Package ‘trafo’

November 30, 2017

Title Package for estimation, comparison and selection of transformations

Version 0.0.0.9000

Description Estimation, selection and comparison of several families of transformations. The families of transformations included in the package are the following: Bickel-Docksum, Box-Cox, Dual, Glog, Gpower, Log, Log-shift opt, Manly, Modulus, Neglog, Reciprocal and Yeo-Johnson. The package simplifies to compare linear models with untransformed and transformed dependent variable as well as linear models where the dependent variable is transformed with different transformations. Furthermore, the package employs maximum likelihood approaches, skewness and divergence minimization to estimate the optimal transformation parameter.

Depends R (>= 3.2.4)

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports nlme,
FNN,
moments,
pryr,
graphics,
grDevices,
lmtest

R topics documented:

`as.data.frame.trafo` *Data frame with transformed variables*

Description

The data frame that is returned contains the variables that are used in the model and additionally a variable with the transformed dependent variable. To the variable name of the dependent variable a t is added for transformed.

Usage

```
## S3 method for class 'trafo'
as.data.frame(x, row.names = NULL, optional = FALSE,
  std = TRUE, ...)
```

Arguments

`x`	an object of type `trafo`.
`row.names`	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
`optional`	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's base package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*, check.names = !optional)
`std`	logical. If TRUE, the data is transformed by the standardized transformation. Defaults to `TRUE`.
`...`	other parameters that can be passed to the function.

Value

A data frame with the original variables and the transformed variable.

See Also

`bickeldoksum`, `boxcox`, `dual`, `glog`, `gpower`, `log`, `logshiftopt`, `manly`, `modulus`, `neglog`, `sqrtshift`, `yeojohnson`

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using divergence minimization following
# Kolmogorov-Smirnof
logshiftopt_trafo <- logshiftopt(object = lm_cars, method = "div.ks",
plotit = FALSE)

# Get a data frame with the added transformed variable
as.data.frame(logshiftopt_trafo)
```

`assumptions` *First check of assumptions to find suitable transformations*

Description

Gives a first overview if a transformation is useful and which transformation is promising to fulfill the model assumptions normality, homoscedasticity and linearity.

Usage

```
assumptions(object, method = "ml", std = TRUE, ...)
```

Arguments

`object`	an object of type `lm`.
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`std`	logical. If `TRUE`, the transformed model is returned based on the standardized transformation. Defaults to `TRUE`.
`...`	other parameters that can be passed to the function, e.g. other lambdaranges. For the default values that are used for the lambdaranges see the documentation for the provided transformations.

Value

A table with tests for normality and homoscedasticity. Furthermore, scatterplots are returned to check the linearity assumption.

See Also

`bickeldoksum`, `boxcox`, `dual`, `glog`, `gpower`, `log`, `logshiftopt`, `manly`, `modulus`, `neglog`, `sqrtshift`, `yeojohnson`

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

assumptions(lm_cars)
```

`bickeldoksum` *Bickel-Doksum transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Bickel-Doksum transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
bickeldoksum(object, lambda = "estim", method = "ml",
  lambdarange = c(1e-11, 2), plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. The Bickel-Doksum transformation is only defined for positive values of lambda. Defaults to `c(1e-11, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

References

Bickel PJ, Doksum KA (1981). An analysis of transformations revisited. Journal of the American Statistical Association, Vol. 76, 296-311.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using a maximum likelihood approach
bickeldoksum(object = lm_cars, plotit = FALSE)
```

`boxcox` *Box-Cox transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Box-Cox transformation. The transformation parameter can either be estimated using different estimation methods or given. The Box-Cox transformation is only defined for positive response values. In case the response contains zero or negative values a shift is automatically added such that y + shift > 0.

Usage

```
boxcox(object, lambda = "estim", method = "ml", lambdarange = c(-2, 2),
  plotit = TRUE)
```

Arguments

`object`	an object of type `lm`.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `c(-2, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using skewness minimization
boxcox(object = lm_cars, method = "skew", plotit = FALSE)
```

`diagnostics` *Diagnostics for fitted models*

Description

Returns information about the transformation and selected diagnostics to check model assumptions.

