since output from parallel workers does not reach the main console.
`ncores`	Positive integer specifying the number of CPU cores to use for the leave–one–out loop. Default is 1 (sequential). Values greater than 1 activate parallel computation via `parLapply`. If `ncores` exceeds `parallel::detectCores() - 1`, it is clamped to that value with a warning to avoid overloading the system. A safe explicit choice is `parallel::detectCores() - 1`.
`label.pos`	Position(s) for outlier labels in plot. Can be a single value (applied to all labels) or a vector. Values: 1 = below, 2 = left, 3 = above, 4 = right.
`plot.type`	Character string controlling the plot style when `plot = TRUE`. If `NULL` (default), uses `"h"` for $n \leq 150$ and `"p"` for $n > 150$ (automatic). Passed to the `type` argument of `plot()`.
`...`	Additional graphical parameters passed to `plot()`.

Details

Let $\boldsymbol{\theta}$ denote the vector of parameters that index the simplex regression model, $\boldsymbol{y}$ the vector of observed values of the response variable, and $\ell(\boldsymbol{\theta}; \boldsymbol{y})$ the total log-likelihood function. Let

$$A_n(\boldsymbol{\theta}; \boldsymbol{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial^2 \ell(\boldsymbol{\theta}; y_i)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} \qquad \text{and} \qquad B_n(\boldsymbol{\theta}; \boldsymbol{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \ell(\boldsymbol{\theta}; y_i)}{\partial \boldsymbol{\theta}} \frac{\partial \ell(\boldsymbol{\theta}; y_i)}{\partial \boldsymbol{\theta}'}$$

denote, respectively, the sample average of the second-order log-likelihood derivatives and the sample average of the outer products of the individual score vectors, both evaluated at the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ obtained from the complete dataset.

For each $i = 1, \ldots, n$, the model is refit on the dataset with observation i omitted, yielding the leave–one–out estimate $\hat{\boldsymbol{\theta}}_{(i)}$. The matrices $A_{n-1}^{(i)}$ and $B_{n-1}^{(i)}$ are then recomputed over the remaining $n - 1$ observations, evaluated at $\hat{\boldsymbol{\theta}}_{(i)}$.

Measures computed:

$$s_{3,i} = m_{3,(i)}/m_3 \qquad \text{and} \qquad s_{5,i} = D_i^{\text{mod}} - D_i^{\text{gen}}.$$

Here $m_3 = \|\text{vech}(P_n^{-1}(\hat{\boldsymbol{\theta}}; \boldsymbol{y}) B_n(\hat{\boldsymbol{\theta}}; \boldsymbol{y}) P_n^{-1}(\hat{\boldsymbol{\theta}}; \boldsymbol{y})' - I)\|_2$, where $P_n(\hat{\boldsymbol{\theta}}; \boldsymbol{y})$ is obtained from the Cholesky decomposition of $-A_n(\hat{\boldsymbol{\theta}}; \boldsymbol{y})$, i.e. $-A_n(\hat{\boldsymbol{\theta}}; \boldsymbol{y}) = P_n(\hat{\boldsymbol{\theta}}; \boldsymbol{y}) P_n(\hat{\boldsymbol{\theta}}; \boldsymbol{y})'$, and I is the identity matrix. The quantity $m_{3,(i)}$ is the same measure recomputed using $P_{n-1}^{(i)}$ and $B_{n-1}^{(i)}$, both evaluated at $\hat{\boldsymbol{\theta}}_{(i)}$ over the remaining $n - 1$ observations.

Furthermore,

$$D_i^{\text{gen}} = (n-1)\,(\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_{(i)})'(-A_{n-1}^{(i)}(\hat{\boldsymbol{\theta}}_{(i)}; \boldsymbol{y}))(\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_{(i)})$$

$$D_i^{\text{mod}} = 0.5(n-1)\,(\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_{(i)})'(-A_{n-1}^{(i)}(\hat{\boldsymbol{\theta}}_{(i)}; \boldsymbol{y}) + B_{n-1}^{(i)}(\hat{\boldsymbol{\theta}}_{(i)}; \boldsymbol{y}))(\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_{(i)})$$

where D_i^{gen} is the generalized Cook's distance and D_i^{mod} is its modification proposed by Cribari-Neto, Vasconcellos and Santana-e-Silva (2025). For the rationale behind these measures and the underlying information-matrix-equality argument, see Cribari-Neto, Vasconcellos and Santana-e-Silva (2025).

Threshold intervals use two asymmetric IQR spreads, as proposed by Cribari-Neto, Vasconcellos and Santana-e-Silva (2025). Let $Q(p)$ denote the empirical p-th quantile of the relevant measure ($s_{3,i}$ or $s_{5,i}$, computed separately for each): $IQR_1 = Q(0.50) - Q(0.125)$ (left) and $IQR_2 = Q(0.875) - Q(0.50)$ (right). Limits are $v - z \cdot IQR_1$ (lower) and $v + z \cdot IQR_2$ (upper), with reference value $v = 1$ for s_3 and $v = 0$ for s_5:

- `I1` (moderate): $z = 2.5$ for s_3; $z = 4.0$ for s_5.
- `I2` (strict): $z = 5.0$ for s_3; $z = 8.0$ for s_5.

An observation i is flagged as influential if its measure ($s_{3,i}$ or $s_{5,i}$) falls below the lower limit or above the upper limit of the corresponding interval.

If $-A_{n-1}^{(i)}$ is not positive definite, `nearPD` from the **Matrix** package is used to find the nearest positive-definite matrix and a message is printed.

`parameter = "gamma"` is not available when the dispersion submodel is fixed (intercept-only); the function stops with an informative error in that case. Leave–one–out refits that fail to converge produce `NA` for that observation in `s3_i`/`s5_i`; a warning is issued via the refit's internal error handler and the observation is kept as `NA` in the output.

Parallel computation: when `ncores > 1`, the leave–one–out loop is distributed across workers using `parLapply` from the **parallel** package (included in base R). Each worker receives the necessary objects and loads the **SimplexRegression** package. The random-number stream is initialized with a fixed seed via `clusterSetRNGStream` to ensure reproducibility across runs. Progress messages (`verbose`) are suppressed in parallel mode because worker output does not reach the main console.

Value

If `plot = FALSE` (default), a list containing only the requested measures:

`s3_i` (if requested) Numeric vector of $s_{3,i}$ values.

`s5_i` (if requested) Numeric vector of $s_{5,i}$ values.

`outliers_s3` (if requested) Data frame of flagged observations for $s_{3,i}$.

`outliers_s5` (if requested) Data frame of flagged observations for $s_{5,i}$.

`limits_s3` (if requested) Named vector with lower and upper thresholds for $s_{3,i}$.

`limits_s5` (if requested) Named vector with lower and upper thresholds for $s_{5,i}$.

`interval` Interval type used.

`parameter` Parameter block used.

`n` Number of observations.

If `plot = TRUE`, the same list is returned invisibly.

References

Cribari-Neto, F.; Vasconcellos, K. L. P. and Santana-e-Silva, J. J. (2025). New strategies for detecting atypical observations based on the information matrix equality. *Journal of Applied Statistics*, **52**, 2873–2893. doi:10.1080/02664763.2025.2487914

See Also

`diag.distances`, `local.influence`, `gleverage`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)


# Sequential (default)
im <- diag.im(fit, data = ReadingSkills, type = "s3", interval = "I1",
              parameter = "theta")

# Produce index plots directly
diag.im(fit, data = ReadingSkills, type = "s3", interval = "I1",
        parameter = "theta", plot = TRUE)
```

`dispersion_links` *Dispersion Link Functions and Their Derivatives*

Description

Provides the link function, its inverse, and derivative for the dispersion submodel in the simplex regression. Supported link types are: `"log"`, `"sqrt"` and `"identity"`.

Usage

```
dispersion_link(sigma2, type = c("log", "sqrt", "identity"))

dispersion_link_inv(eta, type = c("log", "sqrt", "identity"))

dispersion_link_deriv1(sigma2, type = c("log", "sqrt", "identity"))

dispersion_link_inv_deriv1(eta, type = c("log", "sqrt", "identity"))
```

Arguments

`sigma2`	Dispersion parameter (numeric vector, $\sigma^2 > 0$).
`type`	Type of link: `"log"`, `"sqrt"` or `"identity"`.
`eta`	Linear predictor of dispersion (numeric vector).

Details

Available link functions:

- Log (`"log"`): $h(\sigma^2) = \log(\sigma^2)$ (ensures positivity);
- Sqrt (`"sqrt"`): $h(\sigma^2) = \sqrt{\sigma^2}$;
- Identity (`"identity"`): $h(\sigma^2) = \sigma^2$ (no transformation).

Value

Numeric vector with transformed values.

See Also

simplexreg, fixed_mean_links, parametric_mean_links.

Examples

```
dispersion_link(1.5, type = "log")
dispersion_link(c(0.5, 1, 2), type = "sqrt")
dispersion_link_inv(0, type = "log")
dispersion_link_deriv1(1, type = "log")
dispersion_link_inv_deriv1(0, type = "log")
```

---

fixed_mean_links *Fixed Mean Link Functions and Derivatives*

Description

Provides the fixed mean link functions, their inverses, and derivatives for the simplex regression model. Supported link types are: `"logit"`, `"probit"`, `"loglog"`, `"cloglog"`, and `"cauchit"`.

Usage

```
fixed_mean_link(
  mu,
  type = c("logit", "probit", "loglog", "cloglog", "cauchit")
)

fixed_mean_link_inv(
  eta,
  type = c("logit", "probit", "loglog", "cloglog", "cauchit")
)

fixed_mean_link_deriv1(
  mu,
  type = c("logit", "probit", "loglog", "cloglog", "cauchit")
```

