# Package 'qualV'

July 2, 2014

**Title** Qualitative Validation Methods

**Version** 0.3

**Author** K.G. van den Boogaart, Stefanie Rost and Thomas Petzoldt

**Description** Qualitative methods for the validation of models.

**Maintainer** Thomas Petzoldt <thomas.petzoldt@tu-dresden.de>

**Depends** R (>= 2.0.0), KernSmooth

**License** GPL (>= 2)

**URL** http://qualV.R-Forge.R-Project.org/

**Repository** CRAN

**Date/Publication** 2011-03-26 15:44:50

**NeedsCompilation** yes

## R topics documented:

qualV-package *Qualitative Validation Methods*

## Description

Qualitative methods for model validation.

## Details

This package contains functions for a qualitative model comparison. Common quantitative deviance measures underestimate the similarity of patterns if there are shifts in time between measurement and simulation. Qualitative validation methods are additional methods to validate models, especially useful to compare the patterns of observed and simulated values.

For a complete list of functions with individual help pages, use `library(help="qualV")`.

## References

Jachner, S., K.G. v.d. Boogaart, T. Petzoldt (2007) Statistical methods for the qualitative assessment of dynamic models with time delay (R package qualV), Journal of Statistical Software, 22(8), 1–30. URL http://www.jstatsoft.org/v22/i08/.

compareME *Compute Several Deviance Measures for Comparison*

## Description

Various deviance measures are computed allowing the user to find the aspects in which two time series differ.

## Usage

```
compareME(o, p,
          o.t      = seq(0, 1, length.out = length(o)),
          p.t      = seq(0, 1, length.out = length(p)),
          ignore   = c("raw", "centered", "scaled", "ordered"),
          geometry = c("real", "logarithmic", "geometric", "ordinal"),
          measure  = c("mad", "var", "sd"),
          type     = "normalized",
          time     = "fixed", ..., col.vars=c("time", "ignore")
         )
## S3 method for class 'compareME'
print(x, ..., digits = 3)
## S3 method for class 'compareME'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| o | vector of observed values, |
| p | vector of predicted values, |
| o.t | vector of observation times, |
| p.t | vector of times for predicted values, |
| ignore | a subset of c("raw", "centered", "scaled", "ordered") as defined in generalME to specify the aspects of the data to be ignored, |
| geometry | a subset of c("real", "logarithmic", "geometric", "ordinal") as defined in generalME to specify the geometry of the observed data, |
| measure | a subset of c("mad", "var", "sd") to specify the type of error to be measured, |
| type | a subset of c("dissimilarity", "normalized", "similarity", "reference") as defined in generalME to specify the type of deviance measure to be used, |
| time | a subset of c("fixed", "transform"), indicates wether the time should actually be transformed. If this argument and the time arguments are missing the comparison is based on values only without time matching. |
| ... | further arguments passed to timeTransME, |
| col.vars | a subset of c("ignore", "geometry", "measure", "time") to be displayed in the columns of the resulting ftable, |
| digits | number of significant digits displayed, |
| x, object | objects of class compareME. |

## Details

The function provides a simple standard interface to get a first idea on the similarities and dissimilarities of two time series spanning the same time interval. The print and summary methods extract the relevant information, rounded to an optional number of significant digits.

## Value

The result is a list of ftables containing the deviance measures of all requested combinations of parameters. The list is done over the different types of measures requested.

## See Also

timeTransME, generalME

## Examples

```
# a constructed example
x <- seq(0, 2*pi, 0.1)
y <- 5 + sin(x)             # a process
o <- y + rnorm(x, sd = 0.2) # observation with random error
p <- y + 0.1                # simulation with systematic bias

os <- ksmooth(x, o, kernel = "normal",
  bandwidth = dpill(x, o), x.points = x)$y
```

```
plot(x, o); lines(x, p); lines(x, os, col = "red")

compareME(o, p)
compareME(os, p)

# observed and measured data with non-matching time intervals
data(phyto)
compareME(obs$y, sim$y, obs$t, sim$t, time = "fixed")
tt <- timeTransME(obs$y, sim$y, obs$t, sim$t, ME = SMSLE, trials = 5)
compareME(tt$yo, tt$yp)

# show names of deviance measures
compareME(type = "name")
```

---

EF *Efficiency Factor as Suggested by Nash \& Sutcliffe*

---

## Description

The efficiency factor is a dimensionless statistic which directly relates predictions to observed data.

## Usage

```
EF(o, p)
```

## Arguments

| | |
|---|---|
| o | vector of observed values |
| p | vector of corresponding predicted values |

## Details

Two time series are compared. `'EF'` is an overall measure of similarity between fitted and observed values. Any model giving a negative value cannot be recommended, whereas values close to one indicate a 'near-perfect' fit.

## Value

| | |
|---|---|
| EF | efficiency factor |

## References

Nash, J. E. and Sutcliffe, J. V. (1970) River flow forecasting through conceptual models part I - A discussion of principles. Journal of Hydrology, 10, 282-290.

Mayer, D. G. and Butler, D. G. (1993) Statistical Validation. Ecological Modelling, 68, 21-32.

## See Also

MAE, MSE, MAPE, GRI

## Examples