Usage

```
diagnostics(object, ...)
```

Arguments

`object`	an object that contains two models that should be compared.
`...`	other parameters that can be passed to the function.

Value

The return depends on the class of its argument. The documentation of particular methods gives detailed information about the return of that method.

See Also

`diagnostics.trafo_lm`, `diagnostics.trafo_compare`

diagnostics.trafo_compare

Diagnostics for two differently transformed models

Description

Returns information about the applied transformations and selected diagnostics to check model assumptions. Two models are compared where the dependent variable is transformed by different transformations.

Usage

```
## S3 method for class 'trafo_compare'
diagnostics(object, ...)
```

Arguments

`object`	an object of type `trafo_compare`
`...`	additional arguments that are not used in this method

Value

An object of class `diagnostics.trafo_compare`. The method `print.diagnostics.trafo_compare` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform with Bickel-Doksum transformation
bd_trafo <- bickeldoksum(object = lm_cars, plotit = FALSE)

# Transform with Box-Cox transformation
bc_trafo <- boxcox(object = lm_cars, method = "skew", plotit = FALSE)

# Compare transformed models
compare <- trafo_compare(object = lm_cars, trafos = list(bd_trafo, bc_trafo))

# Get diagnostics
diagnostics(compare)
```

`diagnostics.trafo_lm` *Diagnostics for an untransformed and a transformed model*

Description

Returns information about the applied transformation and selected diagnostics to check model assumptions. The return helps to compare the untransformed and the transformed model with regard to model assumptions.

Usage

```
## S3 method for class 'trafo_lm'
diagnostics(object, ...)
```

Arguments

`object`	an object of type `trafo_lm`
`...`	additional arguments that are not used in this method

Value

An object of class `diagnostics.trafo_lm`. The method `print.diagnostics.trafo_lm` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Compare transformed models
BD_lm <- trafo_lm(object = lm_cars, trafo = "bickeldoksum",
method = "skew", lambdarange = c(1e-11, 2))

# Get diagnostics
diagnostics(BD_lm)
```

`dual` *Dual transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Dual transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
dual(object, lambda = "estim", method = "ml", lambdarange = c(0, 2),
  plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. The Dual transformation is not defined for negative values of lambda. Defaults to `c(0, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

References

Yang Z (2006). A modified family of power transformations. Economics Letters, 92(1), 14-19.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using divergence minimization following
# Cramer-von-Mises
dual(object = lm_cars, method = "div.cvm", plotit = TRUE)
```

`glog` *Glog transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Glog transformation.

Usage

```
glog(object)
```

Arguments

`object`	an object of type lm.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable
glog(object = lm_cars)
```

`gpower` *Gpower transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Gpower transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
gpower(object, lambda = "estim", method = "ml", lambdarange = c(-2, 2),
  plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `c(-2, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using divergence minimization following
# Kullback-Leibler
gpower(object = lm_cars, method = "div.kl", plotit = FALSE)
```

`logshiftopt` *Log shift opt transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Log shift opt transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
logshiftopt(object, lambda = "estim", method = "ml", lambdarange = NULL,
  plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `NULL`. In this case the lambdarange is set to the range of the data. In case the lowest value is negative the absolute value of the lowest value plus 1 is the lower bound for the range.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using divergence minimization following
# Kolmogorov-Smirnof
logshiftopt(object = lm_cars, method = "div.ks", plotit = FALSE)
```

`logtrafo` *Log transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Log transformation. The Log transformation is only defined for positive response values. In case the response contains zero or negative values a shift is automatically added such that y + shift > 0.

