Package 'pastecs'

April 19, 2009

Title Package for Analysis of Space-Time Ecological Series

Version 1.3-8

Date 2009-25-23

Author Frederic Ibanez <ibanez@obs-vlfr.fr>, Philippe Grosjean <phgrosjean@sciviews.org> & Michele Etienne <etienne@obs-vlfr.fr>

Description Regulation, decomposition and analysis of space-time series. The pastecs library is a PNEC-Art4 and IFREMER (Benoit Beliaeff <Benoit.Beliaeff@ifremer.fr>) initiative to bring PASSTEC 2000 (http://www.obs-vlfr.fr/~enseigne/anado/passtec/passtec.htm) functionnalities to R.

URL http://www.sciviews.org/pastecs

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Encoding latin1

License GPL (>= 2)

Depends boot, stats

Repository CRAN

Date/Publication 2009-02-23 11:20:40

R topics documented:

`.gleissberg.table` *Table of probabilities according to the Gleissberg distribution*

Description

The table of probabilities to have 0 to n-2 extrema in a series, for n=3 to 50 (for n > 50, the Gleissberg distribution is approximated with a normal distribution

Note

This dataset is used internally by `pgleissberg()`. You do not have to access it directly. See `pgleissberg()` for more information

See Also

`pgleissberg`

`abund` *Sort variables by abundance*

Description

Sort variables (usually species in a species x stations matrix) in function of their abundance, either in number of non-null values, or in number of individuals (in log). The `f` coefficient allows adjusting weight given to each of these two criteria.

Usage

```
abund(x, f=0.2)
## S3 method for class 'abund':
summary(abd)
## S3 method for class 'abund':
plot(abd, n=abd$n, lvert=TRUE, lvars=TRUE, lcol=2, llty=2,
        all=TRUE, dlab=c("cumsum", "% log(ind.)", "% non-zero"),
        dcol=c(1, 2, 4), dlty, dpos=c(1.5, 20), ...)
## S3 method for class 'abund':
lines(abd, n=abd$n, lvert=TRUE, lvars=TRUE, ...)
## S3 method for class 'abund':
identify(abd, label.pts=FALSE, lvert=TRUE, lvars=TRUE, ...)
## S3 method for class 'abund':
extract(abd, n=abd$n, left=TRUE)
```

Arguments

`x`	A data frame containing the variables to sort according to their abundance in columns
`f`	Weight given to the number of individuals criterium (strictly included between 0 and 1; weight for the non-null values is `1-f`. The default value, `f=0.2`, gives enough weight to the number of non-null values to get abundant species according to this criterium first, but allowing to get at the other extreme rare, but locally abundant species
`abd`	An 'abund' object returned by `abund`
`n`	The number of variables selected at left
`lvert`	If `TRUE` then a vertical line separate the n variables at left from the others

lvars	If TRUE then the x-axis labels of the n left variables are printed in a different color to emphasize them
lcol	The color to use to draw the vertical line (lvert=TRUE) and the variables labels (lvars=TRUE) at left af the nth variable. By default, color 2 is used
llty	The style used to draw the vertical line (lvert=TRUE). By default, a dashed line is used
all	If TRUE then all lines are drawn (cumsum, %log(ind.) and %non-null). If FALSE, only the cumsum line is drawn
dlab	The legend labels
dcol	Colors to use for drawing the various curves on the graph
dlty	The line style to use for drawing the various curves on the graph
dpos	The position of the legend box on the graph (coordinates of its top-left corner). A legend box is drawn only if all=TRUE
...	additional graph parameters
label.pts	Do we have to label points on the graph or to chose an extraction level with the identify() method?
left	If TRUE, the n variables at left are extracted. Otherwise, the total-n variables at right are extracted

Details

Successive sorts can be applied. For instance, a first sort with f = 0.2, followed by an extraction of rare species and another sort with f = 1 allows to collect only rare but locally abundant species.

Value

An object of type 'abund' is returned. It has methods print(), summary(), plot(), lines(), identify(), extract().

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

References

Ibanez, F., J.-C. Dauvin & M. Etienne, 1993. *Comparaison des évolutions à long terme (1977-1990) de deux peuplements macrobenthiques de la baie de Morlaix (Manche occidentale): relations avec les facteurs hydroclimatiques.* J. Exp. Mar. Biol. Ecol., 169:181-214.

