Package ‘osktnorm’

March 17, 2026

Type Package

Title A Moment-Targeting Normality Transformation Based on Tukey g-h Distribution

Version 1.1.2

Date 2026-03-07

Maintainer Zeynel Cebeci <cebeciz@gmail.com>

Description Implements a moment-targeting normality transformation based on the simultaneous optimization of Tukey g-h distribution parameters. The method is designed to minimize both asymmetry (skewness) and excess peakedness (kurtosis) in non-normal data by mapping it to a standard normal distribution Cebeci et al (2026) <doi:10.3390/sym18030458>. Optimization is performed by minimizing an objective function derived from the Anderson-Darling goodness-of-fit statistic with Stephens's correction factor, utilizing the L-BFGS-B algorithm for robust parameter estimation. This approach provides an effective alternative to power transformations like Box-Cox and Yeo-Johnson, particularly for data requiring precise tail-behavior adjustment.

Depends R (>= 4.1.0)

Imports Rcpp, stats, doParallel, foreach, parallel, groupcompare

LinkingTo Rcpp

Suggests knitr, rmarkdown, readxl, writexl

VignetteBuilder knitr

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation yes

Author Zeynel Cebeci [aut, cre],
Figen Ceritoglu [aut],
Melis Celik Guney [aut],
Adnan Unalan [aut]

Repository CRAN

Date/Publication 2026-03-17 18:20:02 UTC

Contents

---

abc *Adaptive Box-Cox (ABC) Normalization*

Description

Adaptive Box-Cox (ABC) transformation (Yu et al, 2022) is a data transformation method to transform a non-normal numeric variable toward normality by tuning a power parameter based on a normality test result. The method selects the optimal transformation parameter by maximizing the Shapiro-Wilk normality test p-value.

Usage

```
abc(x, lrange = seq(-3, 3, 0.01))
```

Arguments

`x`	A numeric vector containing the data to be transformed. If non-positive values are present, the data are automatically shifted to ensure positivity.
`lrange`	A numeric vector specifying the range of lambda values to be evaluated. The default is a sequence from -3 to 3 with step size 0.01.

Details

The ABC method searches over a predefined range of transformation parameters (λ) and applies a Box-Cox-type transformation for each candidate value.

For each λ, the transformed data are tested for normality using the Shapiro-Wilk test. The optimal λ is selected as the value that maximizes the logarithm of the Shapiro-Wilk p-value.

The transformation is defined as:

$$x^{(\lambda)} = \begin{cases} \log(x), & \lambda = 0 \\ (x^{\lambda} - 1)/\lambda, & \lambda \neq 0 \end{cases}$$

In this implementation, to satisfy the Box-Cox requirement of strictly positive data, the function automatically shifts the input vector if any non-positive values are detected.

Value

A list with the following components:

`transformed`	A numeric vector containing the transformed data.
`lambda`	The selected lambda value that maximizes the Shapiro–Wilk log p-value.

Author(s)

Zeynel Cebeci

References

Yu, H., Sang, P., & Huan, T. (2022). Adaptive Box-Cox transformation: A highly flexible feature-specific data transformation to improve metabolomic data normality for better statistical analysis. *Analytical Chemistry, 94*(23), 8267-8276. doi:10.1021/acs.analchem.2c00503.

Examples

```
set.seed(12)
x <- rexp(100)

result <- abc(x)
result$lambda
hist(result$transformed, main = "ABC Transformed Data")
```

`backoskt` *Inverse OSKT Normality Transformation*

Description

Performs numerical back-transformation (inverse transformation) for the Optimized Skewness and Kurtosis Transformation (OSKT) based on the Tukey $g - h$ family (Tukey, 1977). The function recovers the original-scale data from the OSKT-transformed values using either a Newton–Raphson algorithm or a bracketing root-finding method.

Usage

```
backoskt(Z, X_mean, X_sd, g, h, tol = 1e-10, maxiter = 1e6, method = c("ur", "nr"))
```

Arguments

`Z`	A numeric vector of OSKT-transformed and standardized observations.
`X_mean`	Sample mean of the original data used during standardization.
`X_sd`	Sample standard deviation of the original data used during standardization.
`g`	Skewness parameter of the Tukey $g - h$ transformation.
`h`	Kurtosis (tail heaviness) parameter of the Tukey $g - h$ transformation.
`tol`	Numerical tolerance for convergence of the root-finding algorithms. Default is `1e-10`.
`maxiter`	Maximum number of iterations allowed for the `uniroot` method. Default is `1e6`.
`method`	Character string specifying the numerical inversion method. Either `"nr"` for Newton–Raphson (fast but potentially unstable) or `"ur"` for a bracketing root-finding method (robust but slower).

Details

The OSKT transformation is based on the Tukey $g - h$ transformation applied to standardized data. Since the inverse transformation has no closed-form solution, numerical methods are required.

Two inversion strategies are provided:

- `"nr"`: Newton–Raphson iteration initialized at $x = z$, offering fast convergence when the derivative is well-behaved.
- `"ur"`: A safe bracketing method using `uniroot`, ensuring convergence at the expense of computational speed.

After inversion, the results are de-standardized using the supplied mean and standard deviation to recover the original data scale.

Value

A list with the following components:

- `X_orig`: Back-transformed observations on the original scale.
- `X_s`: Back-transformed standardized values.
- `time_seconds`: Total computation time in seconds.
- `method`: The numerical inversion method used.

Author(s)

Zeynel Cebeci

References

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

Headrick, T. C., & Pant, M. D. (2012). Characterizing Tukey g and h distributions through their moments. *Journal of Statistical Distributions and Applications*, 1(1), 1–20.

See Also

`oskt`, `osktfast`, `backosktfast`

Examples

```
set.seed(123)
x <- rt(200, df = 5)

# Example parameters (typically estimated via oskt)
g <- 0.2
h <- 0.15

# Standardize and apply forward g-h transformation
x_s <- scale(x)
z <- ((exp(g * x_s) - 1) / g) * exp(0.5 * h * x_s^2)

# Back-transformation
res <- backoskt(
  Z = z,
  X_mean = mean(x),
  X_sd = sd(x),
  g = g,
  h = h,
  method = "nr"
)

head(x)

head(res$X_orig)

plot(x, res$X_orig, xlab="Original", ylab="Back transformed", col="blue", pch=19)
```

```
hist(x)
hist(res$X_orig)
```

`backosktfast` *Fast Reverse OSKT Transformation*

Description

Computes the inverse of the Optimized Skewness and Kurtosis Transformation (OSKT) using high-performance numerical root-finding algorithms implemented in C++. The function efficiently recovers original-scale observations from OSKT-transformed values by solving a nonlinear equation for each observation.