Usage

```
logtrafo(object)
```

Arguments

`object`	an object of type lm.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable
logtrafo(object = lm_cars)
```

manly — *Manly transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Manly transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
manly(object, lambda = "estim", method = "ml", lambdarange = c(-2, 2),
  plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `c(-2, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

References

Manly BFJ (1976). Exponential data transformations. Journal of the Royal Statistical Society: Series D, Vol. 25, 37-42.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using a maximum likelihood approach
manly(object = lm_cars, plotit = FALSE)
```

modulus *Modulus transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Modulus transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
modulus(object, lambda = "estim", method = "ml", lambdarange = c(-2, 2),
  plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `c(-2, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

References

John JA, Draper NR (1980). An alternative family of transformations. Journal of the Royal Statistical Society: Series C, Vol. 29, 190-197.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable with fixed lambda
modulus(object = lm_cars, lambda = 0.8, plotit = FALSE)
```

`neglog` *Neg log transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Neg log transformation.

Usage

```
neglog(object)
```

Arguments

`object`	an object of type lm.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

References

Whittaker J, Whitehead C, Somers M (2005). The neglog transformation and quantile regression for the analysis of a large credit scoring database. Journal of the Royal Statistical Society. Series C (Applied Statistics), 54(4), 863-878.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable
neglog(object = lm_cars)
```

`plot.trafo_compare` *Plots for linear models with transformed dependent variable*

Description

For the two transformed models a range of plots is returned in order to check model assumptions graphically.

Usage

```
## S3 method for class 'trafo_compare'
plot(x, ...)
```

Arguments

`x`	an object of type `trafo_compare`
`...`	additional arguments that are not used in this method

`plot.trafo_lm` *Plot for regression models with untransformed and transformed dependent variable*

Description

For the untransformed and transformed model a range of plots is returned in order to check model assumptions graphically.

Usage

```
## S3 method for class 'trafo_lm'
plot(x, ...)
```

Arguments

`x`	an object of type `trafo_lm`
`...`	additional arguments that are not used in this method

`print.diagnostics.trafo_compare` *Prints diagnostics of two trafo objects*

Description

Prints diagnostics of two trafo objects.

Usage

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

Arguments

`x`	an object of type `diagnostics.trafo_compare`
`...`	additional arguments that are not used in this method

print.diagnostics.trafo_lm

Prints diagnostics of an untransformed and transformed model

Description

Prints diagnostics of an untransformed and transformed model.

Usage

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

Arguments

`x`	an object of type `diagnostics.trafo_lm`
`...`	additional arguments that are not used in this method

print.summary.trafo_compare

Prints summary of trafo_compare objects

Description

Prints objects to be shown in the summary function for objects of type `trafo_compare`.

Usage

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

Arguments

`x`	an object of type `summary.trafo_compare`
`...`	additional arguments that are not used in this method

print.summary.trafo_lm *Print summary trafo*

Description

prints objects to be shown in the summary function for objects of type `trafo_lm`

Usage

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

Arguments

`x`	an object of type `summary.trafo_lm`
`...`	additional arguments that are not used in this method

print.trafo *Prints object of type trafo*

Description

Prints object of type trafo

Usage

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

Arguments

`x`	an object of type `trafo`.
`...`	other parameters that can be passed to the function.

print.trafo_compare *Prints object of type trafo_compare*

Description

Prints object of type trafo_compare

Usage

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

Arguments

`x`	an object of type trafo_compare.
`...`	other parameters that can be passed to the function.

`print.trafo_lm` *Prints object of type trafo_lm*

Description

Prints object of type trafo_lm

Usage

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

Arguments

`x`	an object of type `trafo_lm`.
`...`	other parameters that can be passed to the function.

`reciprocal` *Reciprocal transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Reciprocal transformation.

Usage

```
reciprocal(object)
```

Arguments

`object`	an object of type lm.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable
reciprocal(object = lm_cars)
```

sqrtshift — *Square-root shift transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Square-root shift transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
sqrtshift(object, lambda = "estim", method = "ml", lambdarange = NULL,
  plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `NULL`. In this case the lambdarange is set to the range of the data. In case the lowest value is negative the absolute value of the lowest value plus 1 is the lower bound for the range.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using a maximum likelihood approach
sqrtshift(object = lm_cars, plotit = TRUE)
```

`summary.trafo_compare` *Summary for two differently transformed models*

Description

The summary contains the summary for two transformed models. The summary is based on the summary for objects of type `lm`.