```
# a constructed example
x <- seq(0, 2*pi, 0.1)
y <- 5 + sin(x)           # a process
o <- y + rnorm(x, sd=0.2) # observation with random error
p <- y + 0.1              # simulation with systematic bias

plot(x, o); lines(x, p)
EF(o, p)

# observed and measured data with non-matching time intervals
data(phyto)
obsb <- na.omit(obs[match(sim$t, obs$t), ])
simb <- sim[na.omit(match(obs$t, sim$t)), ]
EF(obsb$y, simb$y)
```

---

`features` *Qualitative Features of Time Series*

---

## Description

A time series is characterised by a sequence of characters, indicating features of the time series itself, of its first or second derivative, steepness or level of values.

## Usage

```
f.slope(x, y, f = 0.1, scale = c("mean", "range", "IQR", "sd", "none"))
f.curve(x, y, f = 0.1, scale = c("mean", "range", "IQR", "sd", "none"))
f.steep(x, y, f1 = 1, f2 = 0.1)
f.level(y, high = 0.8, low = 0.2)
```

## Arguments

| | |
|---|---|
| `x` | vector of time |
| `y` | input y values |
| `f` | factor defining the limit for constant (`f.slope`) or linear (`f.curve`) sequences |
| `f1` | factor for the upper bound of steepness |
| `f2` | factor for the lower bound of steepness |
| `scale` | method for internal scaling, `f` is multiplied with mean value, range, interquartile range (IQR) or standard deviation of increments ($abs(\Delta y/\Delta x)$). |
| `high` | lower limit of high values |
| `low` | upper limit of low values |

## Details

For the first derivative the segment between two values is characterised by increasing ('A'), decreasing ('B') or constant ('C') and for the second by convex ('K'), concave ('I') or linear ('J'). For the property of the first derivative the segment between two values is characterised by very steep ('S'), steep ('T') or not steep ('U') or the values are divided into high ('H'), low ('L') or values in between ('M'). Note that for the last two cases the original values and the not increments are standardised (to $[0, 1]$).

## Value

| | |
|---|---|
| v | interval sequence |

## See Also

LCS, qvalLCS

## Examples

```
data(phyto)
bbobs    <- dpill(obs$t, obs$y)
n        <- tail(obs$t, n = 1) - obs$t[1] + 1
obsdpill <- ksmooth(obs$t, obs$y, kernel = "normal", bandwidth = bbobs,
            n.points = n)
obss     <- data.frame(t = obsdpill$x, y = obsdpill$y)
obss     <- obss[match(sim$t, obss$t), ]
f.slope(obss$t, obss$y)
f.curve(obss$t, obss$y)
f.steep(obss$t, obss$y, f1 = 30, f2 = 10)
f.level(obss$y)
```

---

# GRI *A Geometric Reliability Index as Suggested by Leggett \& Williams*

---

## Description

Given a set of predictions and a corresponding set of observations, the geometric validation index is a reliability index for the predictions.

## Usage

```
GRI(o, p)
```

## Arguments

| | |
|---|---|
| o | vector of observed values |
| p | vector of corresponding predicted values |

## Details

One possible interpretation of 'GRI' is that the simulation is accurate within a multiplicative factor 'GRI', i.e. the observed values fall between 1/GRI and GRI times of the corresponding predicted values. Values close to one indicate a good match.

## Value

| | |
|---|---|
| GRI | geometric reliability index |

## References

Leggett, L. R. and Williams, L. R. (1981) A reliability index for models. Ecological Modelling, 13, 303-312.

## See Also

MAE, MSE, MAPE, EF

## Examples

```
# a constructed example
x <- seq(0, 2*pi, 0.1)
y <- 5 + sin(x)             # a process
o <- y + rnorm(x, sd = 0.2) # observation with random error
p <- y + 0.1                # simulation with systematic bias

plot(x, o); lines(x, p)
GRI(o, p)

# observed and measured data with non-matching time intervals
data(phyto)
obsb <- na.omit(obs[match(sim$t, obs$t), ])
simb <- sim[na.omit(match(obs$t, sim$t)), ]
GRI(obsb$y, simb$y)
```

---

# LCS *Algorithm for the Longest Common Subsequence Problem*

---

## Description

Determines the longest common subsequence of two strings.

## Usage

