Package ‘tsutils’

Type Package
Title Time Series Exploration, Modelling and Forecasting
Version 0.9.4
Description Includes: (i) tests and visualisations that can help the modeller explore time series components and perform decomposition; (ii) modelling shortcuts, such as functions to construct lagmatrices and seasonal dummy variables of various forms; (iii) an implementation of the Theta method; (iv) tools to facilitate the design of the forecasting process, such as ABC-XYZ analyses; and (v) “quality of life” functions, such as treating time series for trailing and leading values.

Imports RColorBrewer, forecast, MAPA, plotrix
Suggests thief
License GPL-3
Encoding UTF-8
LazyData true
URL https://github.com/trnnick/tsutils/
RoxygenNote 7.2.3
NeedsCompilation no
Author Nikolaos Kourentzes [aut, cre],
Ivan Svetunkov [ctb],
Oliver Schaer [ctb]
Maintainer Nikolaos Kourentzes <nikolaos@kourentzes.com>
Repository CRAN
Date/Publication 2023-11-15 00:20:02 UTC

R topics documented:

  abc ................................................................. 2
  abcyxz ............................................................ 3
  cnav .............................................................. 4
  coxstuart ....................................................... 5
abc

Description
Perform ABC analysis on a set of time series.

Usage
abc(x, prc = c(0.2, 0.3, 0.5))

## S3 method for class 'abc'
plot(x, cex.prc = 0.8, ...)

Arguments

  x
  this can either be an array, where each column is a series, or a vector of values.
  If x is an array of time series, then the importance of each series is calculated as
  the mean of the observations (sales volume). Otherwise, value can be whatever
  quantity is provided.

  prc
  a vector of percentages indicating how many items are included in each class.
  By default this is c(0.2, 0.3, 0.5), but any set of percentage values can be used
  as long as 0<=prc[1]<=1 and sum(prc)==1.

  cex.prc
  font size of percentages reported in plot.

  ... additional arguments passed to the plot.
abcxyz

Value

Return object of class abc and contains:

- value: a vector containing the importance value of each series.
- class: a vector containing the class membership of each series.
- rank: a vector containing the rank of each series, with 1 being the highest ranking series.
- conc: the importance concentration of each class, as percentage of total value.

Methods (by generic)

- plot(abc): plot ABC or XYZ analyses.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References


See Also

xyz, abcxyz.

Examples

x <- abs(matrix(cumsum(rnorm(5400,0,1)),36,150))
z <- abc(x)
print(z)
plot(z)

---

abcxyz ABC-XYZ visualisation

Description

Jointly visualise ABC and XYZ analyses.

Usage

abcxyz(imp, frc, outplot = c(TRUE, FALSE), error = NULL, ...)

Arguments

- `imp`: an object of class `abc` that is the output of function `abc`.
- `frc`: an object of class `abc` that is the output of function `xyz`.
- `outplot`: if `TRUE`, then provide a visualisation of the analyses.
- `error`: vector of forecast errors for each series that will be distributed in each class, presented as an average.
- ...: additional arguments passed to the plot.

Value

A list containing:

- `class`: a matrix containing the number of time series in each class.
- `error`: a matrix containing the averaged error for each class, if the argument `error` was used.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References


See Also

`abc`, `xyz`.

Examples

```r
x <- abs(matrix(cumsum(rnorm(5400,0,1)),36,150))
abcxyz(abc(x),xyz(x,type="cv"))
```

---

**cmav**

Centred moving average

Description

Calculate the Centred Moving Average (CMA) for time series.
Usage

```r
cmav(
  y,
  ma = NULL,
  fill = c(TRUE, FALSE),
  outplot = c(FALSE, TRUE),
  fast = c(TRUE, FALSE)
)
```

Arguments

- `y`: input time series. Can be `ts` or `msts` object.
- `ma`: length of centred moving average. If `y` is a `ts` object then the default is its frequency. If it is a `msts` object the default is the maximum frequency.
- `fill`: if TRUE, then fill first and last `ma/2` observations using exponential smoothing.
- `outplot`: if TRUE, then output a plot of the time series and the moving average.
- `fast`: if TRUE, then only a limited set of models are evaluated for CMA extrapolation.