Usage

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

Arguments

`object`	an object of type `trafo_compare`
`...`	additional arguments that are not used in this method

Value

An object of class `summary.trafo_compare`. The method `print.summary.trafo_compare` can be used for this class.

`summary.trafo_lm` *Summary for linear models with untransformed and transformed dependent variable*

Description

The summary method for class `trafo_lm` contains a summary for an untransformed and a transformed model. The resulting summary is based on the summary for objects of type `lm`.

Usage

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

Arguments

`object`	an object of type `trafo_lm`
`...`	additional arguments that are not used in this method

Value

An object of class `summary.trafo_lm`. The method `print.summary.trafo_lm` can be used for this class.

trafo *An R package supporting the selection of a suitable transformation*

Description

Estimation, selection and comparison of several families of transformations. The families of transformations included in the package are the following: Bickel-Docksum, Box-Cox, Dual, Glog, Gpower, Log, Log-shift opt, Manly, Modulus, Neglog, Reciprocal and Yeo-Johnson. The package simplifies to compare linear models with untransformed and transformed dependent variable as well as linear models where the dependent variable is transformed with different transformations. Furthermore, the package employs maximum likelihood approaches, skewness and divergence minimization to estimate the optimal transformation parameter.

Details

An overview of all currently provided functions can be requested by `library(help=trafo)`.

trafo_compare *Compares linear models with transformed dependent variable*

Description

Function `trafo_compare` compares linear models where the dependent variable is transformed by different transformations.

Usage

```
trafo_compare(object, trafos, std = TRUE)
```

Arguments

`object`	an object of type lm
`trafos`	a list of two `trafo` objects based on the same model given in object.
`std`	logical. If TRUE, the transformed models are returned based on the standardized transformation. Defaults to `TRUE`.

Value

An object of class `trafo_compare`. Methods such as `diagnostics.trafo_compare`, `print.trafo_compare`, `plot.trafo_compare` and `summary.trafo_compare` can be used for this class. @seealso `bickeldoksum`, `boxcox`, `dual`, `glog`, `gpower`, `log`, `logshiftopt`, `manly`, `modulus`, `neglog`, `sqrtshift`, `yeojohnson`

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform with Bickel-Doksum transformation
bd_trafo <- bickeldoksum(object = lm_cars, plotit = FALSE)

# Transform with Box-Cox transformation
bc_trafo <- boxcox(object = lm_cars, method = "skew", plotit = FALSE)

# Compare transformed models
trafo_compare(object = lm_cars, trafos = list(bd_trafo, bc_trafo))
```

`trafo_lm` *Fits transformed linear models*

Description

Function `trafo_lm` fits linear models with transformed dependent variable. The main return are two `lm` objects where one is the untransformed linear model and the other one the transformed linear model.

Usage

```
trafo_lm(object, trafo = "boxcox", lambda = "estim", method = "ml",
  lambdarange = NULL, std = TRUE, custom_trafo = NULL)
```

Arguments

`object`	an object of type `lm`.
`trafo`	a character string. Different transformations can be used for transforming the dependent variable in a linear model: (i) "bickeldoksum", (ii) "boxcox", (iii) "dual", (iv) "glog", (v) "gpower", (vi) "log", (vii) "logshiftopt", (viii) "manly", (ix) "modulus", (x) "neglog", (xi) "reciprocal", (xii) "yeojohnson". Defaults to "boxcox".
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `NULL` which means that the default value of the chosen transformation is used.
`std`	logical. If TRUE, the transformed model is returned based on the standardized transformation. Defaults to `TRUE`.

`custom_trafo`	a list. The list has two elements where the first element is a function specifying the desired transformation and the second element is a function specifying the corresponding standardized transformation. Defaults to `NULL`.

Value

An object of class `trafo_lm`. Methods such as `diagnostics.trafo_lm`, `print.trafo_lm`, `plot.trafo_lm` and `summary.trafo_lm` can be used for this class.

See Also

`bickeldoksum`, `boxcox`, `dual`, `glog`, `gpower`, `log`, `logshiftopt`, `manly`, `modulus`, `neglog`, `sqrtshift`, `yeojohnson`

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Compare untransformed and transformed model
trafo_lm(object = lm_cars, trafo = "bickeldoksum", method = "skew",
lambdarange = c(1e-11, 2))
```

