Package ‘power.transform’

April 12, 2025

Title Location and Scale Invariant Power Transformations

Version 1.0.1

Description Location- and scale-invariant Box-Cox and Yeo-Johnson power transformations allow for transforming variables with distributions distant from 0 to normality. Transformers are implemented as S4 objects. These allow for transforming new instances to normality after optimising fitting parameters on other data. A test for central normality allows for rejecting transformations that fail to produce a suitably normal distribution, independent of sample number.

URL https://github.com/oncoray/power.transform

BugReports https://github.com/oncoray/power.transform/issues

License EUPL

Encoding UTF-8

RoxygenNote 7.3.2

Imports data.table, methods, rlang (>= 1.0.0), nloptr

Collate 'TransformationObjects.R' 'TransformationYeoJohnson.R' 'TransformationBoxCox.R' 'AccessorsMutatorsLambda.R' 'AccessorsMutatorsScale.R' 'AccessorsMutatorsShift.R' 'AccessorsTransformationMethod.R' 'Checks.R' 'FindParameters.R' 'GoodnessOfFit.R' 'ParameterEstimators.R' 'ParameterEstimatorEDF.R' 'ParameterEstimatorMLE.R' 'ParameterEstimatorRaymaekers.R' 'ParameterEstimatorSkewnessKurtosis.R' 'PlotQQPlot.R' 'PlotResidualPlot.R' 'PlotUtilities.R' 'TransformationSkeleton.R' 'Utilities.R' 'WeightingFunctions.R' 'WeightingFunctionParameters.R' 'power.transform-package.R'

Suggests ggplot2 (>= 3.4.0), testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5)

NeedsCompilation no

Author Alex Zwanenburg [aut, cre] (<https://orcid.org/0000-0002-0342-9545>),
Steffen Löck [aut],
German Cancer Research Center (DKFZ) [cph]

Maintainer Alex Zwanenburg <alexander.zwanenburg@nct-dresden.de>

Repository CRAN

Date/Publication 2025-04-12 14:30:02 UTC

Contents

`assess_transformation` *Assess normality of transformed data*

Description

Not all data allows for a reasonable transformation to normality using power transformation. For example, uniformly distributed data or multi-modal data cannot be transformed to normality. This function computes a p-value for an empirical goodness of fit test for central normality. A distribution is centrally normal if the central 80% of the data are approximately normally distributed. The null-hypothesis is that the transformed distribution is centrally normal.

Usage

```
assess_transformation(x, transformer, verbose = TRUE, ...)
```

Arguments

`x`	A vector with numeric values that should be transformed to normality.
`transformer`	A transformer object created using `find_transformation_parameters`.
`verbose`	Sets verbosity of the fubction.
`...`	Unused arguments.

Details

This function is a wrapper around `ecn.test`.

Value

p-value for empirical goodness of fit test.

Examples

```
x <- exp(stats::rnorm(1000))
transformer <- find_transformation_parameters(
  x = x,
  method = "box_cox")

assess_transformation(
  x = x,
  transformer = transformer)
```

`cn.test` *Central normality test*

Description

Assesses central normality of input data.

Usage

```
cn.test(x, transformer = NULL, robust = FALSE)
```

Arguments

`x`	vector of input data, of at least length 5.
`transformer`	A transformer object created using `find_transformation_parameters`. Optional, if present residuals are determined from x after transformation.
`robust`	Determines the use of the strict test for central normality (`FALSE`) or the more robust version where test statistics where determined from normally distributed data with 10% outliers (`TRUE`).

Value

list with mean absolute error (`tau`), critical value (at significance level = 0.95) of the test statistic (`tau_critical`) and p-value (`p_value`) for (empirical) central normality test.

`create_transformer_skeleton` *Create transformation object skeleton*

Description

Creates skeleton objects. This generates objects without fitting parameters. This is primarily intended for creating transformers externally, where fitting parameters are known.