Value

Centred moving average. If `y` is a `ts` object, then `cmav` has the same properties.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References


Examples

```r
cmav(referrals,outplot=TRUE)
```

---

**coxstuart**

Cox-Stuart test

Description

Perform Cox-Stuart test for location or dispersion.

Usage

```r
coxstuart(y, type = c("trend", "deviation", "dispersion"), alpha = 0.05)
```
Arguments

y     input data.
type  type of test. Can be:
      • "trend": test for changes in trend.
      • "deviation": test for changes in deviation.
      • "dispersion": test for changes in dispersion (range).

alpha significance level.

Value

A list containing:

• H: hypothesis outcome.
• p.value: corresponding p-value.
• Htxt: textual description of the hypothesis outcome.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

Examples

coxstuart(referrals)

decomp

Classical time series decomposition

Description

Perform classical time series decomposition.

Usage

decomp(
  y,
  m = NULL,
  s = NULL,
  trend = NULL,
  outplot = c(FALSE, TRUE),
  decomposition = c("multiplicative", "additive", "auto"),
  h = 0,
  type = c("mean", "median", "pure.seasonal"),
  w = NULL
)
Arguments

**y**: input time series. Can be `ts` object.

**m**: seasonal period. If `y` is a `ts` object then the default is its frequency.

**s**: starting period in the season. If `y` is a `ts` object then this is picked up from `y`.

**trend**: vector of the level/trend of `y`. Use `NULL` to estimate internally.

**outplot**: if `TRUE`, then provide a plot of the decomposed components.

**decomposition**: type of decomposition. This can be "multiplicative", "additive" or "auto". If `y` contains non-positive values then this is forced to "additive".

**h**: forecast horizon for seasonal component.

**type**: calculation for seasonal component:
- "mean": the mean of each seasonal period.
- "median": the median of each seasonal period.
- "pure.seasonal": estimate using a pure seasonal model.

**w**: percentage or number of observations to winsorise in the calculation of mean seasonal indices. If `w>1` then it is the number of observations, otherwise it is a percentage. If `type != "mean"` then this is ignored.

Value

A list containing:
- **trend**: trend component.
- **season**: season component.
- **irregular**: irregular component.
- **f.season**: forecasted seasonal component if `h>0`.
- **g**: pure seasonal model parameters.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References


Examples

decomp(referrals)
geomean

Geometric mean

Description
Calculate the geometric mean.

Usage
geomean(x, na.rm = c(FALSE, TRUE), ...)

Arguments
- **x**: input data (will be considered as a vector).
- **na.rm**: a logical value indicating whether NA values should be stripped before the computation proceeds.
- **...**: further arguments passed to or from other methods.

Value
The geometric mean of the values in `x`.

Author(s)
Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

Examples
geomean(c(0.5, 1, 1.5))

getOptK

Optimal temporal aggregation level for AR(1), MA(1), ARMA(1,1)

Description
Calculate the theoretically optimal temporal aggregation level for AR(1), MA(1) and ARMA(1,1) time series.

Usage
getOptK(y, m = 12, type = c("ar", "ma", "arma"))
**Arguments**

- **y**: a time series that must be of either `ts` or `msts` class.
- **m**: maximum aggregation level.
- **type**: type of data generating process. Can be:
  - "ar": For AR(1) series.
  - "ma": For MA(1) series.
  - "arma": For ARMA(1,1) series.

**Value**

Identified optimal temporal aggregation level.

**Author(s)**

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

**References**


**Examples**

```r
getOptK(referrals)
```

---

**Description**

Create lead/lags of a variable.

**Usage**

```r
lagmatrix(x, lag)
```

**Arguments**

- **x**: input variable.
- **lag**: vector of leads and lags. Positive numbers are lags, negative are leads. O is the original x.
Value
An array with the resulting leads and lags (columns).