```
)

fixed_mean_link_deriv2(
  mu,
  type = c("logit", "probit", "loglog", "cloglog", "cauchit")
)

fixed_mean_link_inv_deriv1(
  eta,
  type = c("logit", "probit", "loglog", "cloglog", "cauchit")
)
```

Arguments

`mu`	Mean parameter (numeric vector, $0 < \mu < 1$).
`type`	Type of link function: `"logit"`, `"probit"`, `"loglog"`, `"cloglog"`, or `"cauchit"`.
`eta`	Linear predictor of mean (numeric vector).

Details

Available link functions:

- Logit (`"logit"`): $g(\mu) = \log(\mu/(1-\mu))$;
- Probit (`"probit"`): $g(\mu) = \Phi^{-1}(\mu)$;
- Log-log (`"loglog"`): $g(\mu) = -\log(-\log(\mu))$;
- Complementary log-log (`"cloglog"`): $g(\mu) = \log(-\log(1-\mu))$;
- Cauchit (`"cauchit"`): $g(\mu) = \tan(\pi(\mu - 0.5))$.

Value

A numeric vector corresponding to the evaluated link, its inverse, or derivative depending on the function.

See Also

simplexreg, parametric_mean_links, dispersion_links.

Examples

```
fixed_mean_link(0.5, type = "logit")
fixed_mean_link(c(0.2, 0.5, 0.8), type = "probit")
fixed_mean_link_inv(eta = 0.2, type = "logit")
fixed_mean_link_deriv1(mu = 0.5, type = "logit")
```

`gleverage` *Generalized Leverage Values for Simplex Regression Models*

Description

Compute the generalized leverage values for simplex regression models with parametric or fixed mean link function.

Usage

```
gleverage(model)

## S3 method for class 'simplexregression'
gleverage(model)
```

Arguments

`model`	An object of class `"simplexregression"`.

Details

`gleverage` computes generalized leverage values as suggested by Wei, Hu, and Fung (1998). Generalized leverage extends the concept of hat values to account for both mean and dispersion parameters. High leverage values indicate observations that have potentially large influence on parameter estimates.

Value

A numeric vector of generalized leverage values.

References

Justino, M. E. C. and Cribari-Neto, F. (2026). Simplex regression with a flexible logit link: Inference and application to cross-country impunity data. *Applied Mathematical Modelling*, **154**, 116713. doi:10.1016/j.apm.2025.116713

Wei, B. C., Hu, Y. Q. and Fung, W. K. (1998). Generalized leverage and its applications. *Scandinavian Journal of Statistics*, **25**, 25–37.

See Also

`hatvalues.simplexregression`, `cooks.distance.simplexregression`, `local.influence`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

# Compute generalized leverage
glev <- gleverage(fit)

# Plot leverage values
plot(glev, type = "h", ylab = "Generalized leverage",
     xlab = "Observation index")
abline(h = 2 * mean(glev), lty = 2, col = "red")
```

`halfnormal.plot` *Half-Normal Plots with Simulated Envelopes for Simplex Regression*

Description

Produces half-normal plots with simulated envelopes for simplex regression model with parametric or fixed mean link function.

Usage

```
halfnormal.plot(
  model,
  type = c("weighted", "quantile", "pearson", "deviance", "standardized", "variance",
    "biasvariance", "score", "dualscore", "response"),
  nsim = 100,
  level = 0.95,
  seed = 1987,
  ...
)
```

Arguments

`model`	An object of class `"simplexregression"`.
`type`	Character string specifying the residual type (default: `"weighted"`). See `residuals.simplexregression` for available options.
`nsim`	Number of simulations for envelope construction (default: `100`).
`level`	Confidence level for envelope bounds (default: `0.95`).
`seed`	Integer setting the random seed for reproducibility (default: `1987`).
`...`	Additional graphical parameters.

Details

The envelope is based on the following steps:

1. Simulate `nsim` response vectors from the fitted model using its estimated mean and dispersion parameter;
2. Refitting the model to each simulated dataset;
3. Computing absolute residuals and their order statistics;
4. Obtaining envelope bounds from empirical quantiles.

Simulated datasets whose refitted model fails to converge are discarded and resampled, up to `5 * nsim` total attempts. If `nsim` converged fits cannot be obtained within this limit, the function stops with an error.

Points outside the envelope may indicate model inadequacy.

Value

Called for its side effects (half-normal plot with simulated envelope). Returns `NULL` invisibly.

See Also

`residuals.simplexregression`, `plot.simplexregression`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)


halfnormal.plot(fit, seed = 2008)
```

`local.influence` *Local Influence for Simplex Regression Models*

Description

Computes local influence measures under case-weight and response perturbation schemes for simplex regression models with parametric or fixed mean link function.

Usage

```
local.influence(
  model,
  scheme = c("case.weight", "response"),
  parameter = c("theta", "beta", "gamma"),
  type = c("Ci", "dmax"),
  plot = FALSE,
  threshold = NULL,
  label.pos = 3,
  plot.type = NULL,
  ...
)
```

Arguments

model	An object of class "simplexregression".
scheme	Character string specifying the perturbation scheme: "case.weight" or "response".
parameter	Character string indicating the parameter block: "theta" (default, all parameters), "beta" (mean submodel), or "gamma" (dispersion submodel).
type	Character string specifying the influence measure to compute: "Ci" (default, total local influence / normal curvature) or "dmax" (maximum influence direction).
plot	Logical; if TRUE, produces an index plot of the selected measure. Default is FALSE.
threshold	Numeric threshold for identifying influential observations. If NULL (default), no observations are highlighted.
label.pos	Position(s) for outlier labels in plot. Can be a single value (applied to all labels) or a vector. Values: 1 = below, 2 = left, 3 = above, 4 = right.
plot.type	Character string controlling the plot style when plot = TRUE. If NULL (default), uses "h" (vertical lines). Passed to the type argument of plot().
...	Additional graphical parameters passed to plot().

Details

Measures local influence based on the curvature of the log-likelihood surface under small perturbations. Two perturbation schemes are implemented:

- **Case-weight**: Perturbs observation weights;
- **Response perturbation**: Perturbs response values.

The index plot of dmax can be used to detect observations that are jointly influential for parameters. The index plot of the normal curvature Ci can be used to detect observations that are individually influential for parameters.

Computing these measures requires inverting the observed information matrix L and certain of its sub-blocks. If L or a relevant sub-block is singular (e.g., due to near-collinear predictors in the mean or dispersion submodel), the function stops with an informative error rather than propagating R's raw solve error.

Value

If `plot = FALSE` (default), a list containing:

- `dmax.beta`: Maximum influence direction for mean parameters;
- `dmax.gamma`: Maximum influence direction for dispersion parameters;
- `dmax.theta`: Maximum influence direction for all parameters;
- `Ci.beta`: Total local influence for mean parameters;
- `Ci.gamma`: Total local influence for dispersion parameters;
- `Ci.theta`: Total local influence for all parameters.

If `plot = TRUE`, the same list is returned invisibly.

References

Espinheira, P. L. and Silva, A. O. (2020). Residual and influence analysis to a general class of simplex regression. *TEST*, **29**, 523–552. doi:10.1007/s11749019006653

See Also

`hatvalues.simplexregression`, `cooks.distance.simplexregression`, `gleverage.simplexregression`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

# Local influence under case-weight perturbation — return results
infl_cw <- local.influence(fit, scheme = "case.weight")

# Plot Ci for beta directly from the function call
local.influence(fit, scheme = "case.weight",
                parameter = "beta", type = "Ci", plot = TRUE)

# Plot dmax for all parameters under response perturbation
local.influence(fit, scheme = "response",
                parameter = "theta", type = "dmax", plot = TRUE)
```

`parametric_mean_links` *Parametric Mean Link Functions and Derivatives*

Description

Provides the parametric mean link functions, their inverses, and derivatives for the simplex regression models. Two parametric link types are supported: `"plogit1"` and `"plogit2"`.

Usage

```
parametric_mean_link(mu, lambda, type = c("plogit1", "plogit2"))

parametric_mean_link_inv(eta, lambda, type = c("plogit1", "plogit2"))

parametric_mean_link_deriv1(mu, lambda, type = c("plogit1", "plogit2"))

parametric_mean_link_inv_deriv1(eta, lambda, type = c("plogit1", "plogit2"))

parametric_mean_link_deriv2(mu, lambda, type = c("plogit1", "plogit2"))
```

Arguments

`mu`	Mean parameter (numeric vector, $0 < \mu < 1$).
`lambda`	Power parameter (numeric scalar, $\lambda > 0$).
`type`	Type of link function: `"plogit1"` or `"plogit2"`.
`eta`	Linear predictor of mean (numeric vector).

Details

Two parametric mean link functions are available, as proposed by Justino and Cribari-Neto (2026):

- Parametric logit type 1 (`"plogit1"`): $g(\mu; \lambda) = \log((1-\mu)^{-\lambda} - 1)$;
- Parametric logit type 2 (`"plogit2"`): $g(\mu; \lambda) = \log(\mu^{\lambda}/(1-\mu^{\lambda}))$.

Their inverses and derivatives with respect to μ are also implemented.

Value

Numeric vector with transformed values.

References

Justino, M. E. C. and Cribari-Neto, F. (2026). Simplex regression with a flexible logit link: Inference and application to cross-country impunity data. *Applied Mathematical Modelling*, **154**, 116713. doi:10.1016/j.apm.2025.116713

See Also

`simplexreg`, `fixed_mean_links`, `dispersion_links`.