See Also

escouf

Examples

```
data(bnr)
bnr.abd <- abund(bnr)
summary(bnr.abd)
plot(bnr.abd, dpos=c(105, 100))
bnr.abd$n <- 26
# To identify a point on the graph, use: bnr.abd$n <- identify(bnr.abd)
lines(bnr.abd)
bnr2 <- extract(bnr.abd)
names(bnr2)
```

AutoD2 *AutoD2, CrossD2 or CenterD2 analysis of a multiple time-series*

Description

Compute and plot multiple autocorrelation using Mahalanobis generalized distance D2. AutoD2 uses the same multiple time-series. CrossD2 compares two sets of multiple time-series having same size (same number of descriptors). CenterD2 compares subsamples issued from a single multivariate time-series, aiming to detect discontinuities.

Usage

```
AutoD2(series, lags=c(1, nrow(series)/3), step=1, plotit=TRUE,
        add=FALSE, ...)
CrossD2(series, series2, lags=c(1, nrow(series)/3), step=1,
        plotit=TRUE, add=FALSE, ...)
CenterD2(series, window=nrow(series)/5, plotit=TRUE, add=FALSE,
        type="l", level=0.05, lhorz=TRUE, lcol=2, llty=2, ...)
```

Arguments

`series`	regularized multiple time-series
`series2`	a second set of regularized multiple time-series
`lags`	minimal and maximal lag to use. By default, 1 and a third of the number of observations in the series respectively
`step`	step between successive lags. By default, 1
`window`	the window to use for CenterD2. By default, a fifth of the total number of observations in the series
`plotit`	if `TRUE` then also plot the graph
`add`	if `TRUE` then the graph is added to the current figure
`type`	The type of line to draw in the CenterD2 graph. By default, a line without points
`level`	The significance level to consider in the CenterD2 analysis. By default 5%
`lhorz`	Do we have to plot also the horizontal line representing the significance level on the graph?

lcol	The color of the significance level line. By default, color 2 is used
llty	The style for the significance level line. By default: llty=2, a dashed line is drawn
...	additional graph parameters

Value

An object of class 'D2' which contains:

lag	The vector of lags
D2	The D2 value for this lag
call	The command invoked when this function was called
data	The series used
type	The type of 'D2' analysis: 'AutoD2', 'CrossD2' or 'CenterD2'
window	The size of the window used in the CenterD2 analysis
level	The significance level for CenterD2
chisq	The chi-square value corresponding to the significance level in the CenterD2 analysis
units.text	Time units of the series, nicely formatted for graphs

WARNING

If data are too heterogeneous, results could be biased (a singularity matrix appears in the calculations).

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Cooley, W.W. & P.R. Lohnes, 1962. *Multivariate procedures for the behavioural sciences.* Whiley & sons.

Dagnélie, P., 1975. *Analyse statistique à plusieurs variables.* Presses Agronomiques de Gembloux.

Ibanez, F., 1975. *Contribution à l'analyse mathématique des évènements en écologie planctonique: optimisations méthodologiques; étude expérimentale en continu à petite échelle du plancton côtier.* Thèse d'état, Paris VI.

Ibanez, F., 1976. *Contribution à l'analyse mathématique des évènements en écologie planctonique. Optimisations méthodologiques.* Bull. Inst. Océanogr. Monaco, 72:1-96.

Ibanez, F., 1981. *Immediate detection of heterogeneities in continuous multivariate oceanographic recordings. Application to time series analysis of changes in the bay of Villefranche sur mer.* Limnol. Oceanogr., 26:336-349.

Ibanez, F., 1991. *Treatment of the data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis.* In: B. Keegan (ed), *Space and time series data analysis in coastal benthic ecology*, p 5-43.