---

trafo_predict — *Predict method for linear models with transformed dependent variable*

Description

The function returns predicted values based on the linear model. The predicted values are back-transformed corresponding to the transformation used in the model. Note that the back-transformation can induce a bias.

Usage

```
trafo_predict(object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
  interval = c("none", "confidence", "prediction"), level = 0.95,
  type = "response", na.action = na.pass, pred.var = res.var/weights,
  weights = 1, ...)
```

Arguments

`object`	an object of type `trafo_lm`.
`newdata`	an optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`se.fit`	a switch indicating if standard errors are required.
`scale`	scale parameter for std. error calculation.
`df`	degrees of freedom for scale.
`interval`	type of interval calculation.

`level`	tolerance/confidence level.
`type`	type of prediction. In this version, only the response can be predicted. Thus, the default is set to "response".
`na.action`	function determining what should be done with missing value in newdata. The default is to predict NA.
`pred.var`	the variance for future observations to be assumed for prediction intervals.
`weights`	variance weights for prediction. This can be a numeric vector or a one-sided model formula. In the latter case, it is interpreted as an expression evaluated in newdata.
`...`	further arguments passed to or from other methods.

Value

A vector of predictions or a matrix of predictions and bounds with column names fit, lwr and upr if interval is set. If se.fit is `TRUE`, a list with the following components is returned: fit, se.fit, residual.scale, df

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Compare untransformed and transformed model
trafo_cars <- trafo_lm(object = lm_cars, trafo = "bickeldoksum", method = "skew",
lambdarange = c(1e-11, 2))

# Get predictions in the back-transformed scale
trafo_predict(trafo_cars)
```

`yeojohnson` *Yeo-Johnson transformation for linear models*

Description

The function transforms the dependent variable of a linear model using the Yeo-Johnson transformation. The transformation parameter can either be estimated using different estimation methods or given.

Usage

```
yeojohnson(object, lambda = "estim", method = "ml", lambdarange = c(-2,
  2), plotit = TRUE)
```

Arguments

`object`	an object of type lm.
`lambda`	either a character named "estim" if the optimal transformation parameter should be estimated or a numeric value determining a given value for the transformation parameter. Defaults to "estim".
`method`	a character string. Different estimation methods can be used for the estimation of the optimal transformation parameter: (i) Maximum likelihood approach ("ml"), (ii) Skewness minimization ("skew"), (iii) Kurtosis optimization ("kurt"), (iv) Divergence minimization by Kolmogorov-Smirnoff ("div.ks"), by Cramer-von-Mises ("div.cvm") or by Kullback-Leibler ("div.kl"). Defaults to "ml".
`lambdarange`	a numeric vector with two elements defining an interval that is used for the estimation of the optimal transformation parameter. Defaults to `c(-2, 2)`.
`plotit`	logical. If TRUE, a plot that illustrates the optimal transformation parameter or the given transformation parameter is returned. Defaults to `TRUE`.

Value

An object of class `trafo`. Methods such as `as.data.frame.trafo` and `print.trafo` can be used for this class.

References

Yeo IK, Johnson RA (2000). A new family of power transformations to improve normality or symmetry. Biometrika, Vol.87, 954-959.

Examples

```
# Load data
data("cars", package = "datasets")

# Fit linear model
lm_cars <- lm(dist ~ speed, data = cars)

# Transform dependent variable using a maximum likelihood approach
yeojohnson(object = lm_cars, plotit = FALSE)
```

Index