Usage

```
create_transformer_skeleton(method, lambda = 1, shift = 0, scale = 1)
```

Arguments

`method`	Transformation method. Can be `"none"`, `"box_cox"` or `"yeo_johnson"`.
`lambda`	Value of the transformation parameter lambda. Can also be changed using the `set_lambda` method.
`shift`	Value of the shift parameter. Can also be changed using the `set_shift` method.
`scale`	Value of the scale parameter. Can also be changed using the `set_scale` method.

Value

A transformer object

`ecn.test` *Empirical central normality test*

Description

Assesses central normality of input data using an empirical test.

Usage

```
ecn.test(x, transformer = NULL)
```

Arguments

`x`	vector of input data, of at least length 5.
`transformer`	A transformer object created using `find_transformation_parameters`. Optional, if present residuals are determined from x after transformation.

Value

list with mean absolute error (`tau`), critical value (at significance level = 0.95) of the test statistic (`tau_critical`) and p-value (`p_value`) for the empirical central normality test.

`find_transformation_parameters` *Set transformation parameters*

Description

`find_transformation_parameters` is used to find optimal parameters for univariate transformation to normality.

Usage

```
find_transformation_parameters(
  x,
  method = "yeo_johnson",
  robust = TRUE,
  invariant = TRUE,
  lambda = c(-4, 6),
  empirical_gof_normality_p_value = NULL,
  ...
)
```

Arguments

`x`	A vector with numeric values.
`method`	One of the following methods for power transformation: • `box_cox`: Transformation using the Box-Cox transformation (Box and Cox, 1964). The Box-Cox transformation requires that all data are strictly positive. Features that contain zero or negative values cannot be transformed using this transformation. In their work, Box and Cox define a shifted variant. We use this variant to shift values to a strictly positive range, when negative values are present. The Box-Cox transformation relies on a single parameter lambda, which is estimated through maximisation of the log-likelihood function corresponding to a normal distribution. • `yeo_johnson`:Transformation using the Yeo-Johnson transformation (Yeo and Johnson, 2000). Unlike the Box-Cox transformation, the Yeo-Johnson transformation allows for negative and positive values. Like the Box-Cox transformation, this transformation relies on a single parameter lambda, which is estimated through maximisation of the log-likelihood function corresponding to a normal distribution. • `none`: A fall-back method that will not transform values.

`robust`	Flag for using a robust version of Box-Cox or Yeo-Johnson transformation, as defined by Raymaekers and Rousseeuw (2021). This version is less sensitive in the presence outliers.
`invariant`	Flag for using a version of Box-Cox or Yeo-Johnson transformation that simultaneously optimises location and scale in addition to the lambda parameter.
`lambda`	Single lambda value, or range of lambda values that should be considered. Default: c(4.0, 6.0). Can be `NULL` to force optimisation without a constraint in lambda values.
`empirical_gof_normality_p_value`	Significance value for the empirical goodness-of-fit test for central normality. The p-value is computed through the `assess_transformation` function. By setting this parameter to a numeric value other than `NULL`, the transformation will be rejected when the p-value of the test is below the significance value.
`...`	Unused parameters.

Value

A transformer object that can be used to transform values.

References

1. Yeo, I. & Johnson, R. A. A new family of power transformations to improve normality or symmetry. Biometrika 87, 954–959 (2000).
2. Box, G. E. P. & Cox, D. R. An analysis of transformations. J. R. Stat. Soc. Series B Stat. Methodol. 26, 211–252 (1964).
3. Raymaekers, J., Rousseeuw, P. J. Transforming variables to central normality. Mach Learn. (2021).

Examples

```
x <- exp(stats::rnorm(1000))
transformer <- find_transformation_parameters(
  x = x,
  method = "box_cox")
```

`get_lambda` *Get lambda value*

Description

Get the lambda value of a transformer object.

Usage

```
get_lambda(object, ...)

## S4 method for signature 'transformationPowerTransform'
get_lambda(object, ...)

## S4 method for signature 'transformationBoxCox'
get_lambda(object, ...)

## S4 method for signature 'transformationYeoJohnson'
get_lambda(object, ...)
```

Arguments

`object`	Transformer object
`...`	Unused arguments

Value

Lambda value of the transformer.