Usage

```
backosktfast(
  Z, X_mean, X_sd,
  g, h,
  method = "auto",
  tol = 1e-10,
  maxiter_nr = 1000,
  maxiter_brent = 2000
)
```

Arguments

`Z`	Numeric vector of OSKT-transformed values to be inverted. Missing values (`NA`) are allowed and are propagated to the output.
`X_mean`	Numeric scalar. Mean of the original data before standardization.
`X_sd`	Numeric scalar. Standard deviation of the original data before standardization.
`g`	Numeric scalar. Optimized skewness parameter returned by the OSKT transformation function. Values close to zero are handled using a numerically stable limiting form.
`h`	Numeric scalar. Optimized kurtosis parameter returned by the OSKT transformation function. Must be non-negative ($h \geq 0$).
`method`	Character string specifying the numerical root-finding strategy: `"auto"` (Default) Attempts Newton–Raphson first and falls back to Brent–Dekker if convergence fails. Recommended for most use cases. `"nr"` Pure Newton–Raphson method only. Fastest but no fallback. `"brent"` Pure Brent–Dekker method only. Most robust but slower.
`tol`	Positive numeric scalar specifying the convergence tolerance for the root-finding algorithms.
`maxiter_nr`	Positive integer. Maximum number of iterations allowed for the Newton–Raphson phase.
`maxiter_brent`	Positive integer. Maximum number of iterations allowed for the Brent–Dekker phase.

Details

The Optimized Skewness and Kurtosis Transformation (OSKT) is defined as

$$T_{g,h}(x_s) = \frac{e^{gx_s} - 1}{g} \, e^{\frac{1}{2}hx_s^2},$$

where $x_s = (X - \mu)/\sigma$ is the standardized variable.

When $g = 0$, the transformation is defined by the continuous limit

$$T_{0,h}(x_s) = x_s \, e^{\frac{1}{2}hx_s^2}.$$

This function numerically solves the nonlinear equation

$$T_{g,h}(x_s) = Z$$

for x_s, and then applies the inverse standardization

$$X = x_s\sigma + \mu.$$

Numerical Methods:

- **Newton–Raphson** uses analytic derivatives and exhibits quadratic convergence near the solution but may fail for extreme values or poor initial guesses.
- **Brent–Dekker** is a robust bracketing algorithm combining bisection, secant, and inverse quadratic interpolation (Brent, 1973). Convergence is guaranteed if a root is bracketed.
- **Auto mode** combines both approaches, achieving high performance while retaining robustness.

All heavy numerical computations are implemented in C++ via **Rcpp**.

Value

A list with the following components:

X_orig Numeric vector of inverse-transformed values on the original scale. Entries are `NA` where inversion failed or input values were `NA`.

method_used Character vector of the same length as `Z`, indicating which method succeeded for each observation:

`"failed"` Root-finding failed to converge.

`"nr"` Newton–Raphson succeeded (when `method = "nr"`).

`"brent"` Brent–Dekker succeeded (when `method = "brent"`).

`"auto-nr"` Newton–Raphson succeeded in auto mode.

`"auto-brent"` Brent–Dekker fallback succeeded in auto mode.

Author(s)

Zeynel Cebeci

References

Brent, R. P. (1973). *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ.

See Also

`oskt` for the forward OSKT transformation.
`osktfast` for the fast forward OSKT transformation.
`backoskt` for the forward OSKT transformation using uniroot in R.
`uniroot` for R's base root-finding routine.

Examples

```
# Example data
set.seed(123)
X <- c(-50, -10, 0, 10, 50)
Z <- scale(X)

# Newton-Raphson
res_nr <- backosktfast(Z, 0, 1, g = 0.5, h = 0.1, method = "nr")
res_nr$X_orig

# Brent-Dekker
res_br <- backosktfast(Z, 0, 1, g = 0.5, h = 0.1, method = "brent")
res_br$X_orig

# Auto mode
res <- backosktfast(Z, X_mean = 0, X_sd = 1, g = 0.5, h = 0.1)
res$X_orig
table(res$method_used)

# Handling missing values
Z_na <- c(-10, 0, 10, NA)
backosktfast(Z_na, 0, 1, g = 0.3, h = 0.05)$X_orig
```

---

boxcox *Box-Cox Transformation*

Description

Performs a Box-Cox transformation on a numeric vector. Optionally, the data can be shifted to ensure all values are positive before applying the transformation. If lambda is not provided, it is estimated via maximum likelihood.

Usage

```
boxcox(x, lambda = NULL, makepositive = FALSE, eps = 1e-06)
```

Arguments

`x`	A numeric vector to be transformed.
`lambda`	Optional numeric value of the Box–Cox transformation parameter. If `NULL` (default), the value of `lambda` is estimated by maximizing the profile log-likelihood.
`makepositive`	Logical. If `TRUE`, the data are shifted so that all observations are strictly positive before applying the transformation. This is required when `x` contains zero or negative values. Default is `FALSE`.
`eps`	A small positive constant used for numerical stability. It is added implicitly when enforcing positivity or to avoid taking the logarithm of zero. Default is `1e-06`.

Details

The Box-Cox transformation is defined as:

$$y(\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(x), & \lambda = 0 \end{cases}$$

If `makepositive` = `TRUE`, the function shifts the data by `abs(min(x)) + 1` if there are zero or negative values, to make all values positive.

Lambda is estimated via maximum likelihood over a grid of values from -4 to 4 (default 500 points) if not specified.

Value

A list with the following components:

`transformed`	The transformed numeric vector.
`lambda`	The lambda used in the transformation (either provided or estimated).
`shift`	The amount by which the original data was shifted to make it positive (0 if no shift).

Author(s)

Zeynel Cebeci

See Also

`abc`, `rewbc`

Examples

```
# Generate positively skewed example data
set.seed(123)
x <- rlnorm(50, meanlog = 0, sdlog = 1)

# Box-Cox with estimated lambda (MLE)
```

```
res <- boxcox(x)
head(res$transformed)
res$lambda
res$shift

# Box-Cox with specified lambda
res2 <- boxcox(x, lambda = 0.25)
head(res2$transformed)

# Box-Cox with automatic shift for nonpositive values
x2 <- x - quantile(x, 0.2)
res3 <- boxcox(x2, makepositive = TRUE)
head(res3$transformed)
res3$shift
```

`cvmtest` *Cramér-von Mises Normality Test*

Description

Performs a Monte Carlo based Cramér-von Mises (CVM) goodness-of-fit test for normality. The p-value is approximated via simulation under the standard normal distribution.

