# Package 'bestNormalize'

May 7, 2026

**Type** Package

**Title** Normalizing Transformation Functions

**Version** 1.9.2

**Date** 2025-11-29

**Description** Estimate a suite of normalizing transformations, including a new adaptation of a technique based on ranks which can guarantee normally distributed transformed data if there are no ties: ordered quantile normalization (ORQ). ORQ normalization combines a rank-mapping approach with a shifted logit approximation that allows the transformation to work on data outside the original domain. It is also able to handle new data within the original domain via linear interpolation. The package is built to estimate the best normalizing transformation for a vector consistently and accurately. It implements the Box-Cox transformation, the Yeo-Johnson transformation, three types of Lambert WxF transformations, and the ordered quantile normalization transformation. It estimates the normalization efficacy of other commonly used transformations, and it allows users to specify custom transformations or normalization statistics. Finally, functionality can be integrated into a machine learning workflow via recipes.

**URL** https://petersonr.github.io/bestNormalize/, https://github.com/petersonR/bestNormalize

**License** GPL-3

**Depends** R (>= 3.1.0)

**Imports** LambertW (>= 0.6.5), nortest, dplyr, doParallel, foreach, doRNG, recipes, tibble, methods, butcher, purrr, progress, generics

**Suggests** knitr, rmarkdown, MASS, testthat, mgcv, parallel, ggplot2, scales, rlang, covr

**VignetteBuilder** knitr

**LazyData** true

**RoxygenNote** 7.3.3

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Ryan A Peterson [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4650-5798>)

**Maintainer** Ryan A Peterson <ryan-peterson@uiowa.edu>

**Repository** CRAN

**Date/Publication** 2025-11-30 17:10:15 UTC

# Contents



---

`bestNormalize-package` *bestNormalize: Flexibly calculate the best normalizing transformation for a vector*

---

## Description

The `bestNormalize` package provides several normalizing transformations, and introduces a new transformation based off of the order statistics, `orderNorm`. Perhaps the most useful function is `bestNormalize`, which attempts all of these transformations and picks the best one based off of a goodness of fit statistic.

## Author(s)

**Maintainer**: Ryan A Peterson <ryan-peterson@uiowa.edu> (ORCID)

## See Also

Useful links:

- https://petersonr.github.io/bestNormalize/
- https://github.com/petersonR/bestNormalize

---

`arcsinh_x` *arcsinh(x) Transformation*

---

## Description

Perform a arcsinh(x) transformation

## Usage

```
arcsinh_x(x, standardize = TRUE, ...)

## S3 method for class 'arcsinh_x'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'arcsinh_x'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A vector to normalize with with x |
| `standardize` | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| `...` | additional arguments |
| `object` | an object of class 'arcsinh_x' |
| `newdata` | a vector of data to be (potentially reverse) transformed |
| `inverse` | if TRUE, performs reverse transformation |

## Details

`arcsinh_x` performs an arcsinh transformation in the context of bestNormalize, such that it creates a transformation that can be estimated and applied to new data via the `predict` function.

The function is explicitly: log(x + sqrt(x^2 + 1))

## Value

A list of class `arcsinh_x` with elements

| | |
|---|---|
| `x.t` | transformed original data |
| `x` | original data |
| `mean` | mean after transformation but prior to standardization |
| `sd` | sd after transformation but prior to standardization |
| `n` | number of nonmissing observations |
| `norm_stat` | Pearson's P / degrees of freedom |
| `standardize` | was the transformation standardized |

The `predict` function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## Examples

```
x <- rgamma(100, 1, 1)

arcsinh_x_obj <- arcsinh_x(x)
arcsinh_x_obj
p <- predict(arcsinh_x_obj)
x2 <- predict(arcsinh_x_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

---

# `autotrader` *Prices of 6,283 cars listed on Autotrader*

---

## Description

A dataset containing the prices and other attributes of over 6000 cars in the Minneapolis area.

## Usage

```
autotrader
```

## Format

A data frame with 6283 rows and 10 variables:

**price** price, in US dollars

**Car_Info** Raw description from website

**Link** hyperlink to listing (must be appended to https://www.autotrader.com/)

**Make** Car manufacturer

**Year** Year car manufactured

**Location** Location of listing

**Radius** Radius chosen for search

**mileage** mileage on vehicle

**status** used/new/certified

**model** make and model, separated by space

## Source

https://www.autotrader.com/

---

`bestLogConstant` *Calculate and perform best normalizing log transformation (experimental)*

---

## Description

Similar to bestNormalize, this selects the best candidate constant for a log transformation on the basis of the Pearson P test statistic for normality. The transformation that has the lowest P (calculated on the transformed data) is selected. This function is currently in development and may not behave as expected.