`get_residuals` *Compute residuals of transformation to normality*

Description

Compute residuals of transformation to normality

Usage

```
get_residuals(x, transformer, ...)
```

Arguments

`x`	A vector with numeric values that should be transformed to normality.
`transformer`	A transformer object created using `find_transformation_parameters`.
`...`	Unused arguments.

Value

A `data.table` containing the expected (according to a normal distribution) and observed z-scores, and their difference as residuals.

Examples

```
x <- exp(stats::rnorm(1000))
transformer <- find_transformation_parameters(
  x = x,
  method = "box_cox")

residual_data <- get_residuals(
  x = x,
  transformer = transformer)
```

get_scale *Get scale value*

Description

Get the scale value of a transformer object.

Usage

```
get_scale(object, ...)

## S4 method for signature 'transformationPowerTransform'
get_scale(object, ...)

## S4 method for signature 'transformationBoxCox'
get_scale(object, ...)

## S4 method for signature 'transformationYeoJohnson'
get_scale(object, ...)
```

Arguments

`object`	Transformer object
`...`	Unused arguments

Value

scale value of the transformer.

get_shift *Get shift value*

Description

Get the shift value of a transformer object.

Usage

```
get_shift(object, ...)

## S4 method for signature 'transformationPowerTransform'
get_shift(object, ...)

## S4 method for signature 'transformationBoxCox'
get_shift(object, ...)

## S4 method for signature 'transformationYeoJohnson'
get_shift(object, ...)
```

Arguments

`object`	Transformer object
`...`	Unused arguments

Value

shift value of the transformer.

get_transformation_method *Get transformation method*

Description

Get the transformation method of a transformer object.

Usage

```
get_transformation_method(object, ...)

## S4 method for signature 'transformationPowerTransform'
get_transformation_method(object, ...)
```

Arguments

`object`	Transformer object
`...`	Unused arguments

Value

Transformation method

`huber_estimate` *Huber M-estimate*

Description

Iteratively computes M-estimates for location and scale. These are robust estimates of the mean and standard deviation of the data.

Usage

```
huber_estimate(x, k = 1.28, tol = 1e-04)
```

Arguments

`x`	Vector of numeric values for which the location and scale should be estimated.
`k`	Numeric value > 0 that the determines the value beyond which the signal is winsorized.
`tol`	Tolerance for the iterative procedure.

Value

list with location estimate `"mu"` and scale estimate `"sigma"`.

`plot_qq_plot` *Create Q-Q plot*

Description

Create a figure that plots the expected, theoretical normal quantiles (z-scores) against the observed normal quantiles (z-scores) of the data.

Usage

```
plot_qq_plot(
  x,
  transformer,
  show_original = TRUE,
  show_identity = TRUE,
  use_alpha = TRUE,
  ggtheme = NULL
)
```

Arguments

`x`	A vector with numeric values that should be transformed to normality.
`transformer`	A transformer object created using `find_transformation_parameters`.
`show_original`	Show quantiles for original, untransformed, data in addition to transformed data.
`show_identity`	Show identity line that indicates equivalence between expected and observed quantiles.
`use_alpha`	Use transparency for points in case the data contains many instances.
`ggtheme`	`ggplot2` theme to use for the plot. If not provided, `ggplot2::theme_light` is used.

Value

A `ggplot2` plot object for a Q-Q plot.

Examples

```
x <- exp(stats::rnorm(1000))
transformer <- find_transformation_parameters(
  x = x,
  method = "box_cox")

if (rlang::is_installed("ggplot2")) {
  plot_qq_plot(
    x = x,
    transformer = transformer
  )
}
```

`plot_residual_plot` *Create residual plot*

Description

Create a figure that plots the residuals of the data. These residuals are the difference between expected normal quantiles and observed quantiles.