Usage

```
cvmtest(x, nsim = 10000, seed = NULL)
```

Arguments

`x`	A numeric vector of observations to be tested for normality.
`nsim`	Number of Monte Carlo simulations used to approximate the null distribution of the CVM statistic. Default is `1e4`.
`seed`	Optional integer seed for reproducibility. If `NULL`, the random number generator state is not modified.

Details

The Cramér-von Mises statistic measures the squared distance between the empirical distribution function of the data and the theoretical cumulative distribution function under the null hypothesis.

In this implementation, the null hypothesis assumes that the data follow a standard normal distribution. The CVM statistic is computed as:

$$W^2 = \frac{1}{12n} + \sum_{i=1}^{n} \left(F(x_{(i)}) - \frac{2i-1}{2n} \right)^2$$

where F denotes the standard normal cumulative distribution function and $x_{(i)}$ are the ordered observations.

Because the exact distribution of the statistic is not used, the p-value is estimated via Monte Carlo simulation by repeatedly generating samples from the standard normal distribution and recomputing the CVM statistic.

Value

A list with the following components:

`statistic`	Observed Cramér-von Mises test statistic.
`p.value`	Monte Carlo estimated p-value.

Author(s)

Zeynel Cebeci

References

Cramér, H. (1928). On the composition of elementary errors: First paper: Mathematical deductions. *Scandinavian Actuarial Journal, 1928*(1), 13-74. doi:10.1080/03461238.1928.10416862.

von Mises, R. (1931). Wahrscheinlichkeitsrechnung und ihre Anwendung in der Statistik und theoretischen Physik.

Examples

```
# Generate normal distributed data
set.seed(123)
x <- rnorm(50)

result <- cvmtest(x, nsim = 1000) # Increase in real apps
result$statistic
result$p.value

# Generate positively skewed example data
set.seed(123)
x <- rlnorm(50, meanlog = 0, sdlog = 1)

result <- cvmtest(x, nsim = 1000) # Increase in real apps
result$statistic
result$p.value
```

`int` *Inverse Normal Transformation*

Description

Performs a rank-based inverse normal transformation (INT) that maps a numeric vector to approximately standard normal scores.

Usage

```
int(x, ties.method = "average", na.action = "keep")
```

Arguments

`x`	A numeric vector to be transformed. Missing values (NA) are allowed.
`ties.method`	A character string specifying how ties are handled in ranking. Default is `"average"`. See `rank` for available options.
`na.action`	A character string indicating how missing values are treated during ranking. Default is `"keep"`.

Details

The inverse normal transformation (INT) is a nonparametric normalization method based on data ranks. The procedure first maps ranks to uniform quantiles and then applies the inverse standard normal distribution function.

For each non-missing observation x_i, the transformation is defined as: $Z_i = qnorm((rank(x_i) - 0.5)/n)$ where n is the number of non-missing observations.

Value

A list with one element:

transformed A numeric vector of the same length as `x` containing inverse normal transformed values. Missing values are preserved.

See Also

`qnorm`, `rank`

Examples

```
x <- c(5, 2, 8, 8, 3)
res <- int(x)
res$transformed

# With missing values
x2 <- c(1.2, NA, 3.4, 2.1)
int(x2)$transformed
```

`lambert` *Lambert W x F Transformation for Gaussianizing Data*

Description

Transforms a non-normal variable into a Gaussian (Normal) distribution using the Iterative Generalized Method of Moments (IGMM) for Lambert W x F transforms. It handles skewed (`s`), heavy-tailed (`h`), or both (`hh`) distributions.

Usage

```
lambert(x, type = c("s", "h", "hh"), maxiter = 200, tol = 1e-06,
        step_gamma = 0.25, step_delta = 0.1)
```

Arguments

`x`	A numeric vector to be transformed.
`type`	Character string specifying the type of transformation: `"s"` for skewed, `"h"` for symmetric heavy-tailed, and `"hh"` for both skewed and heavy-tailed.
`maxiter`	Maximum number of IGMM iterations. Default is 200.
`tol`	Convergence tolerance for parameter updates. Default is 1e-6.
`step_gamma`	The damping factor for the skewness parameter (γ) update. Default is 0.25.
`step_delta`	The damping factor for the heavy-tail parameter (δ) update. Default is 0.1.

Details

The function uses a robust Halley's method to solve the Lambert W function internally. The IGMM algorithm iteratively updates the transformation parameters (γ and δ) to minimize the skewness and excess kurtosis of the latent variable.

Value

A list containing:

`transformed`	The numeric vector of Gaussianized values, maintaining NA positions.
`params`	A list of estimated parameters: `gamma` (skewness), `delta` (tail heaviness), `mu` (original mean), and `sigma` (original SD).
`iterations`	Number of iterations performed until convergence.
`converged`	Logical indicating if the algorithm converged within `maxiter`.
`method`	Character string indicating the estimation method (`"IGMM"`).

Author(s)

Zeynel Cebeci

References

Goerg, G. M. (2011). Lambert W random variables - A new family of generalized skewed distributions with applications to risk estimation. *The Annals of Applied Statistics*, 5(3), 2197-2230. doi:10.1214/11AOAS457

Goerg, G. M. (2015). The Lambert Way to Gaussianize heavy-tailed data with the inverse of Tukey's h transformation as a special case. *The Scientific World Journal*, 2015, 1-18. doi:10.1155/2015/909231

Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffrey, D. J., & Knuth, D. E. (1996). On the Lambert W function. *Advances in Computational Mathematics*, 5(1), 329-359. doi:10.1007/BF02124750

Examples

```
# Generate skewed data using a Gamma distribution
set.seed(123)
skewed_data <- rgamma(500, shape = 2, scale = 2)

# Apply the Lambert W transformation (skewed type)
result <- lambert(skewed_data, type = "s")

# Visualization
opar <- par(mfrow = c(1, 2))
hist(skewed_data, main = "Original (Gamma)", col = "orange", breaks = 30)
hist(result$transformed, main = "Gaussianized (Lambert)", col = "skyblue", breaks = 30)
par(opar)
```

---

oskt — *Normalization via Skewness and Kurtosis Minimization of Anderson–Darling Test Statistic*

Description

Applies the Tukey g-and-h transformation to a numeric vector and estimates optimal skewness and tail-heaviness parameters by minimizing the Anderson-Darling normality test statistic (A^2).