See Also

mahalanobis, acf

Examples

```
data(marphy)
marphy.ts <- as.ts(as.matrix(marphy[, 1:3]))
AutoD2(marphy.ts)
marphy.ts2 <- as.ts(as.matrix(marphy[, c(1, 4, 3)]))
CrossD2(marphy.ts, marphy.ts2)
# This is not identical to:
CrossD2(marphy.ts2, marphy.ts)
marphy.d2 <- CenterD2(marphy.ts, window=16)
lines(c(17, 17), c(-1, 15), col=4, lty=2)
lines(c(25, 25), c(-1, 15), col=4, lty=2)
lines(c(30, 30), c(-1, 15), col=4, lty=2)
lines(c(41, 41), c(-1, 15), col=4, lty=2)
lines(c(46, 46), c(-1, 15), col=4, lty=2)
text(c(8.5, 21, 27.5, 35, 43.5, 57), 11, labels=c("Peripheral Zone", "D1",
        "C", "Front", "D2", "Central Zone")) # Labels
time(marphy.ts)[marphy.d2$D2 > marphy.d2$chisq]
```

---

bnr *A data frame of 163 benthic species measured across a transect*

Description

The bnr data frame has 103 rows and 163 columns. Each column is a separate species observed at least once in one of all stations. Several species are very abundant, other are very rare.

Usage

```
data(bnr)
```

Source

Unpublished dataset.

buysbal *Buys-Ballot table*

Description

Calculate a Buys-Ballot table from a time-series

Usage

```
buysbal(x, y=NULL, frequency=12, units="years", datemin=NULL,
        dateformat="m/d/Y", count=FALSE)
```

Arguments

x	Either a vector with time values (in this case, y must be defined), or a regular time-series
y	If x is a vector with time values, y must contain corresponding observations
frequency	The frequency of observations per year to use in the Buys-Ballot table. By default frequency=12 which corresponds to monthly samples, and the resulting table has 12 column, one per month
units	either "years" (by default), and time is not rescaled, or "days", and the time is rescaled to "years" with the function daystoyears()
datemin	A character string representing the first date, using a format corresponding to dateformat. For instance, with datemin="04/23/1998" and dateformat="m/d/Y", the first observation is assumed to be made the 23th April 1998. In R, it can also be a POSIXt date (see ?DataTimeClasses). In this case, dateformat is not required and is ignored. By default, datemin=NULL
dateformat	The format used for the date in datemin. For instance, "d/m/Y" or "m/d/Y" (by default). The distinction between "Y" and "y" is not important in Splus, but it is vital in R to use "y" for two-digit years (ex: 89) and "Y" for four-digits years (ex: 1989), or the date will not be correctly converted! In R, you can also use a POSIXt format specification like "%d-%m%Y" for instance (see ?strptime for a complete description of POSIXt format specification. In both Splus and R, you can also use "mon" for abbreviated months like "mon d Y" for "Apr 20 1965", and "month" for fully-spelled months like "d month Y" for "24 September 2003"
count	If FALSE (by default), the Buys-Ballot table is calculated. If TRUE, the function returns only the number of observations that are used in each cell of the Buys-Ballot table

Details

A Buys-Ballot table summarizes data to highlight seasonal variations. Each line is one year and each column is a period of the year (12 months, 4 quarters, 52 weeks,...). A cell ij of this table contain the mean value for all observations made during the year i at the period j.

Value

A matrix containing either the Buys-Ballot table (`count=FALSE`), or the number of observations used to build the table (`count=TRUE`)

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

`daystoyears`, `tsd`

Examples

```
data(releve)
buysbal(releve$Day, releve$Melosul, frequency=4, units="days",
        datemin="21/03/1989", dateformat="d/m/Y")
buysbal(releve$Day, releve$Melosul, frequency=4, units="days",
        datemin="21/03/1989", dateformat="d/m/Y", count=TRUE)
```

---

daystoyears *Convert time units from "days" to "years" or back*

Description

Convert time scales. The time scale "days" corresponds to 1 unit per day. The time scale "years" uses 1 unit for 1 year. It is used in any analysis that requires seasonal decomposition and/or elimination.

Usage

```
daystoyears(x, datemin=NULL, dateformat="m/d/Y")
yearstodays(x, xmin=NULL)
```

Arguments

`x`	A vector of time values
`datemin`	A character string representing the first date, using a format corresponding to `dateformat`. For instance, with `datemin="04/23/1998"` and `dateformat="m/d/Y"`, the first observation is assumed to be made the 23th April 1998. In R, it can also be a POSIXt date (see `?DataTimeClasses`). In this case, `dateformat` is not required and is ignored. By default, `datemin=NULL`
`dateformat`	The format used for the date in `datemin`. For instance, `"d/m/Y"` or `"m/d/y"`. The distinction between "Y" and "y" is not important in Splus, but it is vital in R to use "y" for two-digit years (ex: 89) and "Y" for four-digits years (ex: 1989),

or the date will not be correctly converted! In R, you can also use a POSIXt format specification like "%d-%m%Y" for instance (see `?strptime` for a complete description of POSIXt format specification. In both Splus and R, you can also use "mon" for abbreviated months like "mon d Y" for "Apr 20 1965", and "month" for fully-spelled months like "d month Y" for "24 September 2003"

