Package: SimplexRegression (via r-universe)

June 30, 2026

Type Package

Title Simplex Regression Models with Parametric or Fixed Mean Link Functions

Version 0.1.3

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>, Theoretical foundations)

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>.

License MIT + file LICENSE

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

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

Depends R (>= 3.5)

Imports Formula, lmtest, Matrix, pracma, sandwich

Suggests knitr, moments, 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 https://cran.r-universe.dev

Date/Publication 2026-06-30 12:30:12 UTC

RemoteUrl https://github.com/cran/SimplexRegression

RemoteRef HEAD

RemoteSha 1ce44e435e6ce7b866f3985b8cb7d83c80888eaf

Contents

Biomass . . . 3
dev.unit.simplex . . . 4
diag.distances . . . 5
diag.im . . . 8
dispersion_links . . . 10
fixed_mean_links . . . 11
gleverage . . . 13
halfnormal.plot . . . 14
local.influence . . . 15
parametric_mean_links . . . 17
penalized.ic . . . 19
penalized.ss . . . 21
plot.simplexregression . . . 23
press . . . 25
ReadingSkills . . . 26
RelativeHumidity . . . 28
resettest . . . 29
residuals.simplexregression . . . 31
scoretest . . . 33
simplex_opt . . . 34
simplexreg.control . . . 36
simplexreg.fit . . . 37
simplexreg.methods . . . 41
variance.simplex . . . 45

Index **47**

`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:

SpeciesName Species code (DfH, DfL, HlH, HlL) 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 Species scientific name (D. flexuosa or H. lanatus)

Trt Nitrate treatment level (high or low)

Day Experimental day (0 to 49)

PlNum Plant number (individual plant identifier, 6-8 replicates per treatment)

LDM..mg. Leaf dry mass (mg)

SDM..mg. Stem dry mass (mg)

RDM..mg. Root dry mass (mg)

TDM..mg. Total dry mass (mg)

LMF Leaf mass fraction - proportion of biomass allocated to leaves (0-1)

SMF Stem mass fraction - proportion of biomass allocated to stems (0-1)

RMF Root mass fraction - proportion of biomass allocated to roots (0-1)

lnTDM Natural log of total dry mass (log-transformed for allometric analysis)

Source

Data collected by Hendrik Poorter and co-workers. Published in:

Poorter, H., van de Vijver, C.A.D.M., Boot, R.G.A. & 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.

Poorter, H. & Sack, L. (2012) Pitfalls and possibilities in the analysis of biomass allocation patterns in plants. *Frontiers in Plant Science*, 3, 259.

References

When using this data, please cite:

Poorter, H. & 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)
```

---

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

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 W1, W2, or Hellinger) for simplex regression models.

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

For each observation i, the model is refit on the dataset with observation i removed. The chosen distance between the full-model and leave-one-out fitted distributions is then computed pointwise across all n observations and summed to produce the scalar influence measure d_i.

Ad hoc threshold uses an asymmetric IQR spread adjusted for skewness:

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

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

Warning: for $n > 500$, the numerical integration used internally by the distance functions may be slow. Consider using a subset or a faster integration method.

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 `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). 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

`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 silently clamped to that value 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

For each observation i, the model is refit on the dataset with observation i removed. Let $\hat{\theta}$ and $\hat{\theta}_{(i)}$ denote the full and leave-one-out MLEs, and let $A_{n,(i)}$ and $B_{n,(i)}$ be the corresponding information matrices.

Measures computed:

$$s_{3,i} = m_{3,(i)}/m_3$$

$$s_{5,i} = D_i^{\text{mod}} - D_i^{\text{gen}}$$

where

$$D_i^{\text{gen}} = (\hat{\theta} - \hat{\theta}_{(i)})^\top (-A_{n,(i)})(\hat{\theta} - \hat{\theta}_{(i)})$$

$$D_i^{\text{mod}} = (\hat{\theta} - \hat{\theta}_{(i)})^\top \tfrac{1}{2}(-A_{n,(i)} + B_{n,(i)})(\hat{\theta} - \hat{\theta}_{(i)})$$

and $m_3 = \|\text{vech}(P_n^{-1} B_n P_n^{-\top} - I)\|_2$, with $-A_n = P_n P_n^\top$ (Cholesky factorisation).

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

Threshold intervals use two asymmetric IQR spreads: $IQR_1 = Q(0.50) - Q(0.125)$ (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.

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 `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.; 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

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

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.

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`.

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 computing 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.simplexregressio` 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.

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 in. 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.

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., 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:

- 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`.

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)
```

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. Default is 0.1.
`verbose`	Logical. If `TRUE` (default), prints the criteria values. If `FALSE`, returns results silently.

Details

The penalized information criteria are computed as:

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

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

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

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 model employ a parametric link function in the mean submodel (use `kappa = 0.1`). When candidate models includes 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.