```
LCS(a, b)
```

## Arguments

| | |
|---|---|
| a | vector (numeric or character), missing values are not accepted |
| b | vector (numeric or character), missing values are not accepted |

## Details

A longest common subsequence (LCS) is a common subsequence of two strings of maximum length. The LCS Problem consists of finding a LCS of two given strings and its length (LLCS). The QSI is computed by division of the LLCS over maximum length of 'a' and 'b'.

## Value

| | |
|---|---|
| a | vector 'a' |
| b | vector 'b' |
| LLCS | length of LCS |
| LCS | longest common subsequence |
| QSI | quality similarity index |
| va | one possible LCS of vector 'a' |
| vb | one possible LCS of vector 'b' |

## Note

LCS is now using a C version of the algorithm provided by Dominik Reusser.

## References

Wagner, R. A. and Fischer, M. J. (1974) The String-to-String Correction Problem. Journal of the ACM, 21, 168-173.

Paterson, M. and Dancik, V. (1994) Longest Common Subsequences. Mathematical Foundations of Computer Science, 841, 127-142.

Gusfield, D. (1997) Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, England, ISBN 0-521-58519-8.

## Examples

```
# direct use
a <- c("b", "c", "a", "b", "c", "b")
b <- c("a", "b", "c", "c", "b")
print(LCS(a, b))

# a constructed example
x <- seq(0, 2 * pi, 0.1)  # time
y <- 5 + sin(x)           # a process
o <- y + rnorm(x, sd=0.2) # observation with random error
p <- y + 0.1              # simulation with systematic bias
plot(x, o); lines(x, p)
```

```
lcs <- LCS(f.slope(x, o), f.slope(x, p))  # too much noise
lcs$LLCS
lcs$QSI

os <- ksmooth(x, o, kernel = "normal", bandwidth = dpill(x, o), x.points = x)$y
lcs <- LCS(f.slope(x, os), f.slope(x, p))
lcs$LLCS
lcs$QSI


# observed and measured data with non-matching time intervals
data(phyto)
bbobs    <- dpill(obs$t, obs$y)
n        <- tail(obs$t, n = 1) - obs$t[1] + 1
obsdpill <- ksmooth(obs$t, obs$y, kernel = "normal", bandwidth = bbobs,
                    n.points = n)
obss     <- data.frame(t = obsdpill$x, y = obsdpill$y)
obss     <- obss[match(sim$t, obss$t),]
obs_f1   <- f.slope(obss$t, obss$y)
sim_f1   <- f.slope(sim$t, sim$y)
lcs      <- LCS(obs_f1, sim_f1)
lcs$QSI
```

---

phyto *Observed and Predicted Data of Phytoplankton*

---

## Description

The data contain the day since 1.1.1994 and observed/predicted biovolumes of phytoplankton.

## Usage

```
obs
sim
```

## Format

Two data frames of two variables with the following components:

`obs`: A data frame of observed phytoplankton concentration in Bautzen reservoir 1994 (TU Dresden, Institute of Hydrobiology, workgroup limnology) with the elements:

- `t`: time code
- `y`: observed biovolume (mg/L)

`sim`: A data frame of predicted phytoplankton concentration in Bautzen reservoir 1994 (TU Dresden, Institute of Hydrobiology, workgroup Limnology) with the elements:

- `t`: time code
- `y`: predicted biovolume (mg/L)

quantV *Quantitative Validation Methods*

## Description

Different methods for calculating the difference between two vectors.

## Usage

```
generalME(o, p,
          ignore   = c("raw", "centered", "scaled", "ordered"),
          geometry = c("real", "logarithmic", "geometric", "ordinal"),
          measure  = c("mad", "var", "sd"),
          type     = c("dissimilarity", "normalized", "similarity",
                       "reference", "formula", "name", "function"),
                       method = NULL)
   MAE(o, p, type = "dissimilarity")
  MAPE(o, p, type = "dissimilarity")
   MSE(o, p, type = "dissimilarity")

  RMSE(o, p, type = "dissimilarity")
  CMAE(o, p, type = "dissimilarity")
  CMSE(o, p, type = "dissimilarity")
 RCMSE(o, p, type = "dissimilarity")
  SMAE(o, p, type = "dissimilarity")
  SMSE(o, p, type = "dissimilarity")
 RSMSE(o, p, type = "dissimilarity")
  MALE(o, p, type = "dissimilarity")
  MAGE(o, p, type = "dissimilarity")
 RMSLE(o, p, type = "dissimilarity")
 RMSGE(o, p, type = "dissimilarity")

 SMALE(o, p, type = "dissimilarity")
 SMAGE(o, p, type = "dissimilarity")
 SMSLE(o, p, type = "dissimilarity")

RSMSLE(o, p, type = "dissimilarity")
RSMSGE(o, p, type = "dissimilarity")