`xmin` The minimum value for the "days" time-scale

Details

The "years" time-scale uses one unit for each year. We deliberately "linearized" time in this time-scale and each year is considered to have exactly 365.25 days. There is thus no adjustment for lep years. Indeed, a small shift (less than one day) is introduced. This could result, for some dates, especially the 31st December, or 1st January of a year to be considered as belonging to the next, or previous year, respectively! Similarly, one month is considered to be 1/12 year, no mather if it has 28, 29, 30 or 31 days. Thus, the same warning applies: there are shifts in months introduced by this linearization of time! This representation simplifies further calculations, especially regarding seasonal effects (a quarter is exactly 0.25 units for instance), but shifts introduced in time may or may not be a problem for your particular application (if exact dates matters, do not use this; if shifts of up to one day is not significant, there is no problem, like when working on long-term biological series with years as units). Notice that converting it back to "days", using `yearstodays()` restablishes exact dates without errors. So, no data is lost, it just a conversion to a simplified (linearized) calendar!

Value

A numerical vector of the same length as `x` with the converted time-scale

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

`buysbal`

Examples

```
# A vector with a "days" time-scale (25 values every 30 days)
A <- (1:25)*30
# Convert it to a "years" time-scale, using 23/05/2001 (d/m/Y) as first value
B <- daystoyears(A, datemin="23/05/2001", dateformat="d/m/Y")
B
# Convert it back to "days" time-scale
yearstodays(B, xmin=30)

# Here is an example showing the shift introduced, and its consequence:
C <- daystoyears(unclass(as.Date(c("1970-1-1","1971-1-1","1972-1-1","1973-1-1"),
        format = "%Y-%m-%d")))
C
```

`decaverage` *Time series decomposition using a moving average*

Description

Decompose a single regular time series with a moving average filtering. Return a 'tsd' object. To decompose several time series at once, use `tsd()` with the argument `method="average"`

Usage

```
decaverage(x, type="additive", order=1, times=1, sides=2, ends="fill",
        weights=NULL)
```

Arguments

`x`	a regular time series ('rts' under S+ and 'ts' under R)
`type`	the type of model, either `type="additive"` (by default), or `type="multiplicative"`
`order`	the order of the moving average (the window of the average being 2*order+1), centered around the current observation or at left of this observation depending upon the value of the `sides` argument. Weights are the same for all observations within the window. However, if the argument `weights` is provided, it supersedes `order`. One can also use `order="periodic"`. In this case, a deseasoning filter is calculated according to the value of `frequency`
`times`	The number of times to apply the method (by default, once)
`sides`	If 2 (by default), the window is centered around the current observation. If 1, the window is at left of the current observation (including it)
`ends`	either "NAs" (fill first and last values that are not calculable with NAs), or "fill" (fill them with the average of observations before applying the filter, by default), or "circular" (use last values for estimating first ones and vice versa), or "periodic" (use entire periods of contiguous cycles, deseasoning)
`weights`	a vector indicating weight to give to all observations in the window. This argument has the priority over `order`

Details

This function is a wrapper around the `filter()` function and returns a 'tsd' object. However, it offers more methods to handle ends.

Value

A 'tsd' object

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Kendall, M., 1976. *Time-series.* Charles Griffin & Co Ltd. 197 pp.

Laloire, J.C., 1972. *Méthodes du traitement des chroniques.* Dunod, Paris, 194 pp.

Malinvaud, E., 1978. *Méthodes statistiques de l'économétrie.* Dunod, Paris. 846 pp.

Philips, L. & R. Blomme, 1973. *Analyse chronologique.* Université Catholique de Louvain. Vander ed. 339 pp.