Usage

```
plot_residual_plot(
  x,
  transformer,
  centre_width = NULL,
  show_original = TRUE,
  use_alpha = TRUE,
  use_absolute_deviation = TRUE,
  ggtheme = NULL
)
```

Arguments

`x`	A vector with numeric values that should be transformed to normality.
`transformer`	A transformer object created using `find_transformation_parameters`.
`centre_width`	A numeric value between 0.0 and 1.0 that describes the width of the centre of the data. Can be NULL.
`show_original`	Show residuals for original, untransformed, data in addition to transformed data.
`use_alpha`	Use transparency for points in case the data contains many instances.
`use_absolute_deviation`	Plot absolute deviation instead of residuals.
`ggtheme`	`ggplot2` theme to use for the plot. If not provided, `ggplot2::theme_light` is used.

Value

A `ggplot2` plot object for a Q-Q plot.

Examples

```
x <- exp(stats::rnorm(1000))
transformer <- find_transformation_parameters(
  x = x,
  method = "box_cox"
)

if (rlang::is_installed("ggplot2")) {
  plot_residual_plot(
    x = x,
    transformer = transformer
  )

  # Plot only central 80% of the data.
  plot_residual_plot(
    x = x,
    transformer = transformer,
    centre_width = 0.80,
    show_original = FALSE
    )
  }
```

`power.transform` *power.transform: Transform Data to Normality using Power Transformations*

Description

This package was originally based on, and contains code from, the familiar package (https://cran.r-project.org/package=familiar), under the EUPL license.

Author(s)

Maintainer: Alex Zwanenburg `<alexander.zwanenburg@nct-dresden.de>` (ORCID)

Authors:

- Steffen Löck

Other contributors:

- German Cancer Research Center (DKFZ) [copyright holder]

See Also

Useful links:

- https://github.com/oncoray/power.transform
- Report bugs at https://github.com/oncoray/power.transform/issues

---

`power_transform` *Transform values*

Description

`power_transform` transforms numeric values to normality.

Usage

```
power_transform(x, transformer = NULL, oob_action = "na", ...)
```

Arguments

`x`	A vector with numeric values that should be transformed to normality.
`transformer`	A transformer object created using `find_transformation_parameters`. If `NULL`, a transformer is generated internally.
`oob_action`	Action that should be taken when out-of-bounds values are encountered in `x`. This can for example be 0 or negative values for Box-Cox transformations. • `na` (default): replaces out-of-bounds values by NA values. • `valid`: replaces out-of-bounds values by the closest valid boundary values. This argument has no effect for Yeo-Johnson transformations.
`...`	Arguments passed on to `find_transformation_parameters`

`method` One of the following methods for power transformation:

- `box_cox`: Transformation using the Box-Cox transformation (Box and Cox, 1964). The Box-Cox transformation requires that all data are strictly positive. Features that contain zero or negative values cannot be transformed using this transformation. In their work, Box and Cox define a shifted variant. We use this variant to shift values to a strictly positive range, when negative values are present. The Box-Cox transformation relies on a single parameter lambda, which is estimated through maximisation of the log-likelihood function corresponding to a normal distribution.
- `yeo_johnson`:Transformation using the Yeo-Johnson transformation (Yeo and Johnson, 2000). Unlike the Box-Cox transformation, the Yeo-Johnson transformation allows for negative and positive values. Like the Box-Cox transformation, this transformation relies on a single parameter lambda, which is estimated through maximisation of the log-likelihood function corresponding to a normal distribution.
- `none`: A fall-back method that will not transform values.

`robust` Flag for using a robust version of Box-Cox or Yeo-Johnson transformation, as defined by Raymaekers and Rousseeuw (2021). This version is less sensitive in the presence outliers.

`invariant` Flag for using a version of Box-Cox or Yeo-Johnson transformation that simultaneously optimises location and scale in addition to the lambda parameter.

`lambda` Single lambda value, or range of lambda values that should be considered. Default: c(4.0, 6.0). Can be `NULL` to force optimisation without a constraint in lambda values.