See details for more information.

## Usage

```
bestLogConstant(x, a, standardize = TRUE, ...)

## S3 method for class 'bestLogConstant'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'bestLogConstant'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A vector to normalize |
| `a` | (optional) a list of candidate constants to choose from |
| `standardize` | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal. This will not change the normality statistic. |
| `...` | additional arguments. |
| `object` | an object of class 'bestLogConstant' |
| `newdata` | a vector of data to be (reverse) transformed |
| `inverse` | if TRUE, performs reverse transformation |

## Details

`bestLogConstant` estimates the optimal normalizing constant for a log transformation. This transformation can be performed on new data, and inverted, via the `predict` function.

## Value

A list of class `bestLogConstant` with elements

| | |
|---|---|
| `x.t` | transformed original data |
| `x` | original data |
| `norm_stats` | Pearson's Pearson's P / degrees of freedom |
| `method` | out-of-sample or in-sample, number of folds + repeats |
| `chosen_constant` | the chosen constant transformation (of class 'log_x') |
| `other_transforms` | the other transformations (of class 'log_x') |

The `predict` function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## See Also

`bestNormalize`, `log_x`,

---

# bestNormalize *Calculate and perform best normalizing transformation*

---

## Description

Performs a suite of normalizing transformations, and selects the best one on the basis of the Pearson P test statistic for normality. The transformation that has the lowest P (calculated on the transformed data) is selected. See details for more information.

## Usage

```
bestNormalize(
  x,
  standardize = TRUE,
  allow_orderNorm = TRUE,
  allow_lambert_s = FALSE,
  allow_lambert_h = FALSE,
  allow_exp = TRUE,
  out_of_sample = TRUE,
  cluster = NULL,
  k = 10,
  r = 5,
```

```
  loo = FALSE,
  warn = FALSE,
  quiet = FALSE,
  tr_opts = list(),
  new_transforms = list(),
  norm_stat_fn = NULL,
  ...
)

## S3 method for class 'bestNormalize'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'bestNormalize'
print(x, ...)

## S3 method for class 'bestNormalize'
tidy(x, ...)
```

## Arguments

| | |
|---|---|
| x | A ‘bestNormalize‘ object. |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal. This will not change the normality statistic. |
| allow_orderNorm | set to FALSE if orderNorm should not be applied |
| allow_lambert_s | Set to FALSE if the lambertW of type "s" should not be applied (see details). Expect about 2-3x elapsed computing time if TRUE. |
| allow_lambert_h | Set to TRUE if the lambertW of type "h" and "hh" should be applied (see details). Expect about 2-4x elapsed computing time. |
| allow_exp | Set to TRUE if the exponential transformation should be applied (sometimes this will cause errors with heavy right skew) |
| out_of_sample | if FALSE, estimates quickly in-sample performance |
| cluster | name of cluster set using `makeCluster` |
| k | number of folds |
| r | number of repeats |
| loo | should leave-one-out CV be used instead of repeated CV? (see details) |
| warn | Should bestNormalize warn when a method doesn’t work? |
| quiet | Should a progress-bar not be displayed for cross-validation progress? |
| tr_opts | a list (of lists), specifying options to be passed to each transformation (see details) |
| new_transforms | a named list of new transformation functions and their predict methods (see details) |

| | |
|---|---|
| norm_stat_fn | if specified, a function to calculate to assess normality (default is the Pearson chi-squared statistic divided by its d.f.) |
| ... | not used |
| object | an object of class 'bestNormalize' |
| newdata | a vector of data to be (reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

bestNormalize estimates the optimal normalizing transformation. This transformation can be performed on new data, and inverted, via the predict function.

This function currently estimates the Yeo-Johnson transformation, the Box Cox transformation (if the data is positive), the log_10(x+a) transformation, the square-root (x+a) transformation, and the arcsinh transformation. a is set to max(0, -min(x) + eps) by default. If allow_orderNorm == TRUE and if out_of_sample == FALSE then the ordered quantile normalization technique will likely be chosen since it essentially forces the data to follow a normal distribution. More information on the orderNorm technique can be found in the package vignette, or using ?orderNorm.

Repeated cross-validation is used by default to estimate the out-of-sample performance of each transformation if out_of_sample = TRUE. While this can take some time, users can speed it up by creating a cluster via the parallel package's makeCluster function, and passing the name of this cluster to bestNormalize via the cl argument. For best performance, we recommend the number of clusters to be set to the number of repeats r. Care should be taken to account for the number of observations per fold; too small a number and the estimated normality statistic could be inaccurate, or at least suffer from high variability.

As of version 1.3, users can use leave-one-out cross-validation as well for each method by setting loo to TRUE. This will take a lot of time for bigger vectors, but it will have the most accurate estimate of normalization efficacy. Note that if this method is selected, arguments k, r are ignored. This method will still work in parallel with the cl argument.

Note that the Lambert transformation of type "h" or "hh" can be done by setting allow_lambert_h = TRUE, however this can take significantly longer to run.

Use tr_opts in order to set options for each transformation. For instance, if you want to overide the default a selection for log_x, set tr_opts$log_x = list(a = 1).

See the package's vignette on how to use custom functions with bestNormalize. All it takes is to create an S3 class and predict method for the new transformation and load it into the environment, then the new custom function (and its predict method) can be passed to bestNormalize with new_transform.

## Value

A list of class bestNormalize with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| norm_stats | Pearson's Pearson's P / degrees of freedom |
| method | out-of-sample or in-sample, number of folds + repeats |

`chosen_transform`
: the chosen transformation (of appropriate class)

`other_transforms`
: the other transformations (of appropriate class)

`oos_preds`
: Out-of-sample predictions (if loo == TRUE) or normalization stats

The `predict` function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## See Also

boxcox, orderNorm, yeojohnson

## Examples