Usage

```
oskt(x, init_params = c(0.1, 0.1), lower_bounds = c(-1, 0), upper_bounds = c(1, 0.5))
```

Arguments

`x`	A numeric vector of observations to be transformed. The data are internally standardized to zero mean and unit variance.
`init_params`	Numeric vector of length two giving the initial values of the Tukey g-and-h parameters (`g`, `h`).
`lower_bounds`	Numeric vector of length two specifying lower bounds for `g` and `h`.
`upper_bounds`	Numeric vector of length two specifying upper bounds for `g` and `h`.

Details

The Tukey g-and-h transformation is defined as:

$$Y = \begin{cases} \dfrac{\exp(gX) - 1}{g} \exp\left(\dfrac{hX^2}{2}\right), & g \neq 0, \\ X \exp\left(\dfrac{hX^2}{2}\right), & g = 0. \end{cases}$$

where g controls skewness and h controls tail heaviness.

To assess normality of the transformed data, the Anderson–Darling statistic A^2 is computed directly from its analytical form under the standard normal distribution. The implementation includes the Stephens correction used when location and scale parameters are estimated from the data:

$$A^{2*} = A^2 \left(1 + \frac{0.75}{n} + \frac{2.25}{n^2} \right).$$

Optimal parameters are obtained by minimizing the corrected statistic A^{2*} using constrained optimization via the `"L-BFGS-B"` algorithm.

Value

A list with the following components:

`transformed`	Numeric vector containing the transformed data.
`g`	Estimated skewness parameter of the Tukey g-and-h transformation.
`h`	Estimated tail-heaviness parameter of the Tukey g-and-h transformation.

Note

This function does not rely on external normality testing packages. The Anderson–Darling statistic is computed explicitly to ensure numerical consistency and package independence.

If optimization fails, the standardized input data are returned and `g` and `h` are set to `NA`.

Author(s)

Zeynel Cebeci

References

Stephens, M. A. (1974). EDF statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347), 730-737.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

Examples

```
set.seed(123)
x <- rexp(100)

restrans <- oskt(x)
restrans$g
restrans$h

hist(restrans$transformed, xlab="Transformed Values", main = "OSKT Transformed Data")

set.seed(123)
x <- rlnorm(300, meanlog = 0, sdlog = 0.75)

restrans <- oskt(x)
restrans$g
```

```
restrans$h

hist(restrans$transformed, xlab="Transformed Values", main = "OSKT Transformed Data")
```

---

osktfast *Optimized Skewness-Kurtosis Transformation (OSKT)*

Description

Performs a g-and-h transformation to reduce skewness and kurtosis, minimizing the Anderson-Darling A2 statistic. The transformation aims to normalize non-normal data by targeting skewness and kurtosis simultaneously.

Usage

```
osktfast(x,
         init_params = c(0.1, 0.1),
         lower_bounds = c(-1, 0),
         upper_bounds = c(1, 0.5),
         maxiter = 200)
```

Arguments

`x`	Numeric vector of input data.
`init_params`	Numeric vector of length 2: initial values for g and h. Default is c(0.1, 0.1).
`lower_bounds`	Numeric vector of length 2: lower bounds for g and h. Default is c(-1, 0).
`upper_bounds`	Numeric vector of length 2: upper bounds for g and h. Default is c(1, 0.5).
`maxiter`	Integer: maximum number of iterations for the optimizer. Default is 200.

Details

This function uses a pure C++ implementation of the L-BFGS-B algorithm to optimize the g-and-h transformation parameters for normality. The objective function is the modified Anderson-Darling A2 statistic with Stephen's correction for small samples. It is suitable for moment-targeting normalization is desired.

Value

A list containing:

`transformed`	The transformed numeric vector.
`g`	Estimated g parameter of the g-and-h transformation.
`h`	Estimated h parameter of the g-and-h transformation.
`value`	Anderson-Darling A2 statistic at the optimum.

Author(s)

Zeynel Cebeci

Examples

```
set.seed(123)
x <- rnorm(100, mean=5, sd=2) # Generate non-normal data
res <- osktfast(x)

# Check transformed data
head(res$transformed)
res$g
res$h
res$value
```

`osktnorm` *Column-wise OSKT Normalization for Numeric Traits*

Description

Applies Optimal Skewness–Kurtosis Transformation (OSKT) column-wise to numeric variables in a matrix or data frame. Non-numeric columns are automatically excluded and reported. Optional normality diagnostics can be computed for transformed traits.

Usage

```
osktnorm(data,
         normtests = FALSE,
         nsim = 100,
         shapiro_limit = 5000,
         verbose = TRUE,
         keep_raw = FALSE)
```

Arguments

`data` A `matrix` or `data.frame`. Non-numeric columns are automatically removed prior to transformation.

`normtests` Controls which normality diagnostics are computed for transformed traits.

Possible values:

- `FALSE`: no normality tests are performed.
- `TRUE` or `"all"`: compute all available diagnostics.
- Character vector selecting specific tests, e.g., `"cvm"`, `"zc"`, or combinations (e.g., `c("zc","cvm")`).

Available test identifiers (case-insensitive):

- `"skew"`: sample skewness
- `"kurt"`: excess kurtosis

- `"sw"`: Shapiro–Wilk test p-value
- `"za"`: ZA test p-value (`zatest`)
- `"zc"`: ZC test p-value (`zctest`)
- `"cvm"`: Cramér–von Mises test p-value (`cvmtest`)
- `"ppm"`: Pearson PPM statistic (`pearsonp`)

`nsim`	Integer. Number of Monte Carlo simulations used in `zatest()` and `zctest()`.
`shapiro_limit`	Integer. Maximum sample size allowed for Shapiro–Wilk test. If the sample size exceeds this limit, the Shapiro p-value is returned as `NA`.
`verbose`	Logical. If `TRUE`, reports excluded non-numeric columns.
`keep_raw`	Logical. If `TRUE`, includes full `osktfast()` outputs for each trait in the returned object.