Examples

```
parametric_mean_link(0.2, lambda = 1.2, type = "plogit2")
parametric_mean_link(c(0.2, 0.5, 0.8), lambda = 1.5, type = "plogit1")
parametric_mean_link_inv(0, lambda = 1, type = "plogit2")
parametric_mean_link_deriv1(0.5, lambda = 1, type = "plogit2")
parametric_mean_link_inv_deriv1(0, lambda = 1, type = "plogit2")
parametric_mean_link_deriv2(0.5, lambda = 1, type = "plogit2")
```

`penalized.ic` *Penalized Information Criteria for Simplex Regression Model Selection*

Description

Implements the Akaike, Schwarz, and Hannan–Quinn information criteria with a penalty term for selecting among competing simplex regression models with parametric mean link functions.

Usage

```
penalized.ic(
  ...,
  kappa = 0.1,
  verbose = TRUE,
  digits = max(3, getOption("digits") - 3)
)
```

Arguments

`...`	One or more objects of class `"simplexregression"` fitted with a parametric mean link functions (`"plogit1"` or `"plogit2"`).
`kappa`	A numeric value controlling the additional penalty for the link mean parameter, $\kappa \geq 0$. Default is `0.1`.
`verbose`	Logical. If `TRUE` (default), prints the criteria values. If `FALSE`, returns results silently.
`digits`	Integer specifying the number of decimal places for output. Default is `max(3, getOption("digits") - 3)`.

Details

The penalized information criteria, as proposed by Justino and Cribari-Neto (2026), extend the classical Akaike, Schwarz and Hannan–Quinn criteria with an additional penalty term for the link parameter λ:

$$AIC^{(\lambda)} = -2\ell + (2 + \kappa\,|\log(\lambda)|)r$$

$$BIC^{(\lambda)} = -2\ell + (\log(n) + \kappa\,|\log(\lambda)|)r$$

$$HQIC^{(\lambda)} = -2\ell + (2\log(\log(n)) + \kappa\,|\log(\lambda)|)r$$

where:

- ℓ denotes the maximized log-likelihood function;
- $\kappa \geq 0$ controls the additional penalty associated with the link parameter;
- λ is the parameter of the parametric mean link function;
- r indicate the dimension of their parameter vector;
- n is the number of observations.

Important: These penalized versions of the criteria should only be used when the candidate models employ a parametric link function in the mean submodel (use `kappa = 0.1`). When candidate models include specifications with fixed link functions, the standard unpenalized versions of these criteria should be applied instead (use `kappa = 0`).

Value

A data frame with rows named after the candidate models and four columns:

`df` Number of estimated parameters.

`AICc` Penalized AIC value.

`BICc` Penalized BIC value.

`HQICc` Penalized HQIC value.

When `verbose = TRUE`, the results are also printed to the console and the data frame is returned invisibly. When `verbose = FALSE`, the data frame is returned visibly without printing. When `kappa = 0`, the columns are named `AIC`, `BIC`, and `HQIC` instead of `AICc`, `BICc`, and `HQICc`.

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. *Akadémiai Kiadó*, 267–281.

Hannan, E. J. and Quinn, B. G. (1979). The determination of the order of an autoregression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **41**(2), 190–195. doi:10.1111/j.25176161.1979.tb01072.x

Justino, M. E. C. and Cribari-Neto, F. (2026). Simplex regression with a flexible logit link: Inference and application to cross-country impunity data. *Applied Mathematical Modelling*, **154**, 116713. doi:10.1016/j.apm.2025.116713

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136

See Also

`penalized.ss`

Examples

```
# Simulate data
set.seed(2026)
n <- 100
x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
z1 <- runif(n, 0, 1)
mu <- parametric_mean_link_inv(0.6 - 2*x1 - 1.5*x2, 0.5, "plogit1")
sigma2 <- dispersion_link_inv(-2 - 2.5*z1, "log")
y <- rsimplex(n, mu, sigma2)
data <- data.frame(y = y, x1 = x1, x2 = x2, z1 = z1)

# Fit two models with parametric mean link functions
```

```
fit1 <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "plogit1")
fit2 <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "plogit2")

# Compute penalized criteria
penalized.ic(fit1, fit2)
```

`penalized.ss` *Scout Score Criterion for Simplex Regression Model Selection*

Description

Implements the Scout Score (SS) criterion for selecting among competing simplex regression models with parametric and fixed mean link functions.

Usage

```
penalized.ss(
  ...,
  kappa = 0.1,
  verbose = TRUE,
  digits = max(3, getOption("digits") - 3)
)
```

Arguments

`...`	Two or more objects of class `"simplexregression"` to be compared.
`kappa`	A numeric value controlling the additional penalty for the link mean parameter, $\kappa \geq 0$. Default is `0.1`. Use `kappa = 0` for standard Scout Score.
`verbose`	Logical. If `TRUE` (default), prints the SS values for all models and the selected model. If `FALSE`, returns results silently.
`digits`	Integer specifying the number of decimal places for output. Default is `max(3, getOption("digits") - 3)`.

Details

The Scout Score criterion, originally proposed by Costa et al. (2024) for selecting link functions in the βARMA (beta autoregressive moving average) model, extends Vuong's test statistic to compare $M \geq 2$ competing non-nested models using their individual log-likelihood contributions.

For each candidate model $j \in 1, \ldots, M$, the Scout Score is defined as:

$$SS_j = 1 - M + \sum_{k=1, k \neq j}^{M} (1 + \dot{\Delta}_{jk})^2,$$

where $\dot{\Delta}_{jk} = \max\{0, \Delta_{jk}\}$ and Δ_{jk} is Vuong's (1989) test statistic comparing models j and k, penalized by a term δ_{jk} that combines the difference in parameter-vector dimensions with, when

applicable, a link-complexity penalty controlled by kappa. The model with the highest Scout Score is selected as the most adequate. For the full derivation of Δ_{jk}, δ_{jk}, and the rationale behind the link-complexity penalty, see Justino and Cribari-Neto (2026) and Costa et al. (2024).

When at least one of the candidate models does not use a parametric mean link function, κ is internally set to `0` (see **Important** below), so that δ_{jk} reduces to the classical dimension penalty with no link-complexity term.

The model with the highest Scout Score is selected as the most adequate.

Important: The penalty term δ_{jk} is only applied when *all* candidate models employ a parametric mean link function, in which case kappa (default `0.1`) controls the additional penalty. In any other case — whether all candidate models use fixed mean links, or the set of candidate models mixes parametric and fixed mean links — the penalty is disabled and the standard, unpenalized Scout Score is computed for all models (kappa is internally reset to `0`). If the user explicitly requested kappa > `0` in either of these situations, a warning is issued; if kappa was left at its default value, no warning is issued, since falling back to the standard Scout Score is the expected behavior.

Note: all candidate models must be fitted to the same response vector y; the function verifies this and stops with an error if the response vectors differ.

Value

A data frame with rows named after the candidate models and two columns:

`df` Number of estimated parameters in each model.

`SS` Scout Score value. The model with the highest value is the selected one.

When `verbose = TRUE`, the selected model is also printed to the console. The data frame is returned invisibly in this case, and visibly when `verbose = FALSE`.

References

Costa, E., Cribari-Neto, F. and Scher, V. T. (2024). Test inferences and link function selection in dynamic beta modeling of seasonal hydro-environmental time series with temporary abnormal regimes. *Journal of Hydrology*, **638**, 131489. doi:10.1016/j.jhydrol.2024.131489

Justino, M. E. C. and Cribari-Neto, F. (2026). Simplex regression with a flexible logit link: Inference and application to cross-country impunity data. *Applied Mathematical Modelling*, **154**, 116713. doi:10.1016/j.apm.2025.116713

Vuong, Q. H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, **57**(2), 307–333. doi:10.2307/1912557

See Also

penalized.ic

Examples

```
# Simulate data
set.seed(2026)
n <- 100
x1 <- runif(n, 0, 1)
```

```
x2 <- runif(n, 0, 1)
z1 <- runif(n, 0, 1)
mu <- parametric_mean_link_inv(0.6 - 2*x1 - 1.5*x2, 0.5, "plogit1")
sigma2 <- dispersion_link_inv(-2 - 2.5*z1, "log")
y <- rsimplex(n, mu, sigma2)
data <- data.frame(y = y, x1 = x1, x2 = x2, z1 = z1)

# Fit models
fit1 <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "plogit1")
fit2 <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "plogit2")
fit3 <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "logit")
fit4 <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "probit")

# Compare models with verbose output
result <- penalized.ss(fit1, fit2, kappa = 0.1)

# Compare models silently
result <- penalized.ss(fit1, fit2, kappa = 0.1, verbose = FALSE)

# Use standard Scout Score (no parametric link penalty)
result <- penalized.ss(fit1, fit2, fit3, fit4, kappa = 0)
```

`plot.simplexregression` *Diagnostic Plots for Simplex Regression Models*

Description

Produces diagnostic plots for fitted simplex regression models with parametric or fixed mean link function.

Usage

```
## S3 method for class 'simplexregression'
plot(
  x,
  which = 1:7,
  type = c("quantile", "pearson", "deviance", "standardized", "weighted", "variance",
    "biasvariance", "score", "dualscore", "response"),
  ask = prod(par("mfcol")) < length(which) && interactive(),
  threshold = NULL,
  label.pos = 3,
  plot.type = NULL,
  ...
)
```

Arguments

x	An object of class "simplexregression".
which	Numeric vector indicating which plots to display (1:7).
type	Character string specifying the residual type (default: "quantile"). See residuals.simplexregression for available options.
ask	Logical; if TRUE, the user is asked before each plot. Default is TRUE when multiple plots are requested.
threshold	Numeric threshold for identifying influential or outlying observations in plots 1–3 (residuals plots), 6 (Cook's distance), and 7 (generalized leverage). If NULL (default), no observations are highlighted.
label.pos	Position(s) for outlier labels in plots 1–3, 6, and 7. Can be a single value (applied to all labels) or a vector. Values: 1 = below, 2 = left, 3 = above, 4 = right. Default is 3 (above). See text for details.
plot.type	Controls the plot symbol/type for scatter plots and index plots. If NULL (default), uses pch = 1 (open circles) for residual plots (which = 1–4) and type = "h" (vertical lines) for Cook's distance (which = 6) and generalized leverage plots (which = 7). Otherwise, the value is passed directly to pch (for scatter plots) or type (for index plots).
...	Additional graphical parameters.