```
x <- rgamma(100, 1, 1)

## Not run:
# With Repeated CV
BN_obj <- bestNormalize(x)
BN_obj
p <- predict(BN_obj)
x2 <- predict(BN_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)

## End(Not run)


## Not run:
# With leave-one-out CV
BN_obj <- bestNormalize(x, loo = TRUE)
BN_obj
p <- predict(BN_obj)
x2 <- predict(BN_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)

## End(Not run)

# Without CV
BN_obj <- bestNormalize(x, allow_orderNorm = FALSE, out_of_sample = FALSE)
BN_obj
p <- predict(BN_obj)
x2 <- predict(BN_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

`binarize` *Binarize*

## Description

This function will perform a binarizing transformation, which could be used as a last resort if the data cannot be adequately normalized. This may be useful when accidentally attempting normalization of a binary vector (which could occur if implementing bestNormalize in an automated fashion).

Note that the transformation is not one-to-one, in contrast to the other functions in this package.

## Usage

```
binarize(x, location_measure = "median")

## S3 method for class 'binarize'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'binarize'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A vector to binarize |
| `location_measure` | which location measure should be used? can either be "median", "mean", "mode", a number, or a function. |
| `object` | an object of class 'binarize' |
| `newdata` | a vector of data to be (reverse) transformed |
| `inverse` | if TRUE, performs reverse transformation |
| `...` | additional arguments |

## Value

A list of class `binarize` with elements

| | |
|---|---|
| `x.t` | transformed original data |
| `x` | original data |
| `method` | location_measure used for original fitting |
| `location` | estimated location_measure |
| `n` | number of nonmissing observations |
| `norm_stat` | Pearson's P / degrees of freedom |

The `predict` function with `inverse = FALSE` returns the numeric value (0 or 1) of the transformation on `newdata` (which defaults to the original data).

If `inverse = TRUE`, since the transform is not 1-1, it will create and return a factor that indicates where the original data was cut.

## Examples

```
x <- rgamma(100, 1, 1)
binarize_obj <- binarize(x)
(p <- predict(binarize_obj))

predict(binarize_obj, newdata = p, inverse = TRUE)
```

---

boxcox *Box-Cox Normalization*

---

## Description

Perform a Box-Cox transformation and center/scale a vector to attempt normalization

## Usage

```
boxcox(x, standardize = TRUE, ...)