Details

The function performs the following steps:

1. Validates that the input is a matrix or data frame.
2. Detects numeric columns.
3. Excludes non-numeric columns and reports them (if `verbose = TRUE`).
4. Stops with an error if no numeric columns remain.
5. Applies `osktfast()` to each numeric trait.
6. Optionally computes selected normality diagnostics on the transformed data.

Normality tests are done on the transformed values **after** normalization. Even if a single test is requested, the output in `normtests` remains a data frame organized by trait (rows).

Value

An object of class `"osktnorm"` containing:

- `normalized`: A data frame containing the transformed numeric traits.
- `parameters`: A data frame of optimized OSKT parameters (g, h, and objective value `A2`) for each trait.
- `normtests`: A data frame of selected normality diagnostics where each row corresponds to a trait and each column to a test. Returns `NULL` if `normtests = FALSE`.
- `removed_columns`: A character vector of excluded non-numeric column names.

See Also

`osktfast`, `zatest`, `zctest`, `cvmtest`, `pearsonp`

Examples

```
set.seed(123)
origdata <- data.frame(
  id     = factor(sprintf("id%03d", 1:100)),
  trait1 = rexp(100),
  trait2 = rchisq(100, df = 3),
  group  = factor(sample(letters[1:3], 100, TRUE))
)

res1 <- osktnorm(origdata)
head(res1$normalized)
res1$parameters

res2 <- osktnorm(origdata, normtests = "cvm")
head(res2$normalized)
res2$parameters
res2$normtests

res3 <- osktnorm(origdata, normtests = c("cvm","sw","ppm"))
head(res3$normalized)
res3$parameters
res3$normtests

res4 <- osktnorm(origdata, normtests = "all")
print(res4$normtests)
```

`pearsonp` *Pearson P Statistic for Normality Check*

Description

Computes the Pearson P metric for assessing deviation from normality. The statistic is defined as the Pearson Chi-square goodness-of-fit statistic divided by its degrees of freedom. This scaled form is used as a normality metric rather than a formal hypothesis test.

Usage

```
pearsonp(x, nbins = NULL)
```

Arguments

`x`	A numeric vector of observations. Missing values are removed prior to computation.
`nbins`	Optional integer specifying the number of equal-probability bins. If `NULL` (default), the number of bins is set to $\lfloor n^{2/5} \rfloor$, following the default choice in `nortest::pearson.test`.

Details

The data are standardized using the sample mean and standard deviation. The standardized values are then grouped into equal-probability bins defined by the quantiles of the standard normal distribution.

Let P denote the Pearson Chi-square statistic and $df = k - 3$ the degrees of freedom, accounting for estimation of the mean and variance. The Pearson P metric is defined as

$$P/df$$

.

Unlike `nortest::pearson.test`, this function does not compute a p-value and should be interpreted as a descriptive normality metric. Smaller values indicate closer agreement with normality.

Value

An object of class `"htest"` containing:

`statistic`	The Pearson P metric (P/df).
`method`	A character string describing the metric.
`data.name`	The name of the input data.
`df`	Degrees of freedom used in the scaling.

Author(s)

Zeynel Cebeci

Examples

```
set.seed(28)
x <- rnorm(100)
res <- pearsonp(x)
res$statistic

set.seed(42)
x <- rlnorm(100)
pearsonp(x)$statistic

res <- pearsonp(x, nbins = 8)
res$statistic
```

phenodata *Morphological and Agronomic Phenotype Data of Rice (Oryza sativa)*

Description

A comprehensive dataset containing 37 morphological, agronomic, and quality traits measured across 193 rice genotypes. The data covers flowering times across different locations, seed morphology, and grain quality parameters.

Usage

```
data(phenodata)
```

Format

A data frame with 193 observations on the following 37 variables:

`IID` Character vector; Individual Identifier.

`FTA` Numeric; Flowering time at Arkansas.

`FTF` Integer; Flowering time at Faridpur.

`FTB` Integer; Flowering time at Aberdeen.

`RTA` Numeric; Flowering time ratio of Arkansas/Aberdeen.

`RTF` Numeric; Flowering time ratio of Faridpur/Aberdeen.

`CULM` Numeric; Culm habit (stem growth pattern).

`LPUB` Integer; Leaf pubescence (0: absent, 1: present).

`FLL` Numeric; Flag leaf length.

`FLW` Numeric; Flag leaf width.

`AWN` Integer; Awn presence (0: absent, 1: present).

`PNP` Numeric; Panicle number per plant.

`PHT` Numeric; Plant height.

`PLEN` Numeric; Panicle length.

`PPBN` Numeric; Primary panicle branch number.

`SNPP` Numeric; Seed number per panicle.

`FLPP` Numeric; Florets per panicle.

`PFRT` Numeric; Panicle fertility.

`SDL` Numeric; Seed length.

`SDW` Numeric; Seed width.

`SDV` Numeric; Seed volume.

`SDSA` Numeric; Seed surface area.

`BRL` Numeric; Brown rice seed length.

`BRW`	Numeric; Brown rice seed width.
`BRSA`	Numeric; Brown rice surface area.
`BRV`	Numeric; Brown rice volume.
`SLWR`	Numeric; Seed length/width ratio.
`BLWR`	Numeric; Brown rice length/width ratio.
`SCOL`	Integer; Seed color.
`PCOL`	Integer; Pericarp color.
`STRH`	Numeric; Straighthead susceptibility.
`BLST`	Integer; Blast resistance score.
`AMY`	Numeric; protlose content.
`ASV`	Numeric; Alkali spreading value.
`PROT`	Numeric; Protein content.
`Y07A`	Numeric; Year 2007 flowering time at Arkansas.
`Y06A`	Numeric; Year 2006 flowering time at Arkansas.

Details

The dataset is subject to an `omit` action for missing values, with several genotypes excluded due to incomplete phenotypic records. These traits are essential for quantitative trait loci (QTL) mapping and genome-wide association studies (GWAS) in rice diversity research.

Source

A reduced version of data, obtained from the Rice Diversity Project. Original phenotypic and genotypic data are available at `http://www.ricediversity.org/data/`.