Details

Seven diagnostic plots are available:

- Residuals vs observation index (which = 1): Identifies outliers and temporal patterns;
- Residuals vs fitted values (which = 2): Checks for heteroscedasticity and patterns;
- Residuals vs linear predictor (which = 3): Evaluates link function adequacy;
- Observed vs fitted values (which = 4): Assesses overall model fit;
- Normal Q–Q plot (which = 5): Evaluates residual normality (especially useful for quantile residuals);
- Cook's distance vs indices of observations (which = 6): Identifies influential observations;
- Generalized leverage vs indices of observations (which = 7): Identifies influential observations.

Value

Called for its side effects (diagnostic plots). Returns NULL invisibly.

See Also

residuals.simplexregression, cooks.distance.simplexregression, halfnormal.plot.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

# Display all diagnostic plots
oldpar <- par(mfrow = c(3, 3))
plot(fit, which = 1:7)
par(oldpar)
```

`press` *PRESS-Based $P2$ Statistics for Simplex Regression*

Description

Computes the PRESS (Predicted Residual Error Sum of Squares) statistic and the associated P^2 and adjusted P^2 measures for simplex regression models with parametric or fixed mean link function, as proposed by Espinheira and Silva (2026).

Usage

```
press(..., type = c("standardized", "biasvariance"))
```

Arguments

`...`	One or more objects of class `"simplexregression"`.
`type`	Character string specifying the type of residual to use. Options are `"standardized"` (default) or `"biasvariance"` (see `residuals.simplexregression`).

Details

The PRESS statistic for the simplex regression model is given by:

$$\text{PRESS} = \sum_{i=1}^{n} \left(\frac{r_i}{1 - h_{ii}} \right)^2,$$

where r_i denotes the residual for observation i and $\hat{h}_{ii}$ is the i-th diagonal element of the hat matrix (see `hatvalues.simplexregression`).

The P^2 statistic is a cross-validation analog of R^2, defined for the simplex regression mode as:

$$P^2 = 1 - \frac{\text{PRESS}}{\left(\frac{n}{n-r}\right)^2 \text{SST}},$$

where r is the number of estimated parameters, $\text{SST} = \sum_{i=1}^{n} (\check{y}_i - \bar{\check{\boldsymbol{y}}})^2$, with $\bar{\check{y}}$ is the mean of the transformed fitted values $\check{y}_i$ defined by Espinheira and Silva (2026).

The adjusted P^2 is given by:

$$P_c^2 = 1 - (1 - P^2)\frac{n-1}{n-r}.$$

The `type` argument controls which residuals r_i are used in the PRESS computation. Only `"standardized"` and `"biasvariance"` residuals are supported, as these are the residual types for which the PRESS-based cross-validation analog is defined in Espinheira and Silva (2026)

Values of P^2 and P_c^2 closer to 1 indicate better predictive performance of the model. Since PRESS is nonnegative, both measures are bounded above by 1. Hence,

$$P^2, P_c^2 \in (-\infty, 1].$$

Value

When a single model is provided, a named numeric vector with components `P2`, `P2_c`, and `PRESS`. When multiple models are provided, a data frame with one row per model and columns `P2`, `P2_c`, and `PRESS`.

References

Espinheira, P. L. and Silva, A. O. (2020). Residual and influence analysis to a general class of simplex regression. *TEST*, **29**, 523–552. doi:10.1007/s11749019006653

Espinheira, P. L. and Silva, A. O. (2026). Prediction in the nonlinear simplex model. *International Journal of Data Science and Analytics*, **22**, 161. doi:10.1007/s41060026011149

See Also

`simplexreg`, `residuals.simplexregression`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

fit1 <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills, link.mu = "loglog")
# Single model
press(fit)

# Comparing multiple models
press(fit, fit1)

# Using bias-variance residuals
press(fit, fit1, type = "biasvariance")
```

r2 *Pseudo-R2 Measures for Simplex Regression*

Description

Extracts the Ferrari and Cribari-Neto pseudo-R^2 (R^2_{FC}), the likelihood-ratio pseudo-R^2 (R^2_N), the conventional coefficient of determination (R^2), and computes their finite-sample corrections. The corrections for R^2_{FC} and R^2_N follow Bayer and Cribari-Neto (2017), whereas the correction for the conventional R^2 follows Hu and Shao (2008).

Usage

```
r2(..., alpha1 = 0.4, alpha2 = 1, alpha3 = c("1", "log"))
```

Arguments

`...`	One or more objects of class `"simplexregression"` to be evaluated.
`alpha1`	Numeric, $\alpha_1 \in [0, 1]$ controlling the relative penalty weight given to the mean and dispersion submodels in the weighted correction $R^2_{Nw_c}$. Default is `0.4`, as recommended by Bayer and Cribari-Neto (2017).
`alpha2`	Numeric, $\alpha_2 > 0$, controlling the penalization intensity in $R^2_{Nw_c}$. Default is `1`, as recommended by Bayer and Cribari-Neto (2017).
`alpha3`	Penalization constant for the correction of R^2_{HS}. One of `"1"` (default, $\alpha_3 = 1$) or `"log"` ($\alpha_3 = \log(n)$).

Details

R^2_{FC} is the Ferrari and Cribari-Neto (2004) pseudo-R2, defined as the squared correlation between the fitted mean linear predictor and the link-transformed response, i.e.,

$$R^2_{FC} = \mathrm{corr}^2(\hat{\boldsymbol{\eta}}_{\mathbf{1}}, g(\boldsymbol{y}))\,,$$

where $\hat{\boldsymbol{\eta}}_{\mathbf{1}}$ denotes the vector of fitted mean linear predictors, $g(\cdot)$ is the mean link function, and $\boldsymbol{y}$ is the vector of observed values of the response variable.

R^2_N is a likelihood-ratio-based pseudo-R2 (Nagelkerke, 1991), defined as

$$R^2_N = 1 - \left(\frac{L_{null}}{L_{fit}}\right)^{2/n},$$

where L_{null} and L_{fit} are the maximized likelihoods of the null (intercept-only) and fitted models, respectively.

The conventional coefficient of determination is

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{\mu}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

where $\bar{y}$ is the sample mean of the responses.

Corrections: Both finite-sample corrections implemented here for R^2_{FC} and R^2_N were proposed by Bayer and Cribari-Neto (2017) for beta regression with varying precision and are used here in their direct simplex regression analogue.

The simple correction, a function of the total number of estimated parameters r only, is applied to both R^2_{FC} and R^2_N:

$$R^2_{FC_c} = 1 - (1 - R^2_{FC})\frac{n-1}{n-r} \qquad \text{and} \qquad R^2_{N_c} = 1 - (1 - R^2_N)\frac{n-1}{n-r}.$$

A second, weighted correction is defined for R^2_N only, penalizing the mean and dispersion submodels asymmetrically through `alpha1` and controlling penalization intensity through `alpha2`:

$$R^2_{Nw_c} = 1 - (1 - R^2_N)\left(\frac{n-1}{n-(1+\alpha_1)p-(1-\alpha_1)q}\right)^{\alpha 2},$$

where p and q being the number of parameters in the mean and dispersion submodels, respectively. Setting `alpha1 = 0` and `alpha2 = 1` reduces $R^2_{Nw_c}$ to the simple $R^2_{N_c}$ above. Bayer and Cribari-Neto (2017) recommend `alpha1 = 0.4` and `alpha2 = 1` as sensible defaults; both arguments are exposed here for users who wish to specify different values.

Finally, the conventional coefficient of determination admits the finite-sample correction proposed by Hu and Shao (2008):

$$R^2_{HS} = 1 - \frac{n-1}{n-\alpha_3 r}\frac{\sum_{i=1}^n (y_i - \hat{\mu}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}.$$

where α_3 is a penalization constant. When `alpha_3 = "1"` (default), $\alpha_3 = 1$, R^2_{HS} reduces to the modified R2 of Mittlböck and Schemper (2002). When `alpha_3 = "log"`, $\alpha_3 = \log(n)$, as evaluated by Hu and Shao (2008).

The measures R^2_{FC} and R^2_N take values in $[0, 1]$. The conventional coefficient of determination R^2, as well as the corrected measures $R^2_{FC_c}$, $R^2_{N_c}$, $R^2_{Nw_c}$, and R^2_{HS}, take values in $(-\infty, 1]$. Larger values indicate better model fit.

Value

When a single model is provided, a named numeric vector with components `R2_FC`, `R2_FC_c`, `R2_N`, `R2_N_c`, `R2_Nw_c`, `R2`, and `R2_HS`. When multiple models are provided, a data frame with one row per model.