## S3 method for class 'boxcox'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'boxcox'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with Box-Cox |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| ... | Additional arguments that can be passed to the estimation of the lambda parameter (lower, upper, epsilon) |
| object | an object of class 'boxcox' |
| newdata | a vector of data to be (reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

boxcox estimates the optimal value of lambda for the Box-Cox transformation. This transformation can be performed on new data, and inverted, via the predict function.

The function will return an error if a user attempt to transform nonpositive data.

## Value

A list of class boxcox with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| lambda | estimated lambda value for skew transformation |
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | was the transformation standardized |

The predict function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. Journal of the Royal Statistical Society B, 26, 211-252.

## See Also

boxcox

## Examples

```
x <- rgamma(100, 1, 1)

bc_obj <- boxcox(x)
bc_obj
p <- predict(bc_obj)
x2 <- predict(bc_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

---

# double_reverse_log *Double Reverse Log(x + a) Transformation*

---

## Description

First reverses scores, then perform a log_b(x) normalization transformation, and then reverses scores again.

## Usage

```
double_reverse_log(
  x,
  b = 10,
  standardize = TRUE,
  eps = diff(range(x, na.rm = TRUE))/10,
  warn = TRUE,
  ...
)

## S3 method for class 'double_reverse_log'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'double_reverse_log'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with with x |
| b | The base of the log (defaults to 10) |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| eps | The cushion for the transformation range (defaults to 10 percent) |
| warn | Should a warning result from infinite values? |
| ... | additional arguments |
| object | an object of class 'double_reverse_log' |
| newdata | a vector of data to be (potentially reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

`double_reverse_log` performs a simple log transformation in the context of bestNormalize, such that it creates a transformation that can be estimated and applied to new data via the `predict` function. The parameter a is essentially estimated by the training set by default (estimated as the minimum possible to some extent epsilon), while the base must be specified beforehand.

## Value

A list of class `double_reverse_log` with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| b | estimated base b value |

| | |
|---|---|
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | was the transformation standardized |

The predict function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## Examples

```
x <- rgamma(100, 1, 1)

double_reverse_log_obj <- double_reverse_log(x)
double_reverse_log_obj
p <- predict(double_reverse_log_obj)
x2 <- predict(double_reverse_log_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

---

# exp_x *exp(x) Transformation*

---

## Description

Perform a exp(x) transformation

## Usage

```
exp_x(x, standardize = TRUE, warn = TRUE, ...)

## S3 method for class 'exp_x'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'exp_x'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with with x |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| warn | Should a warning result from infinite values? |
| ... | additional arguments |
| object | an object of class 'exp_x' |
| newdata | a vector of data to be (potentially reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

exp_x performs a simple exponential transformation in the context of bestNormalize, such that it creates a transformation that can be estimated and applied to new data via the predict function.

## Value

A list of class exp_x with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | was the transformation standardized |

The predict function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## Examples

```
x <- rgamma(100, 1, 1)

exp_x_obj <- exp_x(x)
exp_x_obj
p <- predict(exp_x_obj)
x2 <- predict(exp_x_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

---

# lambert *Lambert W x F Normalization*

---

## Description

Perform Lambert's W x F transformation and center/scale a vector to attempt normalization via the LambertW package.

## Usage

```
lambert(x, type = "s", standardize = TRUE, warn = FALSE, ...)

## S3 method for class 'lambert'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'lambert'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with Box-Cox |
| type | a character indicating which transformation to perform (options are "s", "h", and "hh", see details) |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| warn | should the function show warnings |
| ... | Additional arguments that can be passed to the LambertW::Gaussianize function |
| object | an object of class 'lambert' |
| newdata | a vector of data to be (reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

`lambert` uses the `LambertW` package to estimate a normalizing (or "Gaussianizing") transformation. This transformation can be performed on new data, and inverted, via the `predict` function.

NOTE: The type = "s" argument is the only one that does the 1-1 transform consistently, and so it is the only method currently used in `bestNormalize()`. Use type = "h" or type = 'hh' at risk of not having this estimate 1-1 transform. These alternative types are effective when the data has exceptionally heavy tails, e.g. the Cauchy distribution.

Additionally, sometimes (depending on the distribution) this method will be unable to extrapolate beyond the observed bounds. In these cases, NaN is returned.

## Value

A list of class `lambert` with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| tau.mat | estimated parameters of LambertW::Gaussianize |
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | was the transformation standardized |

The `predict` function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## References

Georg M. Goerg (2016). LambertW: An R package for Lambert W x F Random Variables. R package version 0.6.4.