Author(s)
Nikolaos Kourentzes, nikolaos@kourentzes.com

Examples
```r
x <- rnorm(10)
lagmatrix(x, c(0, 1, -1))
```

Description
Calculates the lambdaMax value, which is the penalty term (lambda) beyond which coefficients are guaranteed to be all zero and provides a sequence of nLambda values to lambdaMin in logarithmic descent.

Usage
```r
lambdaseq(
  x, y,
  weight = NA,
  alpha = 1,
  standardise = TRUE,
  lambdaRatio = 1e-04,
  nLambda = 100,
  addZeroLambda = FALSE
)
```

Arguments
- `x`: matrix of regressors. See glmnet.
- `y`: response variable. See glmnet.
- `weight`: vector of length(nrow(y)) for weighted LASSO estimation. See glmnet.
- `alpha`: elastic net mixing value. See glmnet.
- `standardise`: if TRUE, then variables are standardised.
- `lambdaRatio`: ratio between lambdaMax and lambdaMin. That is, lambdaMin <- lambdaMax * lambdaRatio.
- `nLambda`: length of the lambda sequence.
- `addZeroLambda`: if TRUE, then set the last value in the lambda sequence to 0, which is the OLS solution.
Value

A list that contains:

- \( \lambda \) sequence of lambda values, from \( \lambda_{\text{Max}} \) to \( \lambda_{\text{Min}} \).
- \( \lambda_{\text{Min}} \): minimal lambda value.
- \( \lambda_{\text{Max}} \): maximal lambda value.
- nullMSE: MSE of the fit using just a constant term.

Author(s)

Oliver Schaer, <info@oliverschaer.ch>,
Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References


Examples

```r
y <- mtcars[,1]
x <- as.matrix(mtcars[,2:11])
lambda <- lambdaseq(x, y)$lambda

## Not run:
library(glmnet)
fit.lasso <- cv.glmnet(x, y, lambda = lambda)
coef.lasso <- coef(fit.lasso, s = "lambda.1se")

## End(Not run)
```

leadtrail

Remove leading/training zeros/NAs

Description

Remove leading or trailing zeros or NAs from a vector.

Usage

```r
leadtrail(
x,
    rm = c("zeros", "na"),
    lead = c(TRUE, FALSE),
    trail = c(TRUE, FALSE)
)
```
### Arguments

- **x** vector of values to check.
- **rm** what to remove, can be "zeros" or "na".
- **lead** If TRUE, then leading values are removed.
- **trail** If TRUE, then trailing values are removed.

### Value

Resulting vector.

### Author(s)

Nikolaos Kourentzes, nikolaos@kourentzes.com.

### Examples

```r
x <- c(rep(0,5),rnorm(100),rep(0,5))
leadtrail(x)
```

### Description

Perform nonparametric multiple comparisons, across columns, using the Friedman and the post-hoc Nemenyi tests.

### Usage

```r
nemenyi(
  data,
  conf.level = 0.95,
  sort = c(TRUE, FALSE),
  plottype = c("vline", "none", "mcb", "vmcb", "line", "matrix"),
  select = NULL,
  labels = NULL,
  ...
)
```

### Arguments

- **data** an array that includes values to be compared for several treatments (in columns) for several observations (rows), of size n x k. For example, if these are forecast errors, different methods should be in columns and errors for different time series or forecast origins in rows.
### nemenyi

- **conf.level**: the confidence level used for the comparison. Default is 0.95.
- **sort**: if TRUE, then function sorts the outputted values of mean ranks. If plots are request, this is forced to TRUE.
- **plottype**: type of plot to produce:
  - "none": no plot.
  - "mcb": *Multiple Comparison with the Best* style plot.
  - "vmcb": vertical *MCB* plot.
  - "line": summarised line plot.
  - "vline": vertical line plot.
  - "matrix": complete matrix visualisation.
- **select**: highlight selected treatment (column). Number 1 to k. Use NULL for no highlighting.
- **labels**: optional labels for models. If NULL column names of data will be used.
- **...**: additional arguments passed to the plot function.