References

Bayer, F. M. and Cribari-Neto, F. (2017). Model selection criteria in beta regression with varying dispersion. *Communications in Statistics - Simulation and Computation*, **46**(1), 729–746. doi:10.1080/03610918.2014.977918

Ferrari, S. L. P. and Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, **31**(7), 799–815. doi:10.1080/0266476042000214501

Hu, B. and Shao, J. (2008). Generalized linear model selection using R2. *Journal of Statistical Planning and Inference*, **138**(12), 3705–3712. doi:10.1016/j.jspi.2007.12.009

Mittlböck, M. and Schemper, M. (2002). Explained variation for logistic regression – small sample adjustments, confidence intervals and other issues. *Statistics in Medicine*, **21**(23), 3547–3562. doi:10.1002/15214036(200204)44:3<263::AIDBIMJ263>3.0.CO;27

Nagelkerke, N. J. D. (1991). A note on a general definition of the coefficient of determination. *Biometrika*, **78**(3), 691–692. doi:10.1093/biomet/78.3.691

See Also

simplexreg, press

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit1 <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                   data = ReadingSkills)

# Single model, default alpha1, alpha2 and alpha3
r2(fit1)

# Comparing multiple models
fit2 <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                   data = ReadingSkills, link.mu = "loglog")
r2(fit1, fit2)

# Custom alpha1/alpha2 for the weighted correction, and alpha3 = log(n)
r2(fit1, fit2, alpha1 = 0.6, alpha2 = 1.5, alpha3 = "log")
```

ReadingSkills *Reading Accuracy in Dyslexic and Non-Dyslexic Children*

Description

This dataset examines the relationship between non-verbal IQ and reading accuracy in children diagnosed with dyslexia and in typical readers. The reading scores are proportions bounded in the open interval (0, 1), making them suitable for beta regression and simplex regression analysis.

Usage

```
data(ReadingSkills)
```

Format

A data frame with 44 observations and 4 variables:

accuracy Numeric. Reading score transformed to the open (0, 1) interval. Values originally equal to 1 were replaced with 0.99. Suitable for standard beta regression.

dyslexia Factor. Indicates whether the child has dyslexia (levels: `"no"`, `"yes"`). Note that sum contrasts are typically used instead of treatment contrasts in beta regression analyses of this data.

iq Numeric. Non-verbal intelligence quotient, transformed to z-scores (mean = 0, standard deviation = 1).

accuracy1 Numeric. Unrestricted reading score in the [0, 1] interval. This version preserves the original maximum value of 1 and can be used with extended–support beta mixture regression models.

Details

The data were originally collected by Pammer and Kevan (2004) and later analyzed by Smithson and Verkuilen (2006) to demonstrate beta regression.

The transformation procedure for `accuracy` was as follows:

1. The original test scores were scaled to the [0, 1] interval using the minimum and maximum possible scores in the reading test, resulting in `accuracy1`.
2. To avoid boundary values (0 and 1) that are problematic for standard beta regression, all observations with value 1 were replaced with 0.99, creating the `accuracy` variable.

The unrestricted `accuracy1` variable can be analyzed using extended–support beta regression methods (Kosmidis & Zeileis, 2025), which naturally accommodate boundary observations.

Source

Pammer, K. and Kevan, A. (2007). The contribution of visual sensitivity, phonological processing and non-verbal IQ to children's reading. *Scientific Studies of Reading*, **11**(1), 33–53. doi:10.1080/10888430709336633

Smithson, M. and Verkuilen, J. (2006). A Better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. *Psychological Methods*, **11**(1), 54–71. doi:10.1037/1082989X.11.1.54

References

Cribari-Neto, F. and Zeileis, A. (2010). Beta regression in R. *Journal of Statistical Software*, **34**(2), 1–24. doi:10.18637/jss.v034.i02

Kosmidis, I. and Zeileis, A. (2025). Extended-support beta regression for [0, 1] responses. *Journal of the Royal Statistical Society C*, **75**(1), 139–157. doi:10.1093/jrsssc/qlaf039

Examples

```
# Load the data
data(ReadingSkills)

# Quick overview
head(ReadingSkills)
str(ReadingSkills)

# Summary statistics by dyslexia status
aggregate(accuracy ~ dyslexia, data = ReadingSkills, summary)
aggregate(iq ~ dyslexia, data = ReadingSkills, summary)

# Visualize the relationship between IQ and reading accuracy
plot(accuracy ~ iq, data = ReadingSkills,
     col = c(4, 2)[dyslexia], pch = 19,
```

```
     main = "Reading Accuracy vs. IQ by Dyslexia Status",
     xlab = "IQ (z-scored)", ylab = "Reading Accuracy")
legend("topleft", legend = c("Non-dyslexic", "Dyslexic"),
       col = c(4, 2), pch = 19, bty = "n")

# Check for boundary values
table(ReadingSkills$accuracy == 0.99)  # Values replaced from 1
table(ReadingSkills$accuracy1 == 1)    # Original boundary values
```

`RelativeHumidity` *Monthly Relative Humidity Data*

Description

Monthly meteorological data including relative humidity (rh), temperature, insolation, precipitation, and wind speed for the meteorological station of Brasília, Brazil (2000-2025), obtained from the National Institute of Meteorology (INMET).

The dataset contains two missing observations in the variables `ins` and `pre`, corresponding to September 2020 and November 2025. To address these missing values, imputation was performed via seasonal interpolation using the **imputeTS** package, with the imputed versions stored as `ins2` and `pre2`.

Usage

```
data(RelativeHumidity)
```

Format

A data frame with 312 observations and 11 variables:

date Date. Last day of the reference month for the monthly measurements (YYYY-MM-DD).

rh Numeric. Monthly mean relative humidity, originally measured in percent and rescaled to the unit interval (0, 1).

ins Numeric. Monthly total insolation (in hours), contains 2 missing values.

ins2 Numeric. Monthly total insolation (in hours) with missing values imputed via seasonal interpolation.

pre Numeric. Monthly total precipitation (in mm), contains 2 missing values.

pre2 Numeric. Monthly total precipitation (in mm) with missing values imputed via seasonal interpolation.

cld Numeric. Monthly mean cloudiness (in tenths).

ap Numeric. Monthly mean atmospheric pressure (in hPa).

tmax Numeric. Monthly mean of daily maximum temperature (in degrees Celsius).

ws Numeric. Monthly mean wind speed (in m/s).

wd Numeric. Monthly predominant wind direction, coded by INMET/BDMEP in tens of degrees (e.g., 9 = 90 degrees, East; 32 = 320 degrees, Northwest). A value of 0 indicates calm conditions (no wind or undefined direction). Multiply by 10 to obtain the direction in degrees (0-360).

Source

Instituto Nacional de Meteorologia (INMET). Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP), Estação de Brasília (https://bdmep.inmet.gov.br). Data accessed in 2026.

Examples

```
# Load the dataset
data(RelativeHumidity)

# View first rows
head(RelativeHumidity)

# Check structure
str(RelativeHumidity)

# Insolation with and without imputation
plot(RelativeHumidity$ins, type = "l", col = "red",
     ylab = "Insolation", main = "Missing values visible as gaps")
points(RelativeHumidity$ins2, type = "l", col = "blue")
```

---

resettest *RESET Test for Simplex Regression*

Description

Performs the Ramsey's RESET misspecification test in simplex regression models.

Usage

```
resettest(model, dispersion = TRUE, power = 2, type = c("lp", "fitted"))
```

Arguments

model	An object of class "simplexregression".
dispersion	Logical. If TRUE, includes the augmented terms in the dispersion submodel as well. Default is TRUE.
power	Integer vector specifying which powers of the fitted mean linear predictor (or fitted mean values) to include as additional regressors. Default is 2 (squared term only). Use power = 2:3 to include both squared and cubic terms, following the convention of lmtest::resettest.
type	Character string specifying the base for the augmented terms. "lp" (default) uses the fitted mean linear predictor $\hat{\boldsymbol{\eta}}_1$; "fitted" uses the fitted mean values $\hat{\boldsymbol{\mu}}$ on the (0, 1) scale.

Details

The RESET test augments the original model by adding powers of the fitted mean linear predictor (or fitted mean values) as additional covariates. Under the null hypothesis of correct functional form, these additional terms should not be significant.

Under H0, the likelihood ratio statistic is asymptotically distributed as chi-squared with degrees of freedom equal to the number of augmented terms added (i.e., `length(power)` if `dispersion = FALSE`, or `2 * length(power)` if `dispersion = TRUE`).

If `dispersion = TRUE`, the augmented terms are added to both the mean and dispersion submodels. If `FALSE`, they are only added to the mean submodel.

Value

An object of class `"htest"` containing:

- `statistic`: The likelihood ratio test statistic,
- `parameter`: Degrees of freedom,
- `p.value`: The p-value of the test,
- `method`: Description of the test,
- `data.name`: Model formula.

References

Ramsey, J. B. (1969). Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society: Series B*, **31**(2), 350–371. doi:10.1111/j.2517-6161.1969.tb00796.x

See Also

simplexreg.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

# Default: squared linear predictor in both submodels
resettest(fit)

# RESET test only for mean submodel
resettest(fit, dispersion = FALSE)

# Include squared and cubic terms
resettest(fit, power = 2:3)

# Use fitted values instead of linear predictor
resettest(fit, type = "fitted")
```

`residuals.simplexregression` *Residuals for Simplex Regression Models*

Description

Extracts various types of residuals for diagnostic analysis in simplex regression models with parametric or fixed mean link functions.

Usage

```
## S3 method for class 'simplexregression'
residuals(
  object,
 type = c("quantile", "pearson", "deviance", "standardized", "weighted", "variance",
    "biasvariance", "score", "dualscore", "response"),
  ...
)
```

Arguments

`object`	An object of class `"simplexregression"`.
`type`	Character string specifying the type of residual to extract. Options: `"quantile"` (default), `"pearson"`, `"deviance"`, `"standardized"`, `"weighted"`, `"variance"`, `"biasvariance"`, `"score"`, `"dualscore"`, `"response"`.
`...`	Additional arguments (currently not used).