Georg M. Goerg (2011): Lambert W random variables - a new family of generalized skewed distributions with applications to risk estimation. Annals of Applied Statistics 3(5). 2197-2230.

Georg M. Goerg (2014): The Lambert Way to Gaussianize heavy-tailed data with the inverse of Tukey's h transformation as a special case. The Scientific World Journal.

## See Also

Gaussianize

## Examples

```
## Not run:
x <- rgamma(100, 1, 1)

lambert_obj <- lambert(x)
lambert_obj
p <- predict(lambert_obj)
x2 <- predict(lambert_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)

## End(Not run)
```

---

# log_x *Log(x + a) Transformation*

---

## Description

Perform a log_b (x+a) normalization transformation

## Usage

```
log_x(x, a = NULL, b = 10, standardize = TRUE, eps = 0.001, warn = TRUE, ...)

## S3 method for class 'log_x'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'log_x'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with with x |
| a | The constant to add to x (defaults to max(0, -min(x) + eps)); see bestLogConstant |
| b | The base of the log (defaults to 10) |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| eps | The allowed error in the expression for the selected a |
| warn | Should a warning result from infinite values? |
| ... | additional arguments |
| object | an object of class 'log_x' |
| newdata | a vector of data to be (potentially reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

log_x performs a simple log transformation in the context of bestNormalize, such that it creates a transformation that can be estimated and applied to new data via the predict function. The parameter a is essentially estimated by the training set by default (estimated as the minimum possible to some extent epsilon), while the base must be specified beforehand.

## Value

A list of class log_x with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| a | estimated a value |
| b | estimated base b value |
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | was the transformation standardized |

The predict function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## Examples

```
x <- rgamma(100, 1, 1)

log_x_obj <- log_x(x)
log_x_obj
p <- predict(log_x_obj)
x2 <- predict(log_x_obj, newdata = p, inverse = TRUE)
```

```
all.equal(x2, x)
```

---

`no_transform` *Identity transformation and center/scale transform*

---

## Description

Perform an identity transformation. Admittedly it seems odd to have a dedicated function to essentially do I(x), but it makes sense to keep the same syntax as the other transformations so it plays nicely with them. As a benefit, the bestNormalize function will also show a comparable normalization statistic for the untransformed data. If `standardize == TRUE`, `center_scale` passes to bestNormalize instead.

## Usage

```
no_transform(x, warn = TRUE, ...)

## S3 method for class 'no_transform'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'no_transform'
print(x, ...)

center_scale(x, warn = TRUE, ...)

## S3 method for class 'center_scale'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'center_scale'
print(x, ...)

## S3 method for class 'no_transform'
tidy(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A ‘no_transform‘ object. |
| `warn` | Should a warning result from infinite values? |
| `...` | not used |
| `object` | an object of class ’no_transform’ |
| `newdata` | a vector of data to be (potentially reverse) transformed |
| `inverse` | if TRUE, performs reverse transformation |

## Details

`no_transform` creates a identity transformation object that can be applied to new data via the `predict` function.

## Value

A list of class `no_transform` with elements

| | |
|---|---|
| `x.t` | transformed original data |
| `x` | original data |
| `n` | number of nonmissing observations |
| `norm_stat` | Pearson's P / degrees of freedom |