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

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

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

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

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)
```

Arguments

`...`	Two or more objects of class `simplexregression` to be compared.
`kappa`	A numeric value controlling the additional penalty for the link mean parameter. 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.

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} denotes Vuong's test statistic comparing models j and k:

$$\Delta_{jk} = n^{-1/2} \hat{\omega}_{jk}^{-1} \left[\sum_{i=1}^{n} \log \left(\frac{f_{ji}(\hat{\boldsymbol{\theta}}_j)}{f_{ki}(\hat{\boldsymbol{\theta}}_k)} \right) - \delta_{jk} \right],$$

with

$\hat{\omega}_{jk}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\log \left(\frac{f_{ji}(\hat{\boldsymbol{\theta}}_j)}{f_{ki}(\hat{\boldsymbol{\theta}}_k)} \right) \right)^2 - \left(\frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{f_{ji}(\hat{\boldsymbol{\theta}}_j)}{f_{ki}(\hat{\boldsymbol{\theta}}_k)} \right) \right)^2.$

Here, $f_{ji}(\hat{\boldsymbol{\theta}}_j)$ and $f_{ki}(\hat{\boldsymbol{\theta}}_k)$ denote the fitted densities under models j and k, respectively.

The penalization term δ_{jk} for simplex models with parametric and fixed mean links is given by:

$$\delta_{jk} = \frac{1}{2} [(r_j - r_k) + \kappa(|\log(\hat{\lambda}_j)| - |\log(\hat{\lambda}_k)|)] \log(n),$$

where r_j and r_k indicate the dimensions of their parameter vectors, and $\kappa \geq 0$ controls the additional penalty associated with the link parameter. The term $\kappa(|\log(\hat{\lambda}_j)| - |\log(\hat{\lambda}_k)|)$ measures link complexity on a logarithmic scale, ensuring symmetry around $\lambda = 1$.

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

Important: This penalized term δ_{jk} should only be used when all candidate models employ a parametric link function in the mean submodel (use `kappa = 0.1`). When the set of candidate models includes 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 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

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

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

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(),
  reset.par = TRUE,
  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.simplexregressio` for available options.
`ask`	Logical; if `TRUE`, the user is asked before each plot. Default is `TRUE` when multiple plots are requested.

reset.par	Logical; if TRUE, resets graphical parameters before plotting. Set to FALSE to preserve user-defined par() settings such as mfrow. Default is TRUE.
threshold	Numeric threshold for identifying influential observations in. If NULL (default), no observations are highlighted.
label.pos	Position(s) for outlier labels in plots 7 and 8. 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 = 1–5) and type = "h" (vertical lines) for Cook's distance and generalized leverage plots (which = 6–7). Otherwise, the value is passed directly to pch (for scatter plots) or type (for index plots).
...	Additional graphical parameters.

Details

Eight 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 P^2 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.

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"`.

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 h_{ii} is the i-th diagonal element of the hat matrix.

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 $\text{SST} = \sum_{i=1}^{n} (\hat{y}_i - \bar{\hat{y}})^2$ and r is the number of estimated parameters.

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.

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

Espinheira, P. L., 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

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")
```

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 (0,1) interval, 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. & Kevan, A. (2004). The Contribution of Visual Sensitivity, Phonological Processing and Non-Verbal IQ to Children's Reading. *Unpublished manuscript*, The Australian National University, Canberra.

Smithson, M. & Verkuilen, J. (2006). A Better Lemon Squeezer? Maximum-Likelihood Regression with Beta-Distributed Dependent Variables. *Psychological Methods*, 11(1), 54-71.

References

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

Kosmidis, I. & Zeileis, A. (2025). Extended-Support Beta Regression for [0, 1] Responses. *Journal of the Royal Statistical Society C*, forthcoming. 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, cloudiness, precipitation, atmospheric pressure, and wind speed for a Brazilian station (2000-2025).

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 Observation date (YYYY-MM-DD)

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

Ins Monthly total insolation (in hours), contains 2 missing values

Ins2 Monthly total insolation (in hours) with missing values imputed via seasonal interpolation

Pre Monthly total precipitation (in mm), contains 2 missing values

Pre2 Monthly total precipitation (in mm) with missing values imputed via seasonal interpolation

Neb Monthly mean cloudiness (in tenths)

AP Monthly mean atmospheric pressure (in hPa)

MT Monthly mean temperature (in Degrees Celsius)

WS Monthly mean wind speed (in m/s)

Dir Monthly predominant wind direction (in degrees)

Source

Brazilian meteorological station

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 test for functional form 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 linear predictor (or fitted 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 mean linear predictor $\hat{\eta}_1$; `"fitted"` uses the fitted mean values $\hat{\mu}$ on the (0, 1) scale. For models with parametric link functions, `"lp"` is generally preferred as it operates on an unrestricted scale.