Details

Several types of residuals are available for model diagnostics:

Quantile residuals (`"quantile"`): Proposed by Dunn and Smyth (1996) as

$$r_i^Q = \Phi^{-1}(F(y_i; \hat{\mu}_i, \hat{\sigma}_i^2)),$$

where $\Phi(\cdot)$ is the standard normal CDF and $F(\cdot;\cdot)$ is the simplex CDF (see `psimplex`). Under correct model specification, these residuals are approximately standard normal and are therefore recommended for general diagnostic use.

Pearson residuals (`"pearson"`): Defined in McCullagh and Nelder (1989) as

$$r_i^P = \frac{y_i - \hat{\mu}_i}{\sqrt{\widehat{\mathrm{Var}}(y_i)}},$$

where $\widehat{\mathrm{Var}}(y_i)$ is the estimated variance of the response (see `variance.simplex`).

Deviance residuals (`"deviance"`): Defined in Jørgensen (1997, p. 115) as

$$r_i^D = \frac{y_i - \hat{\mu}_i}{\hat{\mu}_i(1-\hat{\mu}_i)\sqrt{y_i(1-y_i)}}.$$

Standardized residuals (`"standardized"`): Proposed by Espinheira and Silva (2020, Eq. 15) as

$$r_i^{\beta} = \frac{\hat{u}_i(y_i - \hat{\mu}_i)}{\sqrt{\hat{\sigma}_i^2 \hat{w}_i}},$$

where $\hat{w}_i = 3\hat{\sigma}_i^2[\hat{\mu}_i(1-\hat{\mu}_i)]^{-1} + [\hat{\mu}_i(1-\hat{\mu}_i)]^{-3}$ and $\hat{u}_i = \hat{d}_i[\hat{\mu}_i(1-\hat{\mu}_i)]^{-1} + [\hat{\mu}_i(1-\hat{\mu}_i)]^{-3}$, with $\hat{d}_i$ is the estimated unit deviance (see `dev.unit.simplex`).

Weighted residuals (`"weighted"`): Proposed by Espinheira and Silva (2020, Eq. 16) as

$$r_i^{\beta *} = \frac{\hat{u}_i(y_i - \hat{\mu}_i)}{\sqrt{\hat{\sigma}_i^2 \hat{w}_i(1-\hat{h}_{ii})}},$$

where $\hat{h}_{ii}$ are the diagonal elements of the hat matrix (see `hatvalues.simplexregression`). These residuals are recommended for simulated envelope plots.

Variance residuals (`"variance"`): Proposed by Espinheira et al. (2021, Eq. 7) as

$$r_i^{\gamma} = \frac{\hat{d}_i - \hat{\sigma}_i^2}{\hat{\sigma}_i^2\sqrt{2}}.$$

Bias–variance residuals (`"biasvariance"`): Proposed by Espinheira et al. (2021, Eq. 8) as

$$r_i^{\beta\gamma} = \frac{\hat{u}_i(y_i - \hat{\mu}_i) + \hat{a}_i}{\sqrt{\hat{\sigma}_i^2 \hat{w}_i + 1/(2\hat{\sigma}_i^4)}},$$

where $\hat{a}_i = -(2\hat{\sigma}_i^2)^{-1} + \hat{d}_i(2\hat{\sigma}_i^4)^{-1}$.

Score residuals (`"score"`): Defined in Jørgensen (1997, p. 115) as

$$r_i^{S} = \frac{(y_i - \hat{\mu}_i)(\hat{\mu}_i^2 + y_i - 2y_i\hat{\mu}_i)}{y_i(1-y_i)\hat{\mu}_i^{1.5}(1-\hat{\mu}_i)^{1.5}}.$$

Dual score residuals (`"dualscore"`): Defined in Jørgensen (1997, p. 115) as

$$r_i^{DS} = \frac{(y_i - \hat{\mu}_i)(y_i + \hat{\mu}_i - 2y_i\hat{\mu}_i)}{2\sqrt{y_i(1-y_i)}\hat{\mu}_i^2(1-\hat{\mu}_i)^2}.$$

Response residuals (`"response"`): Simple difference between observed and fitted values,

$$r_i^{R} = y_i - \hat{\mu}_i.$$

Recommendations: Quantile residuals are recommended for general model diagnostics due to their theoretical properties. Weighted residuals are particularly useful for constructing simulated envelope plots, as they account for both the variance structure and leverage effects.

Value

A numeric vector of residuals.

References

Dunn, P. K. and Smyth, G. K. (1996). Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, **5**(3), 236–244. doi:10.2307/1390802

Espinheira, P. L. and Silva, A. O. (2020). Residual and influence analysis to a general class of simplex regression. *TEST*, **29**, 523–552. doi:10.1007/s11749019006653

Espinheira, P. L.; Silva, L. C. M. and Cribari-Neto, F. (2021). Bias and variance residuals for machine learning nonlinear simplex regressions. *Expert Systems With Applications*, **185**, 115656. doi:10.1016/j.eswa.2021.115656

Jørgensen, B. (1997). *The Theory of Dispersion Models*. Chapman and Hall, London.

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. 2nd ed. Chapman and Hall, London.

See Also

`plot.simplexregression`, `halfnormal.plot`, `hatvalues.simplexregression`.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

# Compute different types of residuals
res_quantile <- residuals(fit, type = "quantile")
res_pearson <- residuals(fit, type = "pearson")
res_weighted <- residuals(fit, type = "weighted")
```

`scoretest` *Rao's Score Test for Simplex Regression with Parametric Mean Link Function*

Description

Performs a Rao's score test to test whether the link parameter λ equals 1, which corresponds to evaluating the null hypothesis that the mean link function is the standard logit.

Usage

```
scoretest(model, link.mu = c("plogit1", "plogit2"))
```

Arguments

`model`	An object of class `"simplexregression"` fitted with the fixed logit mean link (`link.mu = "logit"`)
`link.mu`	Character string specifying the parametric link function under the alternative hypothesis. Options are `"plogit1"` or `"plogit2"`.

Details

Given that the fixed logit link function is a particular case of the parametric logit link functions when $\lambda = 1$, it is possible to test whether the mean link function is logit by testing $H_0 : \lambda = 1$ against $H_1 : \lambda \neq 1$.

The score test statistic is computed from the component of the score vector associated with λ and the corresponding element of the inverse Fisher information matrix, both evaluated at the maximum likelihood estimator under the null hypothesis (i.e., with λ fixed at 1). For the full expression of the test statistic, see Justino and Cribari-Neto (2026).

Under regularity conditions, the null hypothesis, and when n is large, the test statistic follows a chi-squared distribution with 1 degree of freedom.

Value

An object of class `"htest"` containing:

- `statistic`: The score test statistic,
- `parameter`: Degrees of freedom (always 1),
- `p.value`: The p-value of the test,
- `method`: Description of the test,
- `data.name`: Description of the link comparison being tested (e.g., `"Logit vs plogit1"`)

References

Justino, M. E. C. and Cribari-Neto, F. (2026). Simplex regression with a flexible logit link: Inference and application to cross-country impunity data. *Applied Mathematical Modelling*, **154**, 116713. doi:10.1016/j.apm.2025.116713

Rao, C. R. (1948). Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, **44**(1), 50–57. doi:10.1017/S0305004100023987

See Also

`parametric_mean_link`

Examples

```
# Simulate data
set.seed(2026)
n <- 100
x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
z1 <- runif(n, 0, 1)
mu <- parametric_mean_link_inv(0.6 - 2*x1 - 1.5*x2, 0.5, "plogit1")
sigma2 <- dispersion_link_inv(-2 - 2.5*z1, "log")
y <- rsimplex(n, mu, sigma2)
data <- data.frame(y = y, x1 = x1, x2 = x2, z1 = z1)

# Fit model with logit
```

```
model <- simplexreg(y ~ x1 + x2 | z1, data = data, link.mu = "logit")

# Test if lambda = 1
scoretest(model, link.mu = "plogit1")
```

`simplexreg.control` *Control Parameters for Simplex Regression*

Description

Auxiliary function for controlling simplex regression fitting.

Usage

```
simplexreg.control(
  method = "BFGS",
  maxit = 5000,
  gradient = TRUE,
  hessian = FALSE,
  trace = FALSE,
  start = NULL,
  fsmaxit = 500,
  fstol = 1e-08,
  reltol = .Machine$double.eps^(0.5),
  ...
)
```

Arguments

`method`	Character string specifying the optimization method passed to `optim` (default: `"BFGS"`).
`maxit`	Integer specifying the maximum number of iterations for `optim` (default: `5000`).
`gradient`	Logical; use analytical gradient? (default: `TRUE`).
`hessian`	Logical; compute Hessian via `optim`? (default: `FALSE`).
`trace`	Logical; trace optimization? (default: `FALSE`).
`start`	An optional vector with starting values for all parameters.
`fsmaxit`	Integer specifying maximal number of additional Fisher scoring iterations (default: `500`).
`fstol`	Numeric tolerance for convergence in Fisher scoring (default: `1e-8`).
`reltol`	Relative convergence tolerance (default: `sqrt(.Machine$double.eps)`).
`...`	Additional parameters passed to `optim`.

Details

All parameters in simplexreg are estimated by maximum likelihood using optim with control options set in simplexreg.control. Most arguments are passed on directly to optim, and start controls how optim is called.

After the optim maximization, an additional Fisher scoring iteration can be performed to further enhance the result by moving the gradient even closer to zero. If fsmaxit is greater than zero, this additional optimization is performed and it converges if the threshold fstol is attained for the absolute value of the step size.

Starting values can be supplied via start or estimated by lm.wfit, using the link-transformed response. For parametric mean link functions ("plogit1", "plogit2"), the link parameter λ is jointly estimated with the regression coefficients. Covariances are derived analytically using the expected Fisher information matrix. The Fisher scoring uses analytical gradients and the expected information matrix to refine the maximum likelihood estimates obtained from optim.

The main parameters of interest are the coefficient vector β in the linear predictor of the mean submodel and the coefficient vector γ in the linear predictor of the dispersion submodel. For parametric links, the additional link parameter λ is also estimated and reported. The dispersion parameter σ^2 can be modeled either as constant (when the dispersion formula contains only an intercept) or as varying across observations through a linear predictor.

Value

A list of control parameters.

See Also

simplexreg

---

simplexreg.fit *Simplex Regression with Parametric or Fixed Mean Link*

Description

Fit simplex regression models for rates and proportions via maximum likelihood estimation, modeling both the mean (via parametric or fixed link function) and the dispersion parameter.

Usage