### Value

Return object of class nemenyi and contains:

- **means**: mean rank of each treatment.
- **intervals**: intervals within there is no evidence of significance difference according to the Nemenyi test at requested confidence level.
- **fpavl**: Friedman test p-value.
- **fH**: Friedman test hypothesis outcome.
- **cd**: Nemenyi critical distance. Output intervals is calculate as means +/- cd.
- **conf.level**: confidence level used for testing.
- **k**: number of treatments (columns).
- **n**: number of observations (rows).

### Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>,
Ivan Svetunkov, <ivan@svetunkov.ru>.

### References


Examples

```r
x <- matrix( rnorm(50*4,mean=0,sd=1), 50, 4)
x[,2] <- x[,2]+1
x[,3] <- x[,3]+0.7
x[,4] <- x[,4]+0.5
colnames(x) <- c("Method A","Method B","Method C - long name","Method D")
nemenyi(x,conf.level=0.95,plottype="vline")
```

---

**Description**

Plots the temporal hierarchy for a given time series of seasonal periodicity.

**Usage**

```r
plotSthief(y, labels = c(TRUE, FALSE), ...)
```

**Arguments**

- `y`: input time series (a `ts` object) or an integer.
- `labels`: if TRUE labels will be added for the temporal aggregation levels if the seasonal period is 4 (quarters), 7 (days in a week), 12 (months), 24 (hours), 48 (half-hours), 52 (weeks) or 364 (days).
- `...`: additional arguments passed to the plotting function.

**Value**

Produces a plot of the temporal hierarchy.

**Author(s)**

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

**References**


**Examples**

```r
plotSthief(AirPassengers)
```
referrals

**NHS A&E Referrals**

**Description**
Monthly Accident & Emergency referrals for England and Wales.

**References**

---

residout

**Residuals control chart**

**Description**
Create a control chart of residuals and identify outliers.

**Usage**

```r
residout(resid, t = 2, outplot = c(TRUE, FALSE))
```

**Arguments**

- `resid`: vector of residuals.
- `t`: threshold value over which standardised residuals are regarded as outliers.
- `outplot`: if TRUE, then a control chart of the standardised residuals is plotted.

**Value**
A list containing:

- `location`: locations of outliers.
- `outliers`: values of outliers.
- `residuals`: standardised residuals.

**Author(s)**
Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

**Examples**

```r
residout(rnorm(50), outplot=TRUE)
```
seasdummy

Create seasonal dummy variables.

Description

Create binary or trigonometric seasonal dummies.

Usage

seasdummy(n, m = NULL, y = NULL, type = c("bin", "trg"), full = c(FALSE, TRUE))

Arguments

n
number of observations to create.

m
seasonal periodicity. If NULL it will take the information from the provided time series (y argument). See notes.

y
this is an optional time series input that can be used to get seasonal periodicity (m) and the start point.

type
type of seasonal dummies to create.

• "bin": binary dummies.

• "trg": trigonometric dummies. See notes.

full
If full is TRUE, then keeps the m-th dummy that is co-linear to the rest. See notes.

Value

An array with seasonal dummies, where rows correspond observations and columns to dummy variables.

Note

If the seasonal periodicity is fractional then the type will be overriden to trigonometric and only two seasonal dummies with be produced. One cosine and one sine.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>

Examples

seasdummy(24, 12)
seasplot

Seasonal plots with simplistic trend/season tests

Description

Construct seasonal plots of various styles for a given time series. The series can automatically detrended as needed.