References

Zhao, K., Tung, C. W., Eizenga, G. C., Wright, M. H., Ali, M. L., Price, A. H., ... & McCouch, S. R. (2011). Genome-wide association mapping reveals a rich genetic architecture of complex traits in Oryza sativa. *Nature Communications*, 2(1), 467. doi:10.1038/ncomms1467

Examples

```
data(phenodata)
# Summary of plant height across the population
summary(phenodata$PHT)

# Correlation between Seed Length and Brown Rice Seed Length
plot(phenodata$SDL, phenodata$BRL,
     xlab = "Seed Length", ylab = "Brown Rice Length",
     main = "Seed Morphology Correlation")

# Normalize protein data
prot <- phenodata$PROT
prot <- as.matrix(prot[!is.na(prot)])
```

```
sw_before <- shapiro.test(prot)

prot_oskt <- osktfast(prot)$transformed
sw_after <- shapiro.test(prot_oskt)

oldpar <- par(mfrow = c(1, 2))
hist(prot,
     breaks = 20,
     col = "lightgreen",
     main = "Original",
     xlab = "Values")

hist(prot_oskt,
     breaks = 20,
     col = "skyblue",
     main = "OSKT Normalized",
     xlab = "Transformed Values")
par(oldpar)

oldpar <- par(mfrow = c(1, 1))
print("Shapiro-Wilk Test (Before OSKT)")
print(sw_before)

print("\nShapiro-Wilk Test (After OSKT)")
print(sw_after)
par(oldpar)
```

---

rewbc — *Box-Cox Transformation Using Reweighted Maximum Likelihood*

Description

Performs a robust reweighted maximum likelihood estimation of the Box-Cox (RBC) transformation parameter for univariate data, following the methodology of Raymaekers and Rousseeuw (2024). The procedure aims at achieving central normality by iteratively downweighting outlying observations using Huber-type weights.

Usage

```
rewbc(x, lrange = seq(-3, 3, by = 0.01), rwsteps = 2, k = 1.5)
```

Arguments

`x`	A numeric vector of observations. Missing values are removed. If non-positive values are present, the data are automatically shifted to ensure positivity.
`lrange`	A numeric vector specifying the grid of candidate Box-Cox transformation parameters λ over which the (weighted) log-likelihood is maximized.

`rwsteps`	An integer specifying the number of iterative reweighting steps used in the reweighted maximum likelihood procedure.
`k`	The tuning constant for the Huber-weight function applied to the standardized transformed data. Smaller values lead to stronger downweighting of extreme observations.

Details

The function first computes the classical maximum likelihood estimate (MLE) of the Box-Cox transformation parameter assuming normality of the transformed data.

In subsequent iterations, the data are transformed using the current estimate of λ. Robust estimates of location and scale (median and MAD) are used to compute standardized residuals, from which Huber-type weights are derived. These weights are then used to re-maximize the Box-Cox log-likelihood over the specified grid of λ values.

The weighted log-likelihood maximized at each step is given by

$$\ell(\lambda) = -\frac{n}{2}\log(\sigma^2) + (\lambda - 1)\sum_{i=1}^{n}\log(x_i),$$

where σ^2 denotes the (possibly weighted) variance of the transformed data, and robustness enters through the estimation of σ^2 via observation weights.

Value

A list with the following components:

`transformed`	The Box-Cox transformed data using the final estimated λ.
`lambda`	The estimated Box-Cox transformation parameter.
`weights`	The final Huber-type weights assigned to each observation.
`steps`	The number of reweighting iterations performed.

Author(s)

Zeynel Cebeci

References

Raymaekers, J., & Rousseeuw, P. J. (2024). Transforming variables to central normality. *Machine Learning*, 113(8), 4953–4975.

See Also

`boxcox`, `yeojohnson`, `abc`, `rewyj`, `osktnorm`

Examples

```
# Generate non-normal data
set.seed(123)
x <- c(rnorm(90), rnorm(10, mean = 5))
head(x)
shapiro.test(x)

# Reweigted Box-Cox with estimated lambda
res <- rewbc(x)
res$lambda
head(res$transformed)
shapiro.test(res$transformed)
hist(res$transformed, main = "Reweighted Box-Cox Transformed Data")

# Reweigted Box-Cox with specified lambda
res2 <- rewbc(x, lrange = c(-1, 0.5, 1))
res2$lambda
head(res2$transformed)
shapiro.test(res2$transformed)
```

`rewyj` *Yeo-Johnson Transformation Using Reweighted Maximum Likelihood*

Description

Performs a robust, reweighted maximum likelihood estimation of the Yeo-Johnson transformation parameter for univariate data. Outliers are downweighted using Huber-type weights in an iteratively reweighted likelihood framework.

Usage

```
rewyj(x, lrange = seq(-3, 3, by = 0.01), rwsteps = 2, k = 1.5)
```

Arguments

`x`	A numeric vector of observations. Missing values are removed. The data may contain both positive and negative values.
`lrange`	A numeric vector specifying the grid of candidate λ values over which the Yeo-Johnson log-likelihood is maximized.
`rwsteps`	An integer specifying the number of reweighting iterations in the iteratively reweighted maximum likelihood procedure.
`k`	Tuning constant for the Huber weight function (Huber, 1981). Larger values reduce robustness, while smaller values increase downweighting of extreme observations.

Details

The function implements the reweighted maximum likelihood (RewML) approach for the Yeo-Johnson transformation (Yeo &Johnson, 2000) as described by Raymaekers and Rousseeuw (2024).

In the first step, the classical maximum likelihood estimate (MLE) of the Yeo-Johnson transformation parameter λ is obtained under a normality assumption.

Subsequently, the algorithm iteratively:

1. Transforms the data using the current estimate of λ.
2. Computes robust location and scale estimates using the median and median absolute deviation (MAD).
3. Standardizes the transformed data and computes Huber-type weights.
4. Re-maximizes a weighted Yeo–Johnson log-likelihood over the specified grid of λ values.

The Jacobian term of the Yeo-Johnson transformation is included unweighted, following the formulation in Raymaekers and Rousseeuw (2024).

The weighted log-likelihood has the form

$$\ell(\lambda) = -\frac{n}{2}\log(\sigma^2) + \sum_{i=1}^{n} g_\lambda(x_i),$$

where σ^2 is the weighted variance of the transformed data and $g_\lambda(x)$ denotes the Jacobian contribution of the Yeo-Johnson transformation.

Value

A list with the following components:

`transformed`	The Yeo-Johnson transformed data using the estimated λ.
`lambda`	The estimated Yeo–Johnson transformation parameter.
`weights`	Final robust weights assigned to each observation.
`steps`	Number of reweighting iterations performed.

Author(s)

Zeynel Cebeci

References

Raymaekers, J., & Rousseeuw, P. J. (2024). Transforming variables to central normality. *Machine Learning*, 113(8), 4953-4975.