```
simplexreg(
  formula,
  data,
  subset,
  na.action,
  weights,
  offset,
 link.mu = c("logit", "probit", "loglog", "cloglog", "cauchit", "plogit1", "plogit2"),
  link.sigma2 = NULL,
```

```
  contrasts = NULL,
  control = simplexreg.control(...),
  model = TRUE,
  y = TRUE,
  x = FALSE,
  ...
)

simplexreg.fit(
  y,
  x,
  z,
  weights = NULL,
  offset = NULL,
 link.mu = c("logit", "probit", "loglog", "cloglog", "cauchit", "plogit1", "plogit2"),
  link.sigma2 = c("log", "sqrt", "identity"),
  x_names = NULL,
  z_names = NULL,
  control = simplexreg.control(...),
  ...
)
```

Arguments

formula	A two-part formula: y ~ x or y ~ x | 1 (mean submodel, constant dispersion), or y ~ x | z (submodels for both mean and dispersion).
data	A data frame containing the variables in formula.
subset	A specification of the rows/observations to be used: defaults to all.
na.action	An optional (name of a) function for treating missing values (NAs).
weights	An optional numeric vector of case weights.
offset	Optional numeric vector specifying a known component to be included in the linear predictor of the mean submodel during fitting. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
link.mu	Character specification of the link function in the mean submodel (parametric functions: "plogit1", "plogit2"; or fixed functions: "logit", "probit", "loglog", "cloglog", "cauchit").
link.sigma2	Character specification of the link function in the dispersion submodel ("log", "sqrt", "identity").
contrasts	An optional list. See the contrasts.arg argument of model.matrix.default.
control	A list of control arguments specified via simplexreg.control.
model, y, x	Logicals. If TRUE the corresponding components of the fit (model frame, response, model matrix) are returned. For simplexreg.fit, x should be a numeric regressor matrix and y should be the numeric response vector (with values in (0, 1)).
...	Additional arguments passed to simplexreg.control.

z	Design matrix for dispersion model (without intercept).
x_names	Column names for mean design matrix (includes intercept).
z_names	Column names for dispersion design matrix (includes intercept).

Details

Simplex regression, introduced by Song and Tan (2000) and extended by Song, Qiu and Tan (2004), is useful for modeling continuous response variables restricted to the unit interval (0, 1), such as rates and proportions. The model assumes that the dependent variable follows the simplex distribution, originally proposed by Barndorff-Nielsen and Jørgensen (1991), which is indexed by the mean μ and the dispersion parameter σ^2.

Similar to generalized linear models (GLMs), simplex regression relates the mean of the response variable and the dispersion parameter to their respective linear predictors through link functions. This package implements five fixed link functions (`"logit"`, `"probit"`, `"loglog"`, `"cloglog"`, `"cauchit"`) and two parametric link functions (`"plogit1"`, `"plogit2"`) for the mean submodel. For the dispersion submodel, the links `"log"`, `"sqrt"` and `"identity"` are supported.

The model is specified through a two-part formula separated by `|`. The left side contains the predictors for the mean submodel and the right side contains the predictors for the dispersion submodel:

- Mean submodel: $g(\mu_i, \lambda) = \boldsymbol{x}_i'\boldsymbol{\beta}$ (parametric link) or $g(\mu_i) = \boldsymbol{x}_i'\boldsymbol{\beta}$ (fixed link), where $g(\cdot)$ is the mean link function, λ is the extra shape parameter of the parametric link, $\boldsymbol{x}_i$ is the vector of covariates for the i-th observation in the mean submodel, and $\boldsymbol{\beta}$ is the corresponding vector of regression coefficients.
- Dispersion submodel: $h(\sigma_i^2) = \boldsymbol{z}_i'\boldsymbol{\gamma}$, where $h(\cdot)$ is the dispersion link function, $\boldsymbol{z}_i$ is the vector of covariates for the i-th observation in the dispersion submodel, and $\boldsymbol{\gamma}$ is the corresponding vector of regression coefficients.

Formula examples: `y ~ x1 + x2 | z1 + z2` (variable dispersion) or `y ~ x1 + x2` (constant dispersion). The link functions for both submodels are specified using `link.mu` and `link.sigma2`.

The parametric mean link functions include a parameter λ that is estimated along with other model parameters. Parameter estimation is performed via maximum likelihood using the `optim` function with analytical gradient and initial values obtained from an auxiliary linear regression of the transformed response. Subsequently, the `optim` result may be enhanced by an additional Fisher scoring iteration using analytical gradients and expected information. The Fisher scoring is just a refinement to move the gradients even closer to zero and can be disabled by setting `fsmaxit = 0` in the control arguments.

Methods for extracting and analyzing results are implemented for objects of class `"simplexregression"`, allowing the use of generic functions such as `summary`, `print`, `fitted`, `coef`, `formula`, `logLik`, `vcov`, `predict`, `terms`, `model.frame`, `model.matrix`, `plot`, `residuals`, `cooks.distance`, `gleverage`, `hatvalues`, `update`, `simulate`, `AIC`, `coeftest` (from the lmtest package), `bread` and `estfun` (from the sandwich package).

Value

An object of class `"simplexregression"`, i.e., a list with components as follows:

- `coefficients`: A list with elements `mean` and `dispersion`, containing the estimated regression coefficients $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\gamma}}$ of the mean and dispersion submodels, respectively. For parametric

mean link functions, the list also includes an additional element, `lambda`, containing the estimated link parameter $\hat{\lambda}$.

- `fitted.values`: a vector of fitted mean values,
- `optim`: a list containing `start` (initial values), `convergence` (convergence code), `counts` (number of iterations) and `method` (optimization method) from the optimization procedure,
- `scoring`: number of iterations from the optimization procedure via Fisher scoring,
- `mu.fv`: a vector of fitted mean values,
- `mu.lp`: a vector of fitted mean linear predictor,
- `mu.x`: design matrix for the mean model (with intercept),
- `mu.link`: character string specifying the mean link function,
- `mu.df`: degrees of freedom for the mean model,
- `sigma2.fv`: a vector of fitted dispersion values,
- `sigma2.lp`: a vector of fitted dispersion linear predictor,
- `sigma2.x`: design matrix for the dispersion model (with intercept),
- `sigma2.link`: character string specifying the dispersion link function,
- `sigma2.df`: degrees of freedom for the dispersion model,
- `lambda.fv`: estimated value of the parametric link function parameter (`NA` for mean fixed links),
- `df.residual`: residual degrees of freedom,
- `nobs`: number of observations,
- `loglik`: maximized log-likelihood value,
- `vcov`: variance-covariance matrix of the parameter estimates,
- `residuals`: a vector of quantile residuals,
- `AIC`, `BIC`, `HQIC`: Akaike, Schwarz, and Hannan-Quinn information criteria,
- `R2_N`, `R2_FC`: Nagelkerke, and Ferrari and Cribari-Neto pseudo R-squared measures,
- `zstat`: z-statistics for the coefficient tests,
- `pvalues`: p-values for the coefficient tests,
- `y`: the response vector,
- `x_names`: column names of the mean design matrix,
- `z_names`: column names of the dispersion design matrix,
- `control`: the control arguments passed to the `optim` call,
- `converged`: logical indicating successful convergence of `optim`,
- `call`: the original function call,
- `formula`: the original two-part formula,
- `formula_mean`: formula for the mean submodel,
- `formula_disp`: formula for the dispersion submodel,
- `terms`: a list with `mean` and `dispersion` terms objects,
- `weights`: the weights used in fitting (if any),
- `offset`: the offset used in fitting (if any),
- `na.action`: the na.action attribute from the model frame,
- `subset`: the subset used in fitting (if any),
- `model`: the full model frame.

References

Barndorff-Nielsen, O. E. and Jørgensen, B. (1991). Some parametric models on the simplex. *Journal of Multivariate Analysis*, **39**(1), 106–116. doi:10.1016/0047259X(91)90008P

Justino, M. E. C. and Cribari-Neto, F. (2026). Simplex regression with a flexible logit link: Inference and application to cross-country impunity data. *Applied Mathematical Modelling*, **154**, 116713. doi:10.1016/j.apm.2025.116713

Jørgensen, B. (1997). *The Theory of Dispersion Models*. Chapman and Hall, London.

Song, P. X.-K. and Tan, M. (2000). Marginal models for longitudinal continuous proportional data. *Biometrics*, **56**(2), 496–502. doi:10.1111/j.0006341X.2000.00496.x

Song, P. X.-K.; Qiu, Z. and Tan, M. (2004). Modelling heterogeneous dispersion in marginal models for longitudinal proportional data. *Biometrical Journal*, **46**(5), 540–553. doi:10.1002/bimj.200110052

Song, P. X.-K. (2009). Dispersion models in regression analysis. *Pakistan Journal of Statistics*, **25**(4), 529–551.

Zhang, P. and Qiu, Z. G. (2014). Regression analysis of proportional data using simplex distribution. *SCIENTIA SINICA Mathematica*, **44**(1), 89–104. doi:10.1360/012013200

See Also

`summary.simplexregression`, `predict.simplexregression`, `residuals.simplexregression`, `penalized.ic`, `penalized.ss`, `scoretest`

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)
summary(fit)
```

---

simplexreg.methods *Methods for simplexregression Objects*

Description

Methods for extracting information from fitted model objects of class `"simplexregression"`.