Usage

seasplot(
  y,
  m = NULL,
  s = NULL,
  trend = NULL,
  colour = NULL,
  alpha = 0.05,
  outplot = c(1, 0, 2, 3, 4, 5),
  decomposition = c("multiplicative", "additive", "auto"),
  cma = NULL,
  labels = NULL,
  ...
)

Arguments

y input time series. Can be ts object.
m seasonal period. If y is a ts object then the default is its frequency.
s starting period in the season. If y is a ts object then this is picked up from y.
trend if TRUE, then presence of trend is assumed and removed. If FALSE no trend is assumed. Use NULL to identify automatically.
colour single colour override for plots.
alpha significance level for statistical tests.
outplot type of seasonal plot
  • 0: none.
  • 1: seasonal diagram.
  • 2: seasonal boxplots.
  • 3: seasonal subseries.
  • 4: seasonal distribution.
  • 5: seasonal density.
decomposition type of seasonal decomposition. This can be "multiplicative", "additive" or "auto". If y contains non-positive values then this is forced to "additive".
cma input precalculated level/trend for the analysis. This overrides trend=NULL.
labels  

external labels for the seasonal periods. Use NULL for none. If length(labels) < m, then this input is ignored.

additional arguments passed to plotting functions. For example, use main="" to replace the title.

Value

An object of class seasexpl containing:

- `season`: matrix of (detrended) seasonal elements.
- `season.exist`: TRUE/FALSE results of seasonality test.
- `season.pval`: p-value of seasonality test (Friedman test).
- `trend`: CMA estimate (using cmav) or NULL if trend=FALSE.
- `trend.exist`: TRUE/FALSE results of trend test.
- `trend.pval`: p-value of trend test (Cox-Stuart).
- `decomposition`: type of decomposition used.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

Examples

seasplot(referrals, outplot=1)

---

Sthief  

Temporal hierarchy S matrix

Description

Calculate the temporal hierarchy summing matrix S for a given time series of seasonal periodicity.

Usage

Sthief(y)

Arguments

y  

input time series (a ts object) or an integer.

Value

S matrix.
Author(s)
Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References

Examples
Sthief(AirPassengers)

theta

Description
Estimate Theta method.
Forecast with fitted Theta method.
Produce a plot of the fitted Theta method.

Usage
theta(
y, 
m = NULL, 
sign.level = 0.05, 
cost0 = c("MSE", "MdSE", "MAE", "MdAE"), 
cost2 = c("MSE", "MdSE", "MAE", "MdAE"), 
costs = c("MSE", "MdSE", "MAE", "MdAE"), 
multiplicative = c("multiplicative", "additive", "auto"), 
cma = NULL, 
outliers = NULL
)

## S3 method for class 'theta'
forecast(object, h = NULL, ...)

## S3 method for class 'theta'
plot(x, thetalines = c(TRUE, FALSE), ...)

theta.thief(y, h = NULL, ...)
Arguments

\(y\)  
input time series. Can be ts object.

\(m\)  
seasonal period. If \(y\) is a ts object then the default is its frequency.

sign.level  
significance level for trend and seasonality tests.

cost0  
cost function of theta0 line. Can be:
  - "MSE": mean squared error.
  - "MdSE": median squared error.
  - "MAE": mean absolute error.
  - "MdAE": median absolute error.

cost2  
cost function of theta2 line. Same options as cost0.

costs  
cost function of seasonal element. Same options as cost0.

multiplicative  
type of seasonal decomposition. This can be "multiplicative", "additive" or "auto". If \(y\) contains non-positive values then this is forced to "additive".

cma  
input precalculated level/trend for the analysis. Use NULL to estimate internally.

outliers  
provide vector of location of observations that are considered outliers (see residout). These will be considered in the estimation of theta0. For no outliers use NULL.

object  
object of class theta.

h  
forecast horizon. If \(h\) is NULL, then the horizon is set equal to the the seasonal frequency.

...  
additional arguments passed to functions.

x  
object of class theta.

thetalines  
if TRUE, then theta lines are included in the plot.

Details

This implementation of the Theta method tests automatically for seasonality and trend. Seasonal decomposition can be done either additively or multiplicatively and the seasonality is treated as a pure seasonal model. The various Theta components can be optimised using different cost functions. The originally proposed Theta method always assumed multiplicative seasonality and presence of trend, while all theta lines were optimised using MSE and seasonality was estimated using classical decomposition.