  MAOE(o, p, type = "dissimilarity")
  MSOE(o, p, type = "dissimilarity")
 RMSOE(o, p, type = "dissimilarity")
```

## Arguments

| | |
|---|---|
| o | vector of observed values |
| p | vector of corresponding predicted values |

| | |
|---|---|
| type | one of "dissimilarity", "normalized", "similarity", "reference", "formula", for the dissimilarity measure, the normalized dissimilarity measure, the similarity measure, or the formula for the normalized measure. For generalME it is additionally possible to specify "function" for getting the corresponding function and "name" for getting the name of the function. |
| ignore | specifies which aspects should be ignored: "raw" compares original values, "centered" removes differences in mean, "scaled" ignores scaling, "ordered" indicates the use of the ordinal geometry only. |
| geometry | indicating the geometry to be used for the data and the output, "real" corresponds to arithmetic differences and means, "logarithmic" to handling relative data on a logarithmic scale, "geometric" to geometric means and differences and "ordinal" to a pure ordinal treatment. |
| measure | indicates how distances should be measured: as mean absolute distances like in MAD, as squared distances like in a variance, or as the root of mean squared distances like in sd. |
| method | optionally the function to be used can specified directly as a function or as a string. |

## Details

These comparison criteria are designed for a semiquantitative comparison of observed values o with predicted values p to validate the performance of the prediction.
The general naming convention follows the grammar scheme
[R][C|S]M[S|A][L|G|O]E
corresponding to [Root] [Centered | Scaled] Mean [Squared | Absolute]
[Logarithmic, Geometric, Ordinal] Error

**Root** is used together with squared errors to indicate, that a root is applied to the mean.

**Centered** indicates that an additive constant is allowed.

**Scaled** indicates that a scaling of the predictive sequence is allowed. Scaled implies centered for real scale.

**Squared** indicates that squared error is used.

**Absolute** indicates that absolute error is used.

**Logarithmic** indicates that the error is calculated based on the logarithms of the values. This is useful for data on a relative scale.

**Geometric** indicates that the result is to be understood as a factor, similar to a geometric mean.

**Ordinal** indicates that only the order of the observations is taken into account by analyzing the data by ranks scaled to the interval [0, 1].

The mean errors for squared error measures are based on the number of degrees of freedom of the residuals.

## Value

| | |
|---|---|
| generalME | selects the best deviance measure according to the description given in the parameters. It has the two additional possibilities of name and function in the type parameter. |

| | |
|---|---|
| MAE | mean absolute error $\frac{1}{n}$ |
| MAPE | mean absolute percentage error |
| MSE | mean squared error |
| RMSE | root mean squared error |
| CMAE | centered mean absolute error |
| CMSE | centered mean squared error |
| RCMSE | root centered mean squared error |
| SMAE | scaled mean absolute error |
| SMSE | scaled mean squared error |
| RSMSE | root scaled mean squared error |
| MALE | mean absolute logarithmic error |
| MAGE | mean absolute geometric error |
| MSLE | mean squared logarithmic error |
| MSGE | mean squared geometric error |
| RMSLE | root mean squared logarithmic error |
| SMALE | scaled mean absolute logarithmic error |
| SMAGE | scaled mean absolute relative error |
| SMSLE | scaled mean squared logarithmic error |
| RSMSLE | root scaled mean squared logarithmic error |
| RSMSGE | root scaled mean squared geometric error |
| MAOE | mean absolute ordinal error |
| MSOE | mean squared ordinal error |
| RMSOE | root mean squared ordinal error |

## References

Mayer, D. G. and Butler, D. G. (1993) Statistical Validation. Ecological Modelling, 68, 21-32.

Jachner, S., K.G. v.d. Boogaart, T. Petzoldt (2007) Statistical methods for the qualitative assessment of dynamic models with time delay (R package qualV), Journal of Statistical Software, 22(8), 1–30. URL http://www.jstatsoft.org/v22/i08/.

## See Also

EF, GRI, compareME

## Examples

```
data(phyto)
obsb <- na.omit(obs[match(sim$t, obs$t), ])
simb <- sim[na.omit(match(obs$t, sim$t)), ]
o <- obsb$y
p <- simb$y
```

```
generalME(o, p, ignore = "raw", geometry = "real")

   MAE(o, p)
  MAPE(o, p)
   MSE(o, p)
  RMSE(o, p)
  CMAE(o, p)
  CMSE(o, p)
 RCMSE(o, p)
  SMAE(o, p)
  SMSE(o, p)
 RSMSE(o, p)
  MALE(o, p)
  MAGE(o, p)
 RMSLE(o, p)
 RMSGE(o, p)

 SMALE(o, p)
 SMAGE(o, p)
 SMSLE(o, p)

RSMSLE(o, p)
RSMSGE(o, p)

  MAOE(o, p)
  MSOE(o, p)
 RMSOE(o, p)
   MAE(o, p)
  MAPE(o, p)


   MSE(o, p, type = "s")
  RMSE(o, p, type = "s")
  CMAE(o, p, type = "s")
  CMSE(o, p, type = "s")
 RCMSE(o, p, type = "s")
  SMAE(o, p, type = "s")
  SMSE(o, p, type = "s")
 RSMSE(o, p, type = "s")
  MALE(o, p, type = "s")
  MAGE(o, p, type = "s")
 RMSLE(o, p, type = "s")
 RMSGE(o, p, type = "s")