The `predict` function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## Examples

```
x <- rgamma(100, 1, 1)

no_transform_obj <- no_transform(x)
no_transform_obj
p <- predict(no_transform_obj)
x2 <- predict(no_transform_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

---

# orderNorm *Calculate and perform Ordered Quantile normalizing transformation*

---

## Description

The Ordered Quantile (ORQ) normalization transformation, `orderNorm()`, is a rank-based procedure by which the values of a vector are mapped to their percentile, which is then mapped to the same percentile of the normal distribution. Without the presence of ties, this essentially guarantees that the transformation leads to a uniform distribution.

The transformation is:

$$g(x) = \Phi^{-1}((rank(x) - .5)/(length(x)))$$

Where $\Phi$ refers to the standard normal cdf, rank(x) refers to each observation's rank, and length(x) refers to the number of observations.

By itself, this method is certainly not new; the earliest mention of it that I could find is in a 1947 paper by Bartlett (see references). This formula was outlined explicitly in Van der Waerden, and expounded upon in Beasley (2009). However there is a key difference to this version of it, as explained below.

Using linear interpolation between these percentiles, the ORQ normalization becomes a 1-1 transformation that can be applied to new data. However, outside of the observed domain of x, it is unclear how to extrapolate the transformation. In the ORQ normalization procedure, a binomial glm with a logit link is used on the ranks in order to extrapolate beyond the bounds of the original domain of x. The inverse normal CDF is then applied to these extrapolated predictions in order to extrapolate the transformation. This mitigates the influence of heavy-tailed distributions while preserving the 1-1 nature of the transformation. The extrapolation will provide a warning unless warn = FALSE.) However, we found that the extrapolation was able to perform very well even on data as heavy-tailed as a Cauchy distribution (paper to be published).

The fit used to perform the extrapolation uses a default of 10000 observations (or length(x) if that is less). This added approximation improves the scalability, both computationally and in terms of memory used. Do not set this value to be too low (e.g. <100), as there is no benefit to doing so. Increase if your test data set is large relative to 10000 and/or if you are worried about losing signal in the extremes of the range.

This transformation can be performed on new data and inverted via the `predict` function.

## Usage

```
orderNorm(x, n_logit_fit = min(length(x), 10000), ..., warn = TRUE)

## S3 method for class 'orderNorm'
predict(object, newdata = NULL, inverse = FALSE, warn = TRUE, ...)

## S3 method for class 'orderNorm'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A vector to normalize |
| `n_logit_fit` | Number of points used to fit logit approximation |
| `...` | additional arguments |
| `warn` | transforms outside observed range or ties will yield warning |
| `object` | an object of class 'orderNorm' |
| `newdata` | a vector of data to be (reverse) transformed |
| `inverse` | if TRUE, performs reverse transformation |

## Value

A list of class `orderNorm` with elements

| | |
|---|---|
| `x.t` | transformed original data |
| `x` | original data |
| `n` | number of nonmissing observations |
| `ties_status` | indicator if ties are present |
| `fit` | fit to be used for extrapolation, if needed |

`norm_stat` Pearson's P / degrees of freedom

The `predict` function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## References

Bartlett, M. S. "The Use of Transformations." Biometrics, vol. 3, no. 1, 1947, pp. 39-52. JSTOR www.jstor.org/stable/3001536.

Van der Waerden BL. Order tests for the two-sample problem and their power. 1952;55:453-458. Ser A.

Beasley TM, Erickson S, Allison DB. Rank-based inverse normal transformations are increasingly used, but are they merited? Behav. Genet. 2009;39(5): 580-595. pmid:19526352

## See Also

`boxcox`, `lambert`, `bestNormalize`, `yeojohnson`

## Examples

```
x <- rgamma(100, 1, 1)

orderNorm_obj <- orderNorm(x)
orderNorm_obj
p <- predict(orderNorm_obj)
x2 <- predict(orderNorm_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

---

# plot.bestNormalize *Transformation plotting*

---

## Description

Plots transformation functions for objects produced by the bestNormalize package

## Usage

```
## S3 method for class 'bestNormalize'
plot(
  x,
  inverse = FALSE,
  bounds = NULL,
  cols = NULL,
  methods = NULL,
  leg_loc = "top",
  ...
)
```

```
## S3 method for class 'orderNorm'
plot(x, inverse = FALSE, bounds = NULL, ...)

## S3 method for class 'boxcox'
plot(x, inverse = FALSE, bounds = NULL, ...)

## S3 method for class 'yeojohnson'
plot(x, inverse = FALSE, bounds = NULL, ...)

## S3 method for class 'lambert'
plot(x, inverse = FALSE, bounds = NULL, ...)
```

## Arguments

| | |
|---|---|
| x | a fitted transformation |
| inverse | if TRUE, plots the inverse transformation |
| bounds | a vector of bounds to plot for the transformation |
| cols | a vector of colors to use for the transforms (see details) |
| methods | a vector of transformations to plot |
| leg_loc | the location of the legend on the plot |
| ... | further parameters to be passed to `plot` and `lines` |

## Details

The plots produced by the individual transformations are simply plots of the original values by the newly transformed values, with a line denoting where transformations would take place for new data.

For the bestNormalize object, this plots each of the possible transformations run by the original call to bestNormalize. The first argument in the "cols" parameter refers to the color of the chosen transformation.

---

# sqrt_x *sqrt(x + a) Normalization*

---

## Description

Perform a sqrt (x+a) normalization transformation

## Usage

```
sqrt_x(x, a = NULL, standardize = TRUE, ...)