`empirical_gof_normality_p_value` Significance value for the empirical goodness-of-fit test for central normality. The p-value is computed through the `assess_transformation` function. By setting this parameter to a numeric value other than `NULL`, the transformation will be rejected when the p-value of the test is below the significance value.

Value

A vector of transformed values of `x`.

See Also

find_transformation_parameters

Examples

```
x <- exp(stats::rnorm(1000))
y <- power_transform(
  x = x,
  method = "box_cox")
```

`ragn` *Random Values from the Asymmetric Generalised Normal Distribution*

Description

Draws random values from an asymmetric generalised normal distribution.

Usage

```
ragn(n, location = 0, scale = 1, alpha = 0.5, beta = 2)
```

Arguments

`n`	number of instances
`location`	central location of the distribution
`scale`	scale of the distribution. Must be strictly positive: `scale > 0.0`
`alpha`	value between 0.0 and 1.0 that determines the skewness of the distribution. `alpha > 0.5` creates a distribution with a negative skew (left-skewed), i.e. the left tail of the distribution is elongated, and the bulk of the distribution is located to the right. `alpha < 0.5` creates a distribution with a positive skew (right-skewed), i.e. the right tail of the distribution is elongated, and the bulk of the distribution is located to the left. For `alpha = 0.0`, the distribution does not have a skew.
`beta`	Strictly positive value (`beta > 0.0`) that determines the overall shape of the generalised normal distribution. For `beta = 1`, an asymmetric Laplace distribution is used. `beta = 2` draws values according to an asymmetric normal distribution. For large `beta` the distribution will approximate the uniform distribution.

Details

Random values drawn according to an asymmetric generalised normal distribution. Here the asymmetric generalised normal distribution is a symmetric general normal distribution, that is made asymmetric using the procedure described by Gijbels et al. To generate random values we use the quantile function of the symmetric generalised normal distribution that was derived by M. Griffin.

The default parameter values produce values as if drawn from the standard normal distribution with $\sigma = \sqrt{2}$, that is, the standard deviation is not $\sqrt{2}$ instead of 1.

Value

One or more numeric values drawn from the asymmetric generalised normal distribution.

References

1. Gijbels I, Karim R, Verhasselt A. Quantile Estimation in a Generalized
2. Griffin M (2018). gnorm: Generalized Normal/Exponential Power Distribution.

Examples

```
# Draw values from a standard normal distribution.
x <- power.transform::ragn(n = 10000, scale = 1/sqrt(2))
hist(x, 50)

# Draw values from a left-skewed normal distribution.
x <- power.transform::ragn(n = 10000, scale = 1/sqrt(2), alpha = 0.8)
hist(x, 50)

# Draw values from a right-skewed normal distribution.
x <- power.transform::ragn(n = 10000, scale = 1/sqrt(2), alpha = 0.2)
hist(x, 50)

# Draw values from a standard laplace distribution.
x <- power.transform::ragn(n = 10000, scale = 1/sqrt(2), beta = 1.0)
hist(x, 50)
```

`revert_power_transform` *Revert transformation*

Description

`revert_power_transform` reverts the transformation of numeric values to normality.

Usage

```
revert_power_transform(y, transformer)
```

Arguments

`y`	A vector with numeric values that was previously transformed to normality.
`transformer`	A transformer object created using `find_transformation_parameters` that was used to transform the values to normality previously. Cannot be `NULL`.

Value

A vector of values.

Examples

```
x0 <- exp(stats::rnorm(1000))

transformer <- find_transformation_parameters(
  x = x0,
  method = "box_cox")

y <- power_transform(
  x = x0,
  transformer = transformer)

x1 <- revert_power_transform(
  y = y,
  transformer = transformer)
```

`set_lambda` *Set lambda value*

Description

Set the lambda value of a transformer object.