 SMALE(o, p, type = "s")
 SMAGE(o, p, type = "s")
 SMSLE(o, p, type = "s")

RSMSLE(o, p, type = "s")
RSMSGE(o, p, type = "s")

  MAOE(o, p, type = "s")
  MSOE(o, p, type = "s")
```

```
RMSOE(o, p, type = "s")
```

---

qvalLCS *Qualitative Validation by Means of Interval Sequences and LCS*

---

## Description

Dividing time series into interval sequences of qualitative features and determining the similarity of the qualitative behavior by means of the length of LCS.

## Usage

```
qvalLCS(o, p,
        o.t     = seq(0, 1, length.out = length(o)),
        p.t     = seq(0, 1, length.out = length(p)),
        smooth  = c("none", "both", "obs", "sim"),
        feature = c("f.slope", "f.curve", "f.steep", "f.level"))
## S3 method for class 'qvalLCS'
print(x, ...)
## S3 method for class 'qvalLCS'
plot(x, y = NULL, ..., xlim = range(c(x$obs$x, x$sim$x)),
ylim = range(c(x$obs$y, x$sim$y)), xlab = "time", ylab = " ",
col.obs = "black", col.pred = "red",
plot.title = paste("LLCS =", x$lcs$LLCS, ", QSI =", x$lcs$QSI),
legend = TRUE)
## S3 method for class 'qvalLCS'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| o | vector of observed values |
| p | vector of predicted values |
| o.t | vector of observation times |
| p.t | vector of times for predicted values |
| smooth | character string to decide if values should be smoothed before validation, default no smoothing "none" is set, "both" observed and predicted values will be smoothed, "obs" only observed, and "sim" only simulated values will be smoothed. |
| feature | one of "f.slope", "f.curve", "f.steep", "f.level" as defined in features to divide the time series into interval sequences of these feature. As default the first derivative "f.slope" is used. |
| x | a result from a call of qvalLCS |
| y | y unused |
| ... | further parameters to be past to plot |

| | |
|---|---|
| xlim | the size of the plot in x-direction |
| ylim | the size of the plot in y-direction |
| xlab | the label of the x-axis of the plot |
| ylab | the label of the y-axis of the plot |
| col.obs | color to plot the observations |
| col.pred | color to plot the predictions |
| plot.title | title for the plot |
| legend | tegend for the plot |
| object | a result from a call of qvalLCS |

## Details

Common quantitative deviance measures underestimate the similarity of patterns if there are shifts in time between measurement and simulation. These methods also assume compareable values in each time series of the whole time sequence. To compare values independent of time the qualitative behavior of the time series could be analyzed. Here the time series are divided into interval sequences of their local shape. The comparison occurs on the basis of these segments and not with the original time series. Here shifts in time are possible, i.e. missing or additional segments are acceptable without losing similarity. The dynamic programming algorithm of the longest common subsequence LCS is used to determine QSI as index of similarity of the patterns.
If selected the data are smoothed using a weighted average and a Gaussian curve as kernel. The bandwidth is automatically selected based on the plug-in methodology (dpill, see package **KernSmooth** for more details).

**print.qvalLCS** prints only the requested value, without additional information.

**summary.qvalLCS** prints all the additional information.

**plot.qvalLCS** shows a picture visualizing a LCS.

## Value

The result is an object of type qvalLCS with the following entries:

| | |
|---|---|
| smooth | smoothing parameter |
| feature | feature parameter |
| o | xy-table of observed values |
| p | xy-table of predicted values |
| obs | xy-table of (smoothed) observed values |
| sim | xy-table of (smoothed) simulated values |
| obsf | interval sequence of observation according to selected features |
| simf | interval sequence of simulation according to selected features |
| lcs | output of LCS function |
| obs.lcs | one LCS of observation |
| sim.lcs | one LCS of simulation |

## References

Agrawal R., K. Lin., H. Sawhney and K. Shim (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In VLDB '95: Proceedings of the 21. International Conference on Very Large Data Bases, pp. 490-501. Morgan Kaufmann Publishers Inc. ISBN 1-55860-379-4.

Cuberos F., J. Ortega, R. Gasca, M. Toro and J. Torres (2002). Qualitative comparison of temporal series - QSI. Topics in Artificial Intelligence. Lecture Notes in Artificial Intelligence, 2504, 75-87.

Jachner, S., K.G. v.d. Boogaart, T. Petzoldt (2007) Statistical methods for the qualitative assessment of dynamic models with time delay (R package qualV), Journal of Statistical Software, 22(8), 1–30. URL http://www.jstatsoft.org/v22/i08/.