See Also

tsd, tseries, deccensus, decdiff, decmedian, decevf, decreg, decloess

Examples

```
data(marbio)
ClausoB.ts <- ts(log(marbio$ClausocalanusB + 1))
ClausoB.dec <- decaverage(ClausoB.ts, order=2, times=10, sides=2, ends="fill")
plot(ClausoB.dec, col=c(1, 3, 2), xlab="stations")
# A stacked graph is more representative in this case
plot(ClausoB.dec, col=c(1, 3), xlab="stations", stack=FALSE, resid=FALSE,
        lpos=c(53, 4.3))
```

---

deccensus *Time decomposition using the CENSUS II method*

Description

The CENSUS II method allows to decompose a regular time series into a trend, a seasonal component and residuals, according to a multiplicative model

Usage

```
deccensus(x, type="multiplicative", trend=FALSE)
```

Arguments

x	A single regular time serie (a 'rts' object under S+ and a 'ts' object under R) with a "years" time scale (one unit = one year) and a complete number of cycles (at least 3 complete years)
type	The type of model. This is for compatibility with other decxxx() functions, but only a multiplicative model is allowed here
trend	If trend=TRUE a trend component is also calculated, otherwise, the decomposition gives only a seasonal component and residuals

Details

The trend component contains both a general trend and long-term oscillations. The seasonal trend may vary from year to year. For a seasonal decomposition using an additive model, use `decloess()` instead

Value

a 'tsd' object

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Béthoux, N., M. Etienne, F. Ibanez & J.L. Rapaire, 1980. *Spécificités hydrologiques des zones littorales. Analyse chronologique par la méthode CENSUS II et estimation des échanges océan-atmosphère appliqués à la baie de Villefranche sur mer.* Ann. Inst. Océanogr. Paris, 56:81-95.

Fromentin, J.M. & F. Ibanez, 1994. *Year to year changes in meteorological features on the French coast area during the last half-century. Examples of two biological responses.* Oceanologica Acta, 17:285-296.

Institut National de Statistique de Belgique, 1965. *Décomposition des séries chronologiques en leurs composantes suivant différentes méthodes. Etudes statistiques et économétriques.* Bull. Stat. INS, 10:1449-1524.

Philips, J. & R. Blomme, 1973. *Analyse chronologique.* Université Catholique de Louvain, Vander ed. 339 pp.

Rosenblatt, H.M., 1968. *Spectral evaluation of BLS and CENSUS revised seasonal adjustment procedures.* J. Amer. Stat. Assoc., 68:472-501.

Shiskin, J. & H. Eisenpress, 1957. *Seasonal adjustment by electronic computer methods.* J. Amer. Stat. Assoc., 52:415-449.

See Also

`tsd`, `tseries`, `decaverage`, `decdiff`, `decmedian`, `decevf`, `decreg`, `decloess`

Examples

```
data(releve)
# Get regulated time series with a 'years' time-scale
rel.regy <- regul(releve$Day, releve[3:8], xmin=6, n=87, units="daystoyears", frequency=24,
rel.ts <- tseries(rel.regy)
# We must have complete cycles to allow using deccensus()
start(rel.ts)
end(rel.ts)
rel.ts2 <- window(rel.ts, end=c(1992,5))
rel.dec2 <- deccensus(rel.ts2[, "Melosul"], trend=TRUE)
plot(rel.dec2, col=c(1,4,3,2))
```

`decdiff` *Time series decomposition using differences (trend elimination)*

Description

A filtering using X(t+lag) - X(t) has the property to eliminate the general trend from the series, whatever its shape

Usage

```
decdiff(x, type="additive", lag=1, order=1, ends="fill")
```

Arguments

`x`	a regular time series ('rts' under S+ and 'ts' under R)
`type`	the type of model, either `type="additive"` (by default), or `type="multiplicative"`
`lag`	The lag between the two observations used to calculate differences. By default, `lag=1`
`order`	The order of the difference corresponds to the number of times it is applied, by default `order=1`
`ends`	either "NAs" (fill first values that are not calculable with NAs), or "fill" (fill them with the average of following observations before applying the filter, by default), or "drop" (do not fill them). If `ends="drop"`, the filtered series will be shorter than the initial one by lag*order. In all other cases, the filtered series is as large as the initial one

Details

This function is a wrapper around the `diff()` function to create a 'tsd' object. It also manages initial values through the `ends` argument.

Value

a 'tsd' object

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Kendall, M., 1976. *Time-series.* Charles Griffin & Co Ltd. 197 pp.

Laloire, J.C., 1972. *Méthodes du traitement des chroniques.* Dunod, Paris, 194 pp.