Usage

```
set_lambda(object, lambda, ...)

## S4 method for signature 'transformationPowerTransform'
set_lambda(object, lambda, ...)

## S4 method for signature 'transformationBoxCox'
set_lambda(object, lambda, ...)

## S4 method for signature 'transformationYeoJohnson'
set_lambda(object, lambda, ...)
```

Arguments

`object`	Transformer object
`lambda`	Lambda value
`...`	Unused arguments

Value

Transformer object with updated lambda value.

`set_scale` *Set scale value*

Description

Set the scale value of a transformer object.

Usage

```
set_scale(object, scale, ...)

## S4 method for signature 'transformationPowerTransform'
set_scale(object, scale, ...)

## S4 method for signature 'transformationBoxCox'
set_scale(object, scale, ...)

## S4 method for signature 'transformationYeoJohnson'
set_scale(object, scale, ...)
```

Arguments

`object`	Transformer object
`scale`	scale value
`...`	Unused arguments

Value

Transformer object with updated scale value.

`set_shift` *Set shift value*

Description

Set the shift value of a transformer object.

Usage

```
set_shift(object, shift, ...)

## S4 method for signature 'transformationPowerTransform'
set_shift(object, shift, ...)

## S4 method for signature 'transformationBoxCox'
```

```
set_shift(object, shift, ...)

## S4 method for signature 'transformationYeoJohnson'
set_shift(object, shift, ...)
```

Arguments

`object`	Transformer object
`shift`	Shift value
`...`	Unused arguments

Value

Transformer object with updated shift value.

---

transformationBoxCox-class *Box-Cox transformation object*

Description

This class is used for Box-Cox transformations.

Slots

`method` Main transformation method, i.e. `"box_cox"`.

`robust` Indicates whether a robust version of the Box-Cox transformation is used to set transformation parameters. The value depends on the `robust` argument of the `find_transformation_parameters` function.

`lambda` Numeric lambda parameter for the Box-Cox transformation.

`shift` Numeric shift parameter for the Box-Cox transformation. The value depends on the data used for setting transformation parameters. If all data are strictly positive, `shift` has a value of `0.0`. When negative or zero values are present, data are shifted to be strictly positive. If `invariant=TRUE` in the `find_transformation_parameters` function, `lambda`, `shift` and `scale` parameters are optimised simultaneously.

`scale` Numeric scale parameter for the Box-Cox transformation. If `invariant=TRUE` in the `find_transformation_parame` function, `lambda`, `shift` and `scale` parameters are optimised simultaneously. Otherwise, the `scale` parameter has a value of `1.0`.

`complete` Indicates whether transformation parameters were set.

See Also

find_transformation_parameters

`transformationNone-class` *No transformation object*

Description

This class is for transformers that do not alter the data.

Slots

`method` Main transformation method, i.e. `"none"`.

`complete` Indicates whether transformation parameters were set.

`transformationPowerTransform-class` *Generic transformation object*

Description

This is the superclass for transformation objects.

Slots

`method` Main transformation method.

`complete` Indicates whether transformation parameters were set.

`version` Version of the power.transform package that was used to create the transformation objecst.

`transformationYeoJohnson-class` *Yeo-Johnson transformation object*

Description

This class is used for Yeo-Johnson transformations.

Slots

`method` Main transformation method, i.e. `"yeo_johnson"`.

`robust` Indicates whether a robust version of the Yeo-Johnson transformation is used to set transformation parameters. The value depends on the `robust` argument of the `find_transformation_parameters` function.

`lambda` Numeric lambda parameter for the Yeo-Johnson transformation.

`shift` Numeric shift parameter for the Yeo-Johnson transformation. If `invariant=TRUE` in the `find_transformation_parameters` function, `lambda`, `shift` and `scale` parameters are optimised simultaneously. Otherwise, the `shift` parameter has a value of `0.0`.

`scale` Numeric scale parameter for the Yeo-Johnson transformation. If `invariant=TRUE` in the `find_transformation_parameters` function, `lambda`, `shift` and `scale` parameters are optimised simultaneously. Otherwise, the `scale` parameter has a value of `1.0`.

`complete` Indicates whether transformation parameters were set.

See Also

find_transformation_parameters

Index