## See Also

LCS, features

## Examples

```
# a constructed example
x <- seq(0, 2*pi, 0.1)
y <- 5 + sin(x)           # a process
o <- y + rnorm(x, sd=0.2) # observation with random error
p <- y + 0.1              # simulation with systematic bias

qvalLCS(o, p)
qvalLCS(o, p, smooth="both", feature="f.curve")

qv <- qvalLCS(o, p, smooth = "obs")
print(qv)
plot(qv, ylim=c(3, 8))

# observed and measured data with non-matching time steps
data(phyto)
qvlcs <- qvalLCS(obs$y, sim$y, obs$t, sim$t, smooth = "obs")

basedate <- as.Date("1960/1/1")
qvlcs$o$x   <- qvlcs$o$x + basedate
qvlcs$obs$x <- qvlcs$obs$x + basedate
qvlcs$sim$x <- qvlcs$sim$x + basedate
qvlcs$obs.lcs$x <- qvlcs$obs.lcs$x + basedate
qvlcs$sim.lcs$x <- qvlcs$sim.lcs$x + basedate

plot.qvalLCS(qvlcs)
summary(qvlcs)
```

---

timetrans *Bijective Transformations of Time*

---

## Description

Various function models for isoton bijective transformation of a time interval to itself.

## Usage

```
transBeta(x, p, interval = c(0, 1), inv = FALSE,
  pmin = -3, pmax = 3, p0 = c(0, 0))
transSimplex(x, p, interval = c(0, 1), inv = FALSE,
  pmin = -2, pmax = 2, p0 = c(0, 0, 0, 0, 0))
transBezier(x, p, interval = c(0, 1), inv = FALSE,
  pmin = 0, pmax = 1, p0 = c(0.25, 0.25, 0.75, 0.75))
```

## Arguments

| | |
|---|---|
| `x` | a vector of values to be transformed, |
| `p` | the vector of parameters for the transformation, |
| `interval` | a vector of length 2 giving the minimum and maximum value in the transformation interval. |
| `inv` | a boolean, if true the inverse transform is computed. |
| `pmin` | a number or a vector giving the minimal useful value for the parameters. This information is not used by the function itself, but rather provides a meta information about the function used in `timeTransME`. The chosen values are quite restrictive to avoid stupid extreme transformation. |
| `pmax` | provides similar to `pmin` the upper useful bounds for the parameters. |
| `p0` | provides similar to `pmin` and `pmax` the parameterization for the identify transform. |

## Details

`transBeta` The transformation provided is the distribution function of the Beta-Distribution with parameters `exp(p[1])` and `exp(p[2])` scaled to the given interval. This function is guaranteed to be strictly isotonic for every choice of p. p has length 2. The strength of the Beta transformation is the reasonable behavior for strong time deformations.

`transSimplex` The transformation provided a simple linear interpolation. The interval is separated into equidistant time spans, which are transformed to non-equidistant length. The length of the new time spans is the proportional to exp(c(p, 0)). This function is guaranteed to be strictly isotonic for every choice of p. p can have any length. The strength of the Simplex transformation is the possibility to have totally different speeds at different times.

`transBezier` The transformation is provided by a Bezier-Curve of order `length(p) / 2 + 1`. The first and last control point are given by `c(0, 0)` and `c(1, 1)` and the intermediate control points are given by `p[c(1, 2) + 2 * i - 2]`. This function is not guaranteed to be isotonic for `length(p) > 4`. However it seams useful. A major theoretical advantage is that this model is symmetric between image and coimage. The strength of the Bezier transformation is fine tuning of transformation.

## Value

The value is a vector of the same length as x providing the transformed values.

## See Also

timeTransME

## Examples

```
t <- seq(0, 1, length.out = 101)
par(mfrow = c(3, 3))
plot(t, transBeta(t, c(0, 0)), type = "l")
plot(t, transBeta(t, c(0, 1)), type = "l")
plot(t, transBeta(t, c(-1,1)), type = "l")
plot(t, transSimplex(t, c(0)), type = "l")
plot(t, transSimplex(t, c(3, 2, 1)), type = "l")
plot(t, transSimplex(t, c(0, 2)), type = "l")
plot(t, transBezier(t, c(0, 1)), type = "l")
plot(t, transBezier(t, c(0, 1, 1, 0)), type = "l")
plot(t, transBezier(t, c(0.4, 0.6)), type = "l")
```

---

# timeTransME *Transformation of Time to Match Two Time Series*

---

## Description

Transforming the time of predicted values by means of a monotonic mapping.