Philips, L. & R. Blomme, 1973. *Analyse chronologique.* Université Catholique de Louvain. Vander ed. 339 pp.

See Also

tsd, tseries, decaverage, deccensus, decmedian, decevf, decreg, decloess

Examples

```
data(marbio)
ClausoB.ts <- ts(log(marbio$ClausocalanusB + 1))
ClausoB.dec <- decdiff(ClausoB.ts, lag=1, order=2, ends="fill")
plot(ClausoB.dec, col=c(1, 4, 2), xlab="stations")
```

---

decevf *Time series decomposition using eigenvector filtering (EVF)*

Description

The eigenvector filtering decomposes the signal by applying a principal component analysis (PCA) on the original signal and a certain number of copies of it incrementally lagged, collected in a multivariate matrix. Reconstructing the signal using only the most representative eigenvectors allows filtering it.

Usage

```
decevf(x, type="additive", lag=5, axes=1:2)
```

Arguments

x	a regular time series ('rts' under S+ and 'ts' under R)
type	the type of model, either type="additive" (by default), or type="multiplicative"
lag	The maximum lag used. A PCA is run on the matrix constituted by vectors lagged from 0 to lag. The defaulf value is 5, but a value corresponding to no significant autocorrelation, on basis of examination of the autocorrelation plot obtained by acf in the library 'ts' should be used (Lag at first time the autocorrelation curve crosses significance lines multiplied by the frequency of the series).
axes	The principal axes to use to reconstruct the filtered signal. For instance, to use axes 2 and 4, use axes=c(2,4). By default, the first two axes are considered (axes=1:2)

Value

a 'tsd' object

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Colebrook, J.M., 1978. *Continuous plankton records: zooplankton and environment, North-East Atlantic and North Sea 1948-1975.* Oceanologica Acta, 1:9-23.

Ibanez, F. & J.C. Dauvin, 1988. *Long-term changes (1977-1987) on a muddy fine sand Abra alba - Melinna palmate population community from the Western English Channel.* J. Mar. Prog. Ser., 49:65-81.

Ibanez, F., 1991. *Treatment of data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis.* In: B. Keegan (ed.) *Space and time series data analysis in coastal benthic ecology.* Pp. 5-43.

Ibanez, F. & M. Etienne, 1992. *Le filtrage des séries chronologiques par l'analyse en composantes principales de processus (ACPP).* J. Rech. Océanogr., 16:27-33.

Ibanez, F., J.C. Dauvin & M. Etienne, 1993. *Comparaison des évolutions à long-terme (1977-1990) de deux peuplements macrobenthiques de la Baie de Morlaix (Manche Occidentale): relations avec les facteurs hydroclimatiques.* J. Exp. Mar. Biol. Ecol., 169:181-214.

See Also

tsd, tseries, decaverage, deccensus, decmedian, decdiff, decreg, decloess

Examples

```
data(releve)
melo.regy <- regul(releve$Day, releve$Melosul, xmin=9, n=87,
        units="daystoyears", frequency=24, tol=2.2, methods="linear",
        datemin="21/03/1989", dateformat="d/m/Y")
melo.ts <- tseries(melo.regy)
acf(melo.ts)
# Autocorrelation is not significant after 0.16
# That corresponds to a lag of 0.16*24=4 (frequency=24)
melo.evf <- decevf(melo.ts, lag=4, axes=1)
plot(melo.evf, col=c(1, 4, 2))
# A superposed graph is better in the present case
plot(melo.evf, col=c(1, 4), xlab="stations", stack=FALSE, resid=FALSE,
        lpos=c(0, 60000))
```

---

decloess *Time series decomposition by the LOESS method*

Description

Compute a seasonal decomposition of a regular time series using a LOESS method (local polynomial regression)

Usage

```
decloess(x, type="additive", s.window=NULL, s.degree=0, t.window=NULL,
        t.degree=2, robust=FALSE, trend=FALSE)
```