Value

An object of class theta, containing:

- "method": "Theta".
- "y": the input time series.
- "m": seasonal periods.
- "exist": Statistical testing results, exist[1] is the result for trend, exist[2] is for season.
- "multiplicative": If TRUE, then seasonality is modelled multiplicatively.
- "theta0": fitted theta0 line values.
"theta2": fitted theta2 line values.
"season": fitted season values.
"x.out": modelled outliers.
"cost": cost functions for theta0, theta2 and season components.
"a": SES parameters of theta2.
"b": regression parameters of theta0.
"p": coefficients of outliers from theta0 and theta2 estimation.
"g": pure seasonal exponential smoothing parameters.
"fitted": fitted values.
"residuals": in-sample residuals.
"MSE": in-sample Mean Squared Error.

Functions

- theta.thief(): Wrapper function to use Theta with thief.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References


Examples

```r
fit <- theta(referrals)
plot(fit)

forecast.theta(fit, h=12) # Or simply use forecast(fit)
```

## Not run:
library(thief)
thief(referrals, forecastfunction=theta.thief)

## End(Not run)
trendtest

Description

Test a time series for trend by either fitting exponential smoothing models and comparing them using the AICc, or by using the non-parametric Cox-Stuart test. The tests can be augmented by using multiple temporal aggregation.

Usage

trendtest(
y,                      
extRACT = c("FALSE", "TRUE"),
  type = c("aicc", "cs"),
  mta = c(FALSE, TRUE)
)

Arguments

y                      a time series that must be of either ts or msts class.
extRACT                   if TRUE then the centred moving average of the time series is calculated and the test is performed on that. Otherwise, the test is performed on the raw data.
type                   type of test. Can be:
  • "aicc": test by comparing the AICc of exponential smoothing models. See details.
  • "cs": test by using the Cox-Stuart test. See details.<
mta                   If TRUE augment testing by using Multiple Temporal Aggregation.

Details

All tests are performed at 5

Value

The function returns TRUE when there is evidence of trend and FALSE otherwise.

Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.

References

Examples

trendtest(referrals,TRUE)

description

The tsutils package provides functions to support various aspects of time series and forecasting modelling. In particular this package includes: (i) tests and visualisations that can help the modeller explore time series components and perform decomposition; (ii) modelling shortcuts, such as functions to construct lagmatrices and seasonal dummy variables of various forms; (iii) an implementation of the Theta method; (iv) tools to facilitate the design of the forecasting process, such as ABC-XYZ analyses; and (v) "quality of life" tools, such as treating time series for trailing and leading values.

Time series exploration

- **cmav**: centred moving average.
- **coxstuart**: Cox-Stuart test for location/dispersion.
- **decomp**: classical time series decomposition.
- **seasplot**: construct seasonal plots.
- **trendtest**: test a time series for trend.

Time series modelling

- **getOptK**: optimal temporal aggregation level for AR(1), MA(1), ARMA(1,1).
- **lagmatrix**: create leads/lags of variable.
- **nemenyi**: nonparametric multiple comparisons.
- **residout**: construct control chart of residuals.
- **seasdummy**: create seasonal dummies.
- **theta**: Theta method.

Hierarchical time series

- **Sthief**: temporal hierarchy S matrix.
- **plotSthief**: plot temporal hierarchy S matrix.

Forecasting process modelling

- **abc**: ABC analysis.
- **xyz**: XYZ analysis.
- **abcxyz**: ABC-XYZ analyses visualisation.
Quality of life

- **geomean**: geometric mean.
- **lambdaseq**: generate sequence of lambda for LASSO regression.
- **leadtrail**: remove leading/training zeros/NAs.
- **wins**: winsorisation, including vectorised versions `colWins` and `rowWins`.

Time series data

- **referrals**: A&E monthly referrals.