Usage

```
## S3 method for class 'simplexregression'
print(x, digits = max(3, getOption("digits") - 3), ...)

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

```
## S3 method for class 'summary.simplexregression'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'simplexregression'
coef(object, model = c("full", "mean", "dispersion"), ...)

## S3 method for class 'simplexregression'
vcov(object, model = c("full", "mean", "dispersion"), ...)

## S3 method for class 'simplexregression'
logLik(object, ...)

## S3 method for class 'simplexregression'
fitted(object, ...)

## S3 method for class 'simplexregression'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "dispersion"),
  ...
)

## S3 method for class 'simplexregression'
nobs(object, ...)

## S3 method for class 'simplexregression'
df.residual(object, ...)

## S3 method for class 'simplexregression'
deviance(object, ...)

## S3 method for class 'simplexregression'
formula(x, ...)

## S3 method for class 'simplexregression'
terms(x, model = c("mean", "dispersion"), ...)

## S3 method for class 'simplexregression'
model.frame(formula, ...)

## S3 method for class 'simplexregression'
model.matrix(object, model = c("mean", "dispersion"), ...)

## S3 method for class 'simplexregression'
update(object, formula., ..., evaluate = TRUE)

## S3 method for class 'simplexregression'
```

```
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'simplexregression'
AIC(object, ..., k = 2)

## S3 method for class 'simplexregression'
BIC(object, ...)

HQIC(object, ...)

## S3 method for class 'simplexregression'
HQIC(object, ...)

## S3 method for class 'simplexregression'
hatvalues(model, ...)

## S3 method for class 'simplexregression'
bread(x, ...)

## S3 method for class 'simplexregression'
estfun(x, ...)

## S3 method for class 'simplexregression'
coeftest(x, vcov. = NULL, df = Inf, ...)

## S3 method for class 'simplexregression'
lrtest(object, ...)
```

Arguments

digits	Number of digits for printing.
...	Additional arguments.
object, x	An object of class "simplexregression".
model	Character specifying for which component of the model coefficients/covariance should be extracted.
newdata	Optional data frame for prediction.
type	Character indicating type of predictions: fitted means of the response (default, "response"), corresponding linear predictor ("link") or fitted dispersion parameter ("dispersion").
formula	A model formula or terms object.
formula.	Changes to the formula.
evaluate	If true evaluate the new call else return the call.
nsim	number of response vectors to simulate. Defaults to 1.
seed	an object specifying if and how the random number generator should be initialized.
k	weight of the penalty term in AIC. Default is 2.

vcov.	a specification of the covariance matrix of the estimated coefficients.
df	the degrees of freedom to be used.

Value

The return value depends on the method called:

- print: returns x invisibly;
- summary: returns an object of class "summary.simplexregression";
- print.summary: returns x invisibly;
- coef: named numeric vector of coefficients;
- vcov: numeric matrix (variance-covariance);
- logLik: object of class "logLik";
- fitted: numeric vector of fitted mean values;
- predict: numeric vector or list depending on type;
- nobs: integer scalar (number of observations);
- df.residual: integer scalar (residual degrees of freedom);
- deviance: numeric scalar (total deviance);
- formula: the model formula;
- terms: the model terms for the selected submodel;
- model.frame: a data frame;
- model.matrix: numeric matrix of regressors;
- update: fitted model or call depending on evaluate;
- simulate: a data frame with nsim columns;
- AIC, BIC, HQIC: numeric scalar (single model) or data frame (multiple models);
- hatvalues: numeric vector of hat values;
- bread: numeric matrix;
- estfun: numeric matrix of score contributions;
- coeftest: object of class "coeftest";
- lrtest: object of class "anova".

See Also

simplexreg.

Examples

```
data(ReadingSkills, package = "SimplexRegression")
fit <- simplexreg(accuracy ~ dyslexia * iq | dyslexia + iq + I(iq^2),
                  data = ReadingSkills)

# Extract information
summary(fit)
```

```
coef(fit)
vcov(fit)
logLik(fit)
fitted(fit)
AIC(fit)
BIC(fit)
HQIC(fit)
hatvalues(fit)
```

`simplex_opt` *Simplex Distribution Functions*

Description

Density, distribution function, quantile function and random generation for the simplex distribution with parameters mean μ and dispersion σ^2.

Usage

```
dsimplex(x, mu, sigma2, log = FALSE)

psimplex(q, mu, sigma2, lower.tail = TRUE, log.p = FALSE)

qsimplex(p, mu, sigma2, lower.tail = TRUE, log.p = FALSE)

rsimplex(n, mu, sigma2)
```

Arguments

`x`, `q`	Numeric vector of quantiles.
`mu`	Mean parameter ($0 < \mu < 1$).
`sigma2`	Dispersion parameter ($\sigma^2 > 0$).
`log`, `log.p`	Logical; if `TRUE`, probabilities/densities p are given as $log(p)$.
`lower.tail`	Logical; if `TRUE` (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.
`p`	Numeric vector of probabilities.
`n`	Number of observations.

Details

The probability density function of the simplex distribution is given by:

$$f(y;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2[y(1-y)]^3}} \exp\left(-\frac{1}{2\sigma^2} d(y;\mu)\right),$$

where $y \in (0,1)$, and $d(y;\mu) = \frac{(y-\mu)^2}{y(1-y)\mu^2(1-\mu)^2}$ is the unit deviance.

The cumulative distribution function and the quantile function of the simplex distribution do not admit closed-form expressions. For small values of σ^2, `psimplex()` and `qsimplex()` use the normal approximation implied by the small-dispersion asymptotic theory (Jørgensen, 1997). Otherwise, `psimplex()` is computed by numerical integration of the density and `qsimplex()` is obtained by numerical root finding.

Random generation in `rsimplex()` is based on the inverse Gaussian mixture (M-IG) representation of the simplex distribution, followed by the transformation $Y = X/(1 + X)$.

Value

`dsimplex` gives the density, `psimplex` the distribution function, `qsimplex` the quantile function, and `rsimplex` generates random deviates.

For `sigma2` values requiring numerical root-finding (i.e., not small enough for the normal approximation), `qsimplex` may return `NA` if `uniroot` fails to find a root in the given interval.

Invalid arguments (`mu` outside (0, 1) or non-positive `sigma2`) will trigger an error.

References

Barndorff-Nielsen, O. E. and Jørgensen, B. (1991). Some parametric models on the simplex. *Journal of Multivariate Analysis*, **39**(1), 106–116. doi:10.1016/0047259X(91)90008P

Jørgensen B (1997). *The Theory of Dispersion Models*. Chapman and Hall, London.

Examples

```
dsimplex(0.5, mu = 0.3, sigma2 = 0.5)
dsimplex(0.5, mu = 0.3, sigma2 = 0.5, log = TRUE)
psimplex(0.5, mu = 0.3, sigma2 = 0.5)
psimplex(0.5, mu = 0.3, sigma2 = 0.5, lower.tail = FALSE)
qsimplex(0.5, mu = 0.3, sigma2 = 0.5)
qsimplex(log(0.5), mu = 0.3, sigma2 = 0.5, log.p = TRUE)
rsimplex(5, mu = 0.5, sigma2 = 0.5)
```

---

variance.simplex *Variance Function of the Simplex Distribution*

Description

Computes the variance of the simplex distribution as a function of the mean parameter μ and dispersion parameter σ^2.

Usage

```
variance.simplex(mu, sigma2)
```

Arguments

`mu`	Numeric scalar or vector of mean parameters ($0 < \mu < 1$). If a vector, must be the same length as `sigma2` or recyclable.
`sigma2`	Numeric scalar or vector of dispersion parameters ($\sigma^2 > 0$). If a vector, must be the same length as `mu` or recyclable.

Details

The variance function for the simplex distribution is given by:

$$Var(Y) = \mu(1-\mu) - \frac{1}{\sqrt{2\sigma^2}} \exp(a)\Gamma(0.5, a),$$

where $a = \frac{1}{2\sigma^2[\mu(1-\mu)]^2}$ and $\Gamma(\cdot,\cdot)$ is the upper incomplete gamma function.

For large values of $a(> 700)$, an asymptotic approximation is used to avoid numerical overflow:

$$Var(Y) \approx \mu(1-\mu) - \frac{1}{\sqrt{2\sigma^2}}\sqrt{\frac{1}{a}}.$$

Value

A numeric scalar or vector of variance values.

References

Jørgensen, B. (1997). *The Theory of Dispersion Models*. Chapman and Hall, London.

Song, P. X.-K. and Tan, M. (2000). Marginal models for longitudinal continuous proportional data. *Biometrics*, **56**(2), 496–502. doi:10.1111/j.0006341X.2000.00496.x

See Also

`dev.unit.simplex`, `dsimplex`.

Examples

```
# Single value
variance.simplex(mu = 0.5, sigma2 = 0.1)

# Vector of values
mu_vec <- c(0.3, 0.5, 0.7)
sigma2_vec <- c(0.1, 0.15, 0.2)
variance.simplex(mu = mu_vec, sigma2 = sigma2_vec)
```

Index