Yeo, I.-K. & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4), 954-959. doi:10.1093/biomet/87.4.954.

Huber, P. J. (1981). *Robust Statistics*. Wiley.

See Also

`yeojohnson`, `boxcox`, `abc`, `rewbc`, `osktnorm`

Examples

```
# Generate non-normal data
set.seed(123)
x <- c(rnorm(90), rnorm(10, mean = 5))
head(x)
shapiro.test(x)

# Reweigted Yeo-Johnson with estimated lambda
res <- rewyj(x)
res$lambda
head(res$transformed)
shapiro.test(res$transformed)

# Reweigted Yeo-Johnson with specified lambda
res2 <- rewyj(x, lrange = c(-1, 0.5, 1))
res2$lambda
head(res2$transformed)
shapiro.test(res2$transformed)
hist(res2$transformed, main = "Reweighted Yeo-Johnson Transformed Data")
```

`rjbtest` *Gel-Gastwirth Robust Jarque-Bera Test*

Description

Performs the Gel-Gastwirth robust version of the Jarque-Bera normality test using quantile-based measures of skewness and kurtosis. The test is designed to reduce sensitivity to outliers by avoiding moment-based estimators.

Usage

```
rjbtest(x)
```

Arguments

`x`	A numeric vector of observations. Missing values are removed prior to computation.

Details

The classical Jarque–Bera test relies on moment-based estimates of skewness and kurtosis, making it highly sensitive to outliers.

The Gel–Gastwirth robust Jarque–Bera (RJB) test replaces these moments with robust, quantile-based measures.

Robust skewness is measured using the Bowley skewness:

$$\hat{\gamma}_1^{(R)} = \frac{Q_{0.75} + Q_{0.25} - 2Q_{0.50}}{Q_{0.75} - Q_{0.25}},$$

where Q_p denotes the empirical p-quantile.

Robust kurtosis is measured using the Moors kurtosis (excess form):

$$\hat{\gamma}_2^{(R)} = \frac{(Q_{0.875} - Q_{0.625}) + (Q_{0.375} - Q_{0.125})}{Q_{0.75} - Q_{0.25}} - 3.$$

The robust Jarque–Bera test statistic is defined as

$$\mathrm{RJB} = \frac{n}{6}\left(\hat{\gamma}_1^{(R)}\right)^2 + \frac{n}{24}\left(\hat{\gamma}_2^{(R)}\right)^2.$$

Under the null hypothesis of normality, the statistic is asymptotically distributed as a chi-squared distribution with 2 degrees of freedom.

Value

An object of class `"htest"` containing the following components:

`statistic`	The value of the robust Jarque–Bera test statistic.
`p.value`	The asymptotic p-value computed from the chi-squared distribution with 2 degrees of freedom.
`method`	A character string describing the test.
`data.name`	A character string giving the name of the data.

Author(s)

Zeynel Cebeci

References

Gel, Y. R. and Gastwirth, J. L. (2008). A robust modification of the Jarque–Bera test of normality. *Economics Letters*, 99(1), 30-32.

Examples

```
# Generate normal distributed data
set.seed(123)
x <- rnorm(150)

result <- rjbtest(x)
result$statistic
result$p.value

# Generate positively skewed example data
set.seed(123)
x <- rlnorm(150, meanlog = 0, sdlog = 1)

result <- rjbtest(x)
result$statistic
result$p.value
```

`yeojohnson` *Yeo-Johnson Transformation*

Description

Performs a Yeo-Johnson transformation (Yeo & Johnson, 2000) on a numeric vector. The transformation estimates the optimal lambda via maximum likelihood if not provided. Optionally, the transformed data can be standardized.

Usage

```
yeojohnson(x, lambda = NULL, standardize = TRUE, eps = 1e-6)
```

Arguments

`x`	A numeric vector to transform.
`lambda`	Optional numeric value of lambda for the Yeo-Johnson transformation. If `NULL` (default), lambda is estimated via maximum likelihood.
`standardize`	Logical. If `TRUE` (default), the transformed values are centered and scaled to have mean 0 and standard deviation 1.
`eps`	Numeric tolerance used to handle cases where lambda is approximately 0 or 2. Default is `1e-6`.

Details

The Yeo-Johnson transformation is a generalization of the Box-Cox transformation that can handle both positive and negative values:

$$y(\lambda) = \begin{cases} ((x+1)^{\lambda} - 1)/\lambda, & x \geq 0, \lambda \neq 0 \\ \log(x+1), & x \geq 0, \lambda = 0 \\ -((-x+1)^{2-\lambda} - 1)/(2-\lambda), & x < 0, \lambda \neq 2 \\ -\log(-x+1), & x < 0, \lambda = 2 \end{cases}$$

If `lambda` is not specified, it is estimated via maximum likelihood over a grid of values from -3 to 3 (step 0.01). Standardization is optional and centers the transformed data at mean 0 with standard deviation 1.

Value

A list with the following components:

`transformed`	The transformed numeric vector.
`lambda`	The lambda value used in the transformation (either provided or estimated via MLE if not defined).

Author(s)

Zeynel Cebeci

References

Yeo, I.-K. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4), 954-959. doi:10.1093/biomet/87.4.954.

See Also

`abc`, `boxcox`, `rewbc`, `rewyj`, `osktnorm`

Examples

```
# Generate log-normal data
set.seed(123)
x <- rlnorm(50)
head(x)
shapiro.test(x)

# Yeo-Johnson with estimated lambda
res <- yeojohnson(x)
res$lambda
head(res$transformed)
shapiro.test(res$transformed)

# Yeo-Johnson with specified lambda
res2 <- yeojohnson(x, lambda = -1)
res2$lambda
head(res2$transformed)
shapiro.test(res2$transformed)

# Standardization turned off
res3 <- yeojohnson(x, standardize = FALSE)
res3$lambda
head(res3$transformed)
shapiro.test(res3$transformed)
```

`zatest` *Zhang-Wu ZA Test for Normality*

Description

Performs the Zhang-Wu ZA test for assessing normality. The test is based on a weighted empirical distribution function statistic and uses Monte Carlo simulation to obtain p-values under the null hypothesis of normality with unknown mean and variance.