## Usage

```
timeTransME(o, p,
            o.t      = seq(0, 1, length.out = length(o)),
            p.t      = seq(0, 1, length.out = length(p)),
            ignore   = "scaled",
            geometry = "real",
            measure  = "mad",
            type     = c("dissimilarity", "normalized",
                         "similarity", "reference"),
            interval = range(c(o.t, p.t)),
            time     = c("transformed", "fixed"),
            trans    = transBeta,
            p0       = eval(formals(trans)$p0),
            pmin     = eval(formals(trans)$pmin, list(p = p0)),
            pmax     = eval(formals(trans)$pmax, list(p = p0)),
            timeMEFactor = 0,
            timeME       = MAE,
            timeMEtype   = "normalized",
            timeScale    = 1,
```

```
            ME     = generalME(o, p, ignore, geometry, measure,
                               type = "function"),
            MEtype = c("dissimilarity", "normalized"),
            trials = 100,
            debug  = FALSE)
## S3 method for class 'timeTransME'
print(x, ..., digits = 3)
## S3 method for class 'timeTransME'
summary(object, ...)
## S3 method for class 'timeTransME'
plot(x, y = NULL, ..., col.obs = "black", col.pred = "green",
     col.map = "red", sub = x$call, xlab = "t",
     xlim = range(x$x), ylim = range(c(0, x$yo, x$yp)))
```

## Arguments

| | |
|---|---|
| x | a result from a call to timeTransME |
| object | a result from a call to timeTransME |
| o | vector of observed values |
| p | vector of predicted values |
| o.t | vector of observation times |
| p.t | vector of times for predicted values |
| ignore | one of "raw", "centered", "scaled" or "ordered" as defined in generalME to specify the aspects of the data to be ignored. |
| geometry | one of "real", "logarithmic", "geometric", "ordinal" as defined in generalME to specify the geometry of the observed data. |
| measure | one of "mad", "sd", "var" to specify the type of error to be measured. |
| type | one of "dissimilarity", "normalized", "similarity" or "reference" as defined in generalME to specify the type of deviance measure to be used. |
| interval | a vector with two entries giving start and end time of the experiment. |
| time | indicates wether the time should actually be transformed. LCS is currently not implemented. Use the LCS method directly. |
| trans | the model function for the time transformation. See transBezier for possible alternatives. |
| p0 | the identity parameters for the time-transformation. A non identity value can be given to force specific parameters for the transformation with time = "fixed". |
| pmin | number or vector providing the minimal allowed values for the parameters of the transformation. |
| pmax | number or vector providing the minimal allowed values for the parameters of the transformation. |
| timeME | The timeTransME minimizes a weighted sum of the deformation of the time scale and of the data values according to totalME = minimum of |

```
ME(o(x), p(trans(x, timep)), MEtype) +
  timeMEFactor * timeME(x * timeScale,
  trans(x, timep) * timeScale, timeMEtype)
```

over p for `x = c(ot, trans(pt, timep, inv = TRUE))`.
`timeME` specifies the function to be used to quantify the temporal deformation.

| | |
|---|---|
| `timeMEtype` | the type of deviance measure ("dissimilarity" or "normalized") to be used for `timeME`. |
| `timeMEFactor` | a real value specifying the weighting of the time deformation against the value deformation. A value of 0 avoids penalty for time deformation. |
| `timeScale` | a scaling applied to the time values before `timeME` is applied. This can be used to change the units of measurement for the time. |
| `ME` | the deviance function to be used for the data. See `MSE` for alternatives. |
| `MEtype` | the type of Mean Error to be used in the calculations. This is not the type of Measure to be reported. |
| `trials` | The number of random starting values that should be used during the optimization of the time transformation. The optimization of the time transformation is a very critical task of this procedure and it had been shown by practical tests that a single local optimization typically fails to find the globally best fit. Depending on the number of parameters a value between 100 and 10000 seems reasonable for this parameter. |
| `debug` | a logical. If true some diagnostic information for the optimization step is printed. |
| `...` | further parameters to be passed to `plot` |
| `col.obs` | color to plot the observations |
| `col.pred` | color to plot the predictions |
| `col.map` | color to plot the mapped predictions |
| `sub` | the sub-headline of the plot |
| `xlab` | the label of the x-axis of the plot |
| `xlim` | the size of the plot in x-direction |
| `ylim` | the size of the plot in y-direction |
| `y` | y unused |
| `digits` | number of significant digits displayed |

## Details

Common quantitative deviance measures underestimate the similarity of patterns if there are shifts in time between measurement and simulation. An alternative to measure model performance independent of shifts in time is to transform the time of the simulation, i.e. to run the time faster or slower, and to compare the performance before and after the transformation. The applied transformation function must be monotonic. `timeTransME` minimizes the joint criterium `ME(o(x), p(trans(x, timep)), MEtype) timeMEFactor * timeME(x * timeScale, trans(x, timep) * timeScale, timeMEtype)` to find a best fitting time transformation.

`print.timeTransME` prints only the requested value, without additional information.

`summary.timeTransME` prints all the additional information.

`plot.timeTransME` shows a picture visualising the fit of the transformed dataset. This can be used as a diagnostic.