Details

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

The likelihood ratio statistic follows a chi-squared distribution 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.

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 $r_i^P = (y_i - \hat{\mu}_i)/\sqrt{\widehat{\mathrm{Var}}(y_i)}$, 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 $r_i^D = (y_i - \hat{\mu}_i)/(\hat{\mu}_i(1 - \hat{\mu}_i)\sqrt{y_i(1 - y_i)})$.

Standardized residuals (`"standardized"`): Proposed by Espinheira and Silva (2020, Eq. 15) as $r_i^\beta = \hat{u}_i(y_i - \hat{\mu}_i)/\sqrt{\hat{\sigma}_i^2 \hat{w}_i}$, where $\hat{w}_i$ and $\hat{u}_i$ are weight functions.

Weighted residuals (`"weighted"`): Proposed by Espinheira and Silva (2020, Eq. 16) as $r_i^{\beta *} = \hat{u}_i(y_i - \hat{\mu}_i)/\sqrt{\hat{\sigma}_i^2 \hat{w}_i(1 - \hat{h}_{ii})}$, 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 $r_i^{\gamma} = (\hat{d}_i - \hat{\sigma}_i^2)/(\hat{\sigma}_i^2\sqrt{2})$, where $\hat{d}_i$ is the estimated unit deviance.

Bias–variance residuals (`"biasvariance"`): Proposed by Espinheira et al. (2021, Eq. 8) as $r_i^{\beta\gamma} = (\hat{u}_i(y_i - \hat{\mu}_i) + \hat{a}_i)/\sqrt{\hat{\sigma}_i^2\hat{w}_i + 1/(2\hat{\sigma}_i^4)}$, where $\hat{a}_i$ is a correction term.

Score residuals (`"score"`): Defined in Jørgensen (1997, p. 115) as $r_i^S = (y_i - \hat{\mu}_i)(\hat{\mu}_i^2 + y_i - 2y_i\hat{\mu}_i)/(y_i(1-y_i)\hat{\mu}_i^{1.5}(1-\hat{\mu}_i)^{1.5})$.

Dual score residuals (`"dualscore"`): Defined in Jørgensen (1997, p. 115) as $r_i^{DS} = (y_i - \hat{\mu}_i)(y_i + \hat{\mu}_i - 2y_i\hat{\mu}_i)/(2\sqrt{y_i(1-y_i)}\hat{\mu}_i^2(1-\hat{\mu}_i)^2)$.

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

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. 2nd ed. Chapman and Hall, London. Jørgensen, B. (1997). *The Theory of Dispersion Models*. Chapman and Hall, London.

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., 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

Espinheira, P. L., 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

`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`.
`link.mu`	Character string specifying the 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 for this hypothesis test is given by:

$$S_R = \boldsymbol{U}_\lambda(\tilde{\boldsymbol{\theta}})^\top \tilde{\boldsymbol{K}}^{\lambda\lambda} \boldsymbol{U}_\lambda(\tilde{\boldsymbol{\theta}}),$$

where $\boldsymbol{U}_\lambda$ and $\tilde{\boldsymbol{K}}^{\lambda\lambda}$ denote, respectively, the component of the score vector and the corresponding element of the inverse Fisher information matrix associated with λ, both evaluated at $\tilde{\boldsymbol{\theta}} = (\tilde{\boldsymbol{\beta}}^\top, \tilde{\boldsymbol{\gamma}}^\top, 1)^\top$, the maximum likelihood estimator under the null hypothesis. For more details see Justino and Cribari-Neto (2025).

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`: Model name and link function being tested.

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

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")
```

`simplex_opt` *Simplex Distribution Functions*

Description

Density, distribution function, quantile function and random generation for the simplex distribution with parameters mean μ and dispersion $\sigma^2 > 0$.

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:

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

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

Value

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

Invalid arguments will result in return value NaN, with a warning.

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

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)
```

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`	Characters string specifying the method argument 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: 1e-8).
`...`	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 coefficients β in the linear predictor of the mean submodel and the coefficients δ 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 μ (via parametric or fixed link function) and the dispersion parameter σ^2.

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 | 1 or y ~ x (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"; 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: $g(\mu_i, \lambda) = x_i'\beta$ (parametric link) or $g(\mu_i) = x_i'\beta$ (fixed link),
- Dispersion submodel: $h(\sigma_i^2) = z_i'\gamma$.

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 coefficients from the respective submodels, and (for parametric mean link functions) an additional element `lambda` with the estimated link parameter,
- `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 linear predictors for the mean submodel,
- `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 linear predictors for the dispersion model,
- `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 link 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`, `HQ`: 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.

Author(s)

Maria Eduarda da Cruz Justino and Francisco Cribari-Neto.

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

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.