## S3 method for class 'sqrt_x'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'sqrt_x'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with with x |
| a | The constant to add to x (defaults to max(0, -min(x))) |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| ... | additional arguments |
| object | an object of class 'sqrt_x' |
| newdata | a vector of data to be (potentially reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

sqrt_x performs a simple square-root transformation in the context of bestNormalize, such that it creates a transformation that can be estimated and applied to new data via the predict function. The parameter a is essentially estimated by the training set by default (estimated as the minimum possible), while the base must be specified beforehand.

## Value

A list of class sqrt_x with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | was the transformation standardized |

The predict function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## Examples

```
x <- rgamma(100, 1, 1)

sqrt_x_obj <- sqrt_x(x)
sqrt_x_obj
p <- predict(sqrt_x_obj)
x2 <- predict(sqrt_x_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

step_best_normalize *Run bestNormalize transformation for* recipes *implementation*

## Description

'step_best_normalize' creates a specification of a recipe step (see 'recipes' package) that will transform data using the best of a suite of normalization transformations estimated (by default) using cross-validation.

## Usage

```
step_best_normalize(
  recipe,
  ...,
  role = NA,
  trained = FALSE,
  transform_info = NULL,
  transform_options = list(),
  num_unique = 5,
  skip = FALSE,
  id = rand_id("best_normalize")
)

## S3 method for class 'step_best_normalize'
tidy(x, ...)

## S3 method for class 'step_best_normalize'
axe_env(x, ...)
```

## Arguments

| | |
|---|---|
| recipe | A formula or recipe |
| ... | One or more selector functions to choose which variables are affected by the step. See [selections()] for more details. For the 'tidy' method, these are not currently used. |
| role | Not used by this step since no new variables are created. |
| trained | For recipes functionality |
| transform_info | A numeric vector of transformation values. This (was transform_info) is 'NULL' until computed by [prep.recipe()]. |
| transform_options | options to be passed to bestNormalize |
| num_unique | An integer where data that have less possible values will not be evaluate for a transformation. |
| skip | For recipes functionality |
| id | For recipes functionality |
| x | A 'step_best_normalize' object. |

## Details

The bestnormalize transformation can be used to rescale a variable to be more similar to a normal distribution. See ‘?bestNormalize‘ for more information; ‘step_best_normalize‘ is the implementation of ‘bestNormalize‘ in the ‘recipes‘ context.

As of version 1.7, the ‘butcher‘ package can be used to (hopefully) improve scalability of this function on bigger data sets.

## Value

An updated version of ‘recipe‘ with the new step added to the sequence of existing steps (if any). For the ‘tidy‘ method, a tibble with columns ‘terms‘ (the selectors or variables selected) and ‘value‘ (the lambda estimate).

## See Also

`bestNormalize` `orderNorm`, [recipe()] [prep.recipe()] [bake.recipe()]

## Examples

```
library(recipes)
rec <- recipe(~ ., data = as.data.frame(iris))

bn_trans <- step_best_normalize(rec, all_numeric())

bn_estimates <- prep(bn_trans, training = as.data.frame(iris))

bn_data <- bake(bn_estimates, as.data.frame(iris))

plot(density(iris[, "Petal.Length"]), main = "before")
plot(density(bn_data$Petal.Length), main = "after")

tidy(bn_trans, number = 1)
tidy(bn_estimates, number = 1)
```

---

# step_orderNorm *ORQ normalization (orderNorm) for* `recipes` *implementation*

---

## Description

‘step_orderNorm‘ creates a specification of a recipe step (see ‘recipes‘ package) that will transform data using the ORQ (orderNorm) transformation, which approximates the "true" normalizing transformation if one exists. This is considerably faster than ‘step_bestNormalize‘.

## Usage

```
step_orderNorm(
  recipe,
  ...,
  role = NA,
  trained = FALSE,
  transform_info = NULL,
  transform_options = list(),
  num_unique = 5,
  skip = FALSE,
  id = rand_id("orderNorm")
)