Arguments

x	a regular time series ('rts' under S+ and 'ts' under R)
type	the type of model. This is for compatibility purpose. The only model type that is accepted for this method is type="additive". For a multiplicative model, use deccensus() instead
s.window	the width of the window used to extract the seasonal component. Use an odd value equal or just larger than the number of annual values (frequency of the time series). Use another value to extract other cycles (circadian, lunar,...). Using s.window="periodic" ensures a correct value for extracting a seasonal component when the time scale is in years units
s.degree	the order of the polynome to use to extract the seasonal component (0 or 1). By default s.degree=0
t.window	the width of the window to use to extract the general trend when trend=TRUE (indicate an odd value). If this parameter is not provided, a reasonable value is first calculated, and then used by the algorithm.
t.degree	the order of the polynome to use to extract the general trend (0, 1 or 2). By default t.degree=2
robust	if TRUE a robust regression method is used. Otherwise (FALSE), by default, a classical least-square regression is used
trend	If TRUE a trend is calculated (under R only). Otherwise, the series is decomposed into a seasonal component and residuals only

Details

This function uses the stl() function for the decomposition. It is a wrapper that create a 'tsd' object

Value

a 'tsd' object

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

References

Cleveland, W.S., E. Grosse & W.M. Shyu, 1992. *Local regression models. Chapter 8 of Statistical Models in S.* J.M. Chambers & T.J. Hastie (eds). Wadsworth & Brook/Cole.

Cleveland, R.B., W.S. Cleveland, J.E. McRae, & I. Terpenning, 1990. *STL: A seasonal-trend decomposition procedure based on loess.* J. Official Stat., 6:3-73.

See Also

tsd, tseries, decaverage, deccensus, decmedian, decdiff, decevf, decreg

Examples

```
data(releve)
melo.regy <- regul(releve$Day, releve$Melosul, xmin=9, n=87,
        units="daystoyears", frequency=24, tol=2.2, methods="linear",
        datemin="21/03/1989", dateformat="d/m/Y")
melo.ts <- tseries(melo.regy)
melo.dec <- decloess(melo.ts, s.window="periodic")
plot(melo.dec, col=1:3)
```

---

decmedian *Time series decomposition using a running median*

Description

This is a nonlinear filtering method used to smooth, but also to segment a time series. The isolated peaks and pits are leveraged by this method.

Usage

```
decmedian(x, type="additive", order=1, times=1, ends="fill")
```

Arguments

x	a regular time series ('rts' under S+ and 'ts' under R)
type	the type of model, either `type="additive"` (by default), or `type="multiplicative"`
order	the window used for the running median corresponds to 2*order + 1
times	the number of times the running median is applied. By default, 1
ends	the method used to calculate ends. Either "NAs" (fill extremes, non-calculable values with NAs), or "fill" (fill these extremes with the closest calculable median)

Value

a 'tsd' object

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Gebski, V.J., 1985. *Some properties of splicing when applied to non-linear smoothers.* Comp. Stat. Data Anal., 3:151-157.

Philips, L. & R. Blomme, 1973. *Analyse chronologique. Université Catholique de Louvain.* Vander ed. 339 pp.

Tukey, J.W., 1977. *Exploratory Data Analysis.* Reading Massachusetts: Addison-Wesley.

See Also

tsd, tseries, decaverage, deccensus, decdiff, decevf, decreg, decloess

Examples

```
data(marbio)
ClausoB.ts <- ts(log(marbio$ClausocalanusB + 1))
ClausoB.dec <- decmedian(ClausoB.ts, order=2, times=10, ends="fill")
plot(ClausoB.dec, col=c(1, 4, 2), xlab="stations")
# This is a transect across a frontal zone:
plot(ClausoB.dec, col=c(0, 2), xlab="stations", stack=FALSE, resid=FALSE)
lines(c(17, 17), c(0, 10), col=4, lty=2)
lines(c(25, 25), c(0, 10), col=4, lty=2)
lines(c(30, 30), c(0, 10), col=4, lty=2)
lines(c(41, 41), c(0, 10), col=4, lty=2)
lines(c(46, 46), c(0, 10), col=4, lty=2)
text(c(8.5, 21, 27.5, 35, 43.5, 57), 8.7, labels=c("Peripheral Zone", "D1",
        "C", "Front", "D2", "Central Zone"))
```

---

decreg *Time series decomposition using a regression model*

Description

Providing values coming from a regression on the original series, a `tsd` object is created using the original series, the regression model and the residuals

Usage

```
decreg(x, xreg, type="additive")
```

Arguments

`x`	a regular time series ('rts' under S+ and 'ts' under R)
`xreg`	a second regular time series or a vector of the same length as `x` with corresponding values from the regression model
`type`	the type of model, either `type="additive"` (by default), or `type="multiplicative"`

Value

a 'tsd' object

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Frontier, S., 1981. *Méthodes statistiques.* Masson, Paris. 246 pp.