## Value

The result is an object of type `timeTransME` with the following entries:

| | |
|---|---|
| `totalME` | the requested measure with specified type, |
| `criterium` | the "dissimilarity" measure, which was calculated as a minimum of<br>`ME(o(x), p(trans(x, timep)), MEtype) + timeMEFactor * timeME(x * timeScale, trans(x, timep) * timeScale, timeMEtype)`<br>. |
| `reference` | the reference value of this criterium achieved without time deformation and full dissimilarity. |
| `call` | the call used to generate this deviance. |
| `x` | the times at which the series were compared from the perspective of the observations. |
| `xp` | the transformed times at which the series were compared from the perspective of the prediction. |
| `yo` | the interpolated values of the observations at times `x`. |
| `yp` | the interpolated values of the time transformed predictions at times `x`. |
| `timeME` | the deviance of the time transformation:<br>`timeME(x, trans(x, ME), timeMEtype))`. |
| `timeMEref` | the reference value of timeME |
| `timeMEFactor` | the factor to be used for timeME in the weighting with respect to `ME`. |
| `timeScale` | the scaling to time to account for an other unit. |
| `p` | the parameter of trans minimizing the criterium. |
| `interval` | the interval of time under consideration |
| `trans` | the transformation function used for the time. |
| `optim` | contains informations about the convergence of the optimization procedure and a list of secondary minima found. This additional list element occurs only if there is actually a minimisation performed. |

## Note

The deviance calculated by `timeTransME(..., time = "fixed")` and the corresponding deviance measure are different because the timeTransME does an interpolation and compares time sequences at different spacing, while a simple deviance measure compares values only.
The CPU usage of the calculation of the minimum, when `trans = "transform"` is very high, because the optimization is done a hundred times with random starting values for the parameters. This is necessary since with the given objective the general purpose optimizers often run into local minima and/or do not converge. The number of iterations can be controlled with the parameter `trials`. Setting `debug = TRUE` gives an impression how long it takes to find an improved optimum.

## See Also

transBeta, transBezier

## Examples

```
set.seed(123)
## a constructed example
x <- seq(0, 2*pi, length=10)
o <- 5 + sin(x) + rnorm(x, sd=0.2) # observation with random error
p <- 5 + sin(x-1)                  # simulation with time shift

# timeTransME(o, p) # reasonably accurate but takes very long!
# timeTransME(o, p, trials=5, debug=TRUE)

ttbeta <- timeTransME(o, p, trials=5)
plot(ttbeta)
## Not run:
ttsimplex <- timeTransME(o, p, trans = transSimplex, trials=5)
plot(ttsimplex)

ttbezier <- timeTransME(o, p, trans = transBezier, trials=5)
plot(ttbezier)

## End(Not run)

## observed and measured data with non-matching time intervals
data(phyto)
bbobs    <- dpill(obs$t, obs$y)
n        <- diff(range(obs$t)) + 1
obss     <- ksmooth(obs$t, obs$y, kernel = "normal", bandwidth = bbobs,
            n.points = n)
names(obss) <- c("t", "y")
obss     <- as.data.frame(obss)[match(sim$t, obss$t), ]

tt       <- timeTransME(obss$y, sim$y, obss$t, sim$t, ME = SMSE,
            timeMEFactor = 0, time = "transform", type = "n", trials = 5)
round(tt$totalME, digits = 3)

basedate <- as.Date("1960/1/1")
plot(basedate + sim$t, sim$y, type="l", ylim = c(min(obs$y, sim$y),
  max(obs$y, sim$y)), xlab = "time", ylab = "Phytoplankton (mg/L)",
  col = 2, font = 2, lwd = 2, cex.lab = 1.2, las = 1)
lines(basedate + obss$t, obss$y, lwd = 2)
points(basedate + obs$t, obs$y, lwd = 2)
lines(basedate + tt$x, tt$yp, lwd = 2, col = 2, lty = 2)
legend(basedate + 12600, 50, c("measurement", "smoothed measurement",
"simulation", "transformed simulation"), lty = c(0, 1, 1, 2),
pch = c(1, NA, NA, NA), lwd = 2, col = c(1, 1, 2, 2))

tt1 <- timeTransME(obs$y, sim$y, obs$t, sim$t, ME = SMSLE, type = "n",
  time = "fixed")
tt1
```

```
plot(tt1)
summary(tt1)

## Not run:
tt2 <- timeTransME(obss$y, sim$y, obss$t, sim$t, ME = SMSLE, type = "n",
  time = "trans", debug = TRUE)
tt2
plot(tt2)  # logarithm (SMSLE) is not appropriate for the example
summary(tt2)
tt3 <- timeTransME(obss$y, sim$y, obss$t, sim$t, ME = SMSE, type = "n",
  time = "trans", trans = transBezier, debug = TRUE)
tt3
plot(tt3)
summary(tt3)
tt4 <- timeTransME(obss$y, sim$y, obss$t, sim$t, ME = MSOE, type = "n",
  time = "trans", trans = transBezier, debug = TRUE)
tt4
plot(tt4)
summary(tt4)

## End(Not run)
```

# Index