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 simplex regression 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'
cooks.distance(model, type = c("pearson", "weighted"), ...)

## 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 to 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 leverage values;
- cooks.distance: numeric vector of Cook's distances;
- 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)
cooks.distance(fit)
```

`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 $\Gamma(0.5, a)$ 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

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

∗ **datasets**
Biomass, 3
ReadingSkills, 26
RelativeHumidity, 28

AIC, *39*
AIC.simplexregression (simplexreg.methods), 41

BIC.simplexregression (simplexreg.methods), 41
Biomass, 3
bread, *39*
bread.simplexregression (simplexreg.methods), 41

clusterSetRNGStream, *6*, *9*
coef, *39*
coef.simplexregression (simplexreg.methods), 41
coeftest, *39*
coeftest.simplexregression (simplexreg.methods), 41
cooks.distance, *39*
cooks.distance.simplexregression, *10*, *13*, *17*, *24*
cooks.distance.simplexregression (simplexreg.methods), 41

dev.unit.simplex, 4
deviance.simplexregression (simplexreg.methods), 41
df.residual.simplexregression (simplexreg.methods), 41
diag.distances, 5
diag.im, *7*, 8
dispersion_link (dispersion_links), 10
dispersion_link_deriv1 (dispersion_links), 10
dispersion_link_inv (dispersion_links), 10
dispersion_link_inv_deriv1 (dispersion_links), 10
dispersion_links, 10
dsimplex (simplex_opt), 34

estfun, *39*
estfun.simplexregression (simplexreg.methods), 41

fitted, *39*
fitted.simplexregression (simplexreg.methods), 41
fixed_mean_link (fixed_mean_links), 11
fixed_mean_link_deriv1 (fixed_mean_links), 11
fixed_mean_link_deriv2 (fixed_mean_links), 11
fixed_mean_link_inv (fixed_mean_links), 11
fixed_mean_link_inv_deriv1 (fixed_mean_links), 11
fixed_mean_links, 11, *18*
formula, *39*
formula.simplexregression (simplexreg.methods), 41

gleverage, *7*, *10*, 13, *39*
gleverage.simplexregression, *17*

halfnormal.plot, 14, *24*, *32*
hatvalues, *39*
hatvalues.simplexregression, *13*, *17*, *31*, *32*
hatvalues.simplexregression (simplexreg.methods), 41
HQIC (simplexreg.methods), 41

lm.wfit, *37*
local.influence, *7*, *10*, *13*, 15
logLik, *39*

logLik.simplexregression
(simplexreg.methods), 41
lrtest.simplexregression
(simplexreg.methods), 41

model.frame, *39*
model.frame.simplexregression
(simplexreg.methods), 41
model.matrix, *39*
model.matrix.default, *38*
model.matrix.simplexregression
(simplexreg.methods), 41
model.offset, *38*

nobs.simplexregression
(simplexreg.methods), 41

offset, *38*
optim, *36*

parametric_mean_link
(parametric_mean_links), 17
parametric_mean_link_deriv1
(parametric_mean_links), 17
parametric_mean_link_deriv2
(parametric_mean_links), 17
parametric_mean_link_inv
(parametric_mean_links), 17
parametric_mean_link_inv_deriv1
(parametric_mean_links), 17
parametric_mean_links, *12*, 17
parLapply, *6*, *8*, *9*
penalized.ic, 19, *22*, *41*
penalized.ss, *20*, 21, *41*
plot, *39*
plot.simplexregression, *15*, 23, *32*
predict, *39*
predict.simplexregression, *41*
predict.simplexregression
(simplexreg.methods), 41
press, 25
print, *39*
print.simplexregression
(simplexreg.methods), 41
print.summary.simplexregression
(simplexreg.methods), 41
psimplex, *31*
psimplex (simplex_opt), 34

qsimplex (simplex_opt), 34

ReadingSkills, 26
RelativeHumidity, 28
resettest, 29
residuals, *39*
residuals.simplexregression, *14*, *15*, *23*, *24*, *26*, 31, *41*
rsimplex (simplex_opt), 34

scoretest, 33, *41*
simplex_opt, 34
simplexreg, *11*, *12*, *18*, *26*, *30*, *36*, *37*, *44*
simplexreg (simplexreg.fit), 37
simplexreg.control, 36, *36*, *38*
simplexreg.fit, 37, *38*
simplexreg.methods, 41
simulate, *39*
simulate.simplexregression
(simplexreg.methods), 41
summary, *39*
summary.simplexregression, *41*
summary.simplexregression
(simplexreg.methods), 41

terms, *39*
terms.simplexregression
(simplexreg.methods), 41
text, *24*

update, *39*
update.simplexregression
(simplexreg.methods), 41

variance.simplex, *31*, 45
vcov, *39*
vcov.simplexregression
(simplexreg.methods), 41