---

### wins

<table>
<thead>
<tr>
<th>Description</th>
<th>Winsorise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winsorise either by number or percentage of observations.</td>
<td></td>
</tr>
</tbody>
</table>

#### Usage

\[
\text{wins}(x, p = 0.05)
\]

\[
\text{colWins}(x, p = 0.05)
\]

\[
\text{rowWins}(x, p = 0.05)
\]

#### Arguments

- **x**: input data. NAs will be removed.
- **p**: percentage or number of observations to be winsorised. If value is <1 then it is used as a percentages. Otherwise it is the number of observations to winsorise. If the resulting \( p > \text{floor}((\text{length}(x)-1)/2) \), then it is set equal to \( \text{floor}((\text{length}(x)-1)/2) \).

#### Value

Winsorised vector.

#### Functions

- **colWins()**: Vectorised version of wins by columns.
- **rowWins()**: Vectorised version of wins by rows.

#### Author(s)

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.
xyz

Examples

```r
x <- rnorm(100, mean=0, sd=1)
xW <- wins(x)
```

---

**xyz**

**XYZ analysis**

---

**Description**

Perform XYZ analysis on a set of time series.

**Usage**

```r
xyz(x, m = NULL, prc = c(0.2, 0.3, 0.5), type = c("naive", "ets", "cv"))
```

**Arguments**

- `x`  
  this can either be an array, where each column is a series, or a vector of values. If `x` is a vector of values forecastability is not calculated and the input is used as such.

- `m`  
  seasonal length for time series. Required when type is "naive" or "ets".

- `prc`  
  a vector of percentages indicating how many items are included in each class. By default this is `c(0.2, 0.3, 0.5)`, but any set of percentage values can be used as long as `0<=prc[i]<=1` and `sum(prc)==1`.

- `type`  
  the type of forecastability calculation. This can be:
  - "naive": fit naive and seasonal naive and calculate forecastability using RMSE/mean level.
  - "ets": fit ets and calculate and calculate forecastability using RMSE/mean level.
  - "cv": use coefficient of variation as a proxy of forecastability.

**Value**

Return object of class abc and contains:

- value: a vector containing the forecastability value of each series.
- class: a vector containing the class membership of each series.
- rank: a vector containing the rank of each series, with 1 being the lowest forecastability series.
- conc: the forecastability concentration of each class, as percentage of total value.
- model: fitted model for each series.

**Author(s)**

Nikolaos Kourentzes, <nikolaos@kourentzes.com>.
References


See Also

abc, plot.abc, abcxyz.

Examples

```r
x <- abs(matrix(cumsum(rnorm(5400,0,1)),36,150))
z <- xyz(x,m=12)
print(z)
plot(z)
```
Index

* Regression
  lambdaseq, 10
* cma
  cmav, 4
* colWins
  wins, 24
* data
  referrals, 15
* geomean
  geomean, 8
* htest
  coxstuart, 5
  getOptK, 8
  nemenyi, 12
  trendtest, 22
* rowWins
  wins, 24
* ts
  abc, 2
  abcxyz, 3
  cmav, 4
  decomp, 6
  lambdaseq, 10
  plotSthief, 14
  residout, 15
  seasplot, 17
  Sthief, 18
  theta, 19
  xyz, 25
* wins
  wins, 24
  abc, 2, 4, 23, 26
  abcxyz, 3, 3, 23, 26
  cmav, 4, 18, 23
  colWins (wins), 24
  coxstuart, 5, 23
  decomp, 6, 23
  forecast.theta (theta), 19
  geomean, 8, 24
  getOptK, 8, 23
  lagmatrix, 9, 23
  lambdaseq, 10, 24
  leadtrail, 11, 24
  nemenyi, 12, 23
  plot.abc, 26
  plot.abc (abc), 2
  plot.theta (theta), 19
  plotSthief, 14, 23
  referrals, 15, 24
  residout, 15, 20, 23
  rowWins (wins), 24
  seasdummy, 16, 23
  seasplot, 17, 23
  Sthief, 18, 23
  theta, 19, 23
  thief, 21
  trendtest, 22, 23
  tsutils, 23
  wins, 24, 24
  xyz, 3, 4, 23, 25