Usage

```
zatest(x, nsim = 10000, eps = 1e-10, ncores = 1, seed = NULL)
```

Arguments

`x`	A numeric vector of observations. Missing and non-finite values are removed prior to computation.
`nsim`	Number of Monte Carlo simulations used to approximate the null distribution of the test statistic. Larger values improve accuracy at the cost of increased computation time.
`eps`	A small positive constant used to truncate probability values away from 0 and 1 to ensure numerical stability in logarithmic computations.
`ncores`	Number of CPU cores to be used for parallel Monte Carlo simulation. Must be a positive integer. The default `ncores = 1` disables parallel computation.
`seed`	Optional integer value used to set the random number generator seed for reproducibility of the Monte Carlo simulations. If `NULL`, the seed is not set.

Details

Let $x_{(1)} \leq \cdots \leq x_{(n)}$ denote the ordered sample. The data are standardized using the sample mean and standard deviation, and the standard normal cumulative distribution function is evaluated at the standardized observations.

The Zhang–Wu ZA test statistic is defined as

$$ZA = -\sum_{i=1}^{n} \left[\frac{\log(F(x_{(i)}))}{n - i + 0.5} + \frac{\log(1 - F(x_{(i)}))}{i - 0.5} \right],$$

where F denotes the standard normal cumulative distribution function.

Because the null distribution of the statistic depends on estimated parameters, asymptotic critical values are not available. Instead, p-values are obtained via Monte Carlo simulation under the null hypothesis by repeatedly generating samples from a normal distribution, re-estimating the mean and variance, and recomputing the test statistic.

Parallel computation is supported via the `ncores` argument.

Value

An object of class `"htest"` containing the following components:

`statistic`	The observed value of the ZA test statistic.
`p.value`	Monte Carlo p-value for the test.
`method`	A character string describing the test.
`data.name`	A character string giving the name of the data.

Author(s)

Zeynel Cebeci

References

Zhang, J. & Wu, Y. (2005). Likelihood-ratio tests for normality. *Computational Statistics & Data Analysis*, 49(3), 709-721. doi:10.1016/j.csda.2004.05.034.

Examples

```
# Normal data
set.seed(123)
x <- rnorm(50)
resx <- zatest(x, nsim=100)
resx$statistic # Test statistic
resx$p.value

# Log-normal data (non-normal)
set.seed(123)
y <- rlnorm(50, meanlog = 0, sdlog = 1)
resy <- zatest(y, nsim=100)
resy$statistic
resy$p.value

# Exponential data (non-normal)
set.seed(123)
w <- rexp(50)
resw <- zatest(w, nsim=100, ncores=1)
resw$p.value

# Parallel execution using multiple CPU cores

 z <- rt(100, 5,2)
 cores <- 2
 resz <- zatest(z, nsim = 10000, ncores = cores)
 resz$statistic
 resz$p.value
```

`zctest` *Zhang-Wu ZC Test for Normality*

Description

Performs the Zhang-Wu ZC goodness-of-fit test for assessing normality. The test is based on a likelihood-ratio type statistic proposed by Zhang (2002)<10.1111/1467-9868.00337> and further discussed by Zhang and Wu (2005)<10.1016/j.csda.2004.05.034>. The p-value is obtained using a Monte Carlo procedure with parameter re-estimation.

Usage

```
zctest(x, nsim = 10000, eps = 1e-10, ncores = 1, seed = NULL)
```

Arguments

`x`	A numeric vector of observations. Missing and non-finite values are removed prior to computation.

`nsim`	Number of Monte Carlo simulations used to approximate the null distribution of the test statistic. Larger values yield more accurate p-values at the expense of increased computation time.
`eps`	A small positive constant used to truncate normal probabilities away from 0 and 1 to ensure numerical stability in logarithmic computations.
`ncores`	Number of CPU cores to be used for parallel Monte Carlo simulation. The default value `1` disables parallel execution. Values less than 1 or greater than the number of available cores result in an error.
`seed`	Optional integer value used to set the random number generator seed for reproducibility of the Monte Carlo simulations. If `NULL`, the seed is not set.

Details

Let $x_1, \ldots, x_n$ denote the observed data. The data are standardized using the sample mean and standard deviation, and the ordered standardized values $z_{(i)}$ are transformed to normal scores $p_i = \Phi(z_{(i)})$, where Φ denotes the standard normal distribution function.

The ZC test statistic is defined as

$$\mathrm{ZC} = \sum_{i=1}^{n} \left[\log \left(\frac{1/p_i - 1}{(n-0.5)/(i-0.75) - 1} \right) \right]^2 .$$

Because the finite-sample null distribution of the statistic is not available in closed form, the p-value is computed using a Monte Carlo procedure. In each simulation, a normal sample of size n is generated, standardized using its own sample mean and standard deviation, and the ZC statistic is recomputed.

The Monte Carlo p-value is computed using the unbiased estimator

$$p = \frac{1 + \sum_{b=1}^{B} I(T_b \geq T_{\text{obs}})}{B+1},$$

where T_{obs} is the observed test statistic and T_b are the simulated statistics.

To ensure numerical stability, probabilities are truncated to lie in the interval $(\varepsilon, 1-\varepsilon)$.

Value

An object of class `"htest"` with the following components:

`statistic`	The value of the ZC test statistic.
`p.value`	Monte Carlo p-value for the test.
`method`	A character string describing the test.
`data.name`	A character string giving the name of the data.

Author(s)

Zeynel Cebeci

References

Zhang, J. (2002). Powerful goodness-of-fit tests based on the likelihood ratio. *Journal of the Royal Statistical Society: Series B*, 64, 281-294. doi:10.1111/14679868.00337.

Zhang, J. & Wu, Y. (2005). Likelihood-ratio tests for normality. *Computational Statistics & Data Analysis*, 49, 709-721. doi:10.1016/j.csda.2004.05.034.

Examples

```
# Normal data
set.seed(123)
x <- rnorm(50)
resx <- zctest(x, nsim=100)
resx$statistic # Test statistic
resx$p.value

# Log-normal data (non-normal)
set.seed(123)
y <- rlnorm(50, meanlog = 0, sdlog = 1)
resy <- zctest(y, nsim=100)
resy$statistic
resy$p.value

# Exponential data (non-normal)
set.seed(123)
w <- rexp(50)
resw <- zctest(w, nsim=100, ncores=1)
resw$p.value

# Parallel execution using multiple CPU cores

 z <- rt(100, 5,2)
 cores <- 2
 resz <- zctest(z, nsim = 10000, ncores = cores)
 resz$statistic
 resz$p.value
```

Index