## S3 method for class 'step_orderNorm'
tidy(x, ...)

## S3 method for class 'step_orderNorm'
axe_env(x, ...)
```

## Arguments

| | |
|---|---|
| recipe | A formula or recipe |
| ... | One or more selector functions to choose which variables are affected by the step. See [selections()] for more details. For the ‘tidy‘ method, these are not currently used. |
| role | Not used by this step since no new variables are created. |
| trained | For recipes functionality |
| transform_info | A numeric vector of transformation values. This (was transform_info) is ‘NULL‘ until computed by [prep.recipe()]. |
| transform_options | options to be passed to orderNorm |
| num_unique | An integer where data that have less possible values will not be evaluate for a transformation. |
| skip | For recipes functionality |
| id | For recipes functionality |
| x | A ‘step_orderNorm‘ object. |

## Details

The orderNorm transformation can be used to rescale a variable to be more similar to a normal distribution. See ‘?orderNorm‘ for more information; ‘step_orderNorm‘ is the implementation of ‘orderNorm‘ in the ‘recipes‘ context.

As of version 1.7, the ‘butcher‘ package can be used to (hopefully) improve scalability of this function on bigger data sets.

## Value

An updated version of ‘recipe‘ with the new step added to the sequence of existing steps (if any). For the ‘tidy‘ method, a tibble with columns ‘terms‘ (the selectors or variables selected) and ‘value‘ (the lambda estimate).

## References

Ryan A. Peterson (2019). Ordered quantile normalization: a semiparametric transformation built for the cross-validation era. Journal of Applied Statistics, 1-16.

## See Also

orderNorm bestNormalize, [recipe()] [prep.recipe()] [bake.recipe()]

## Examples

```
library(recipes)
rec <- recipe(~ ., data = as.data.frame(iris))

orq_trans <- step_orderNorm(rec, all_numeric())

orq_estimates <- prep(orq_trans, training = as.data.frame(iris))

orq_data <- bake(orq_estimates, as.data.frame(iris))

plot(density(iris[, "Petal.Length"]), main = "before")
plot(density(orq_data$Petal.Length), main = "after")

tidy(orq_trans, number = 1)
tidy(orq_estimates, number = 1)
```

---

# yeojohnson *Yeo-Johnson Normalization*

---

## Description

Perform a Yeo-Johnson Transformation and center/scale a vector to attempt normalization

## Usage

```
yeojohnson(x, eps = 0.001, standardize = TRUE, ...)

## S3 method for class 'yeojohnson'
predict(object, newdata = NULL, inverse = FALSE, ...)

## S3 method for class 'yeojohnson'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | A vector to normalize with Yeo-Johnson |
| eps | A value to compare lambda against to see if it is equal to zero |
| standardize | If TRUE, the transformed values are also centered and scaled, such that the transformation attempts a standard normal |
| ... | Additional arguments that can be passed to the estimation of the lambda parameter (lower, upper) |
| object | an object of class 'yeojohnson' |
| newdata | a vector of data to be (reverse) transformed |
| inverse | if TRUE, performs reverse transformation |

## Details

yeojohnson estimates the optimal value of lambda for the Yeo-Johnson transformation. This transformation can be performed on new data, and inverted, via the predict function.

The Yeo-Johnson is similar to the Box-Cox method, however it allows for the transformation of nonpositive data as well. The step_YeoJohnson function in the recipes package is another useful resource (see references).

## Value

A list of class yeojohnson with elements

| | |
|---|---|
| x.t | transformed original data |
| x | original data |
| mean | mean after transformation but prior to standardization |
| sd | sd after transformation but prior to standardization |
| lambda | estimated lambda value for skew transformation |
| n | number of nonmissing observations |
| norm_stat | Pearson's P / degrees of freedom |
| standardize | Was the transformation standardized |

The predict function returns the numeric value of the transformation performed on new data, and allows for the inverse transformation as well.

## References

Yeo, I. K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. Biometrika.

Max Kuhn and Hadley Wickham (2017). recipes: Preprocessing Tools to Create Design Matrices. R package version 0.1.0.9000. https://github.com/topepo/recipes

## Examples

```
x <- rgamma(100, 1, 1)

yeojohnson_obj <- yeojohnson(x)
yeojohnson_obj
p <- predict(yeojohnson_obj)
x2 <- predict(yeojohnson_obj, newdata = p, inverse = TRUE)

all.equal(x2, x)
```

# Index