Kendall, M., 1976. *Time-series.* Charles Griffin & Co Ltd. 197 pp.

Legendre, L. & P. Legendre, 1984. *Ecologie numérique. Tome 2: La structure des données écologiques.* Masson, Paris. 335 pp.

Malinvaud, E., 1978. *Méthodes statistiques de l'économétrie.* Dunod, Paris. 846 pp.

Sokal, R.R. & F.J. Rohlf, 1981. *Biometry.* Freeman & Co, San Francisco. 860 pp.

See Also

tsd, tseries, decaverage, deccensus, decdiff, decevf, decmedian, decloess

Examples

```
data(marphy)
density <- ts(marphy[, "Density"])
plot(density)
Time <- time(density)

# Linear model to represent trend
density.lin <- lm(density ~ Time)
summary(density.lin)
xreg <- predict(density.lin)
lines(xreg, col=3)
density.dec <- decreg(density, xreg)
plot(density.dec, col=c(1, 3, 2), xlab="stations")

# Order 2 polynomial to represent trend
density.poly <- lm(density ~ Time + I(Time^2))
summary(density.poly)
xreg2 <- predict(density.poly)
plot(density)
lines(xreg2, col=3)
density.dec2 <- decreg(density, xreg2)
plot(density.dec2, col=c(1, 3, 2), xlab="stations")

# Fit a sinusoidal model on seasonal (artificial) data
tser <- ts(sin((1:100)/12*pi)+rnorm(100, sd=0.3), start=c(1998, 4),
        frequency=24)
Time <- time(tser)
tser.sin <- lm(tser ~ I(cos(2*pi*Time)) + I(sin(2*pi*Time)))
summary(tser.sin)
tser.reg <- predict(tser.sin)
tser.dec <- decreg(tser, tser.reg)
plot(tser.dec, col=c(1, 4), xlab="stations", stack=FALSE, resid=FALSE,
        lpos=c(0, 4))
plot(tser.dec, col=c(1, 4, 2), xlab="stations")

# One can also use nonlinear models (see 'nls')
# or autoregressive models (see 'ar' and others in 'ts' library)
```

`disjoin` *Complete disjoined coded data (binary coding)*

Description

Transform a factor in separate variables (one per level) with a binary code (0 for absent, 1 for present) in each variable

Usage

```
disjoin(x)
```

Arguments

`x`	a vector containing a factor data

Details

Use `cut()` to transform a numerical variable into a factor variable

Value

a matrix containing the data with binary coding

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Fromentin J.-M., F. Ibanez & P. Legendre, 1993. *A phytosociological method for interpreting plankton data.* Mar. Ecol. Prog. Ser., 93:285-306.

Gebski, V.J., 1985. *Some properties of splicing when applied to non-linear smoothers.* Comput. Stat. Data Anal., 3:151-157.

Grandjouan, G., 1982. *Une méthode de comparaison statistique entre les répartitions des plantes et des climats.* Thèse d'Etat, Université Louis Pasteur, Strasbourg.

Ibanez, F., 1976. *Contribution à l'analyse mathématique des événements en Ecologie planctonique. Optimisations méthodologiques.* Bull. Inst. Océanogr. Monaco, 72:1-96.

See Also

`buysbal`, `cut`

Examples

```
# Artificial data with 1/5 of zeros
Z <- c(abs(rnorm(8000)), rep(0, 2000))
# Let the program chose cuts
table(cut(Z, breaks=5))
# Create one class for zeros, and 4 classes for the other observations
Z2 <- Z[Z != 0]
cuts <- c(-1e-10, 1e-10, quantile(Z2, 1:5/5, na.rm=TRUE))
cuts
table(cut(Z, breaks=cuts))
# Binary coding of these data
disjoin(cut(Z, breaks=cuts))[1:10, ]
```

---

disto *Compute and plot a distogram*

Description

A distogram is an extension of the variogram to a multivariate time-series. It computes, for each observation (with a constant interval h between each observation), the euclidean distance normated to one (chord distance)

Usage

```
disto(x, max.dist=nrow(x)/4, plotit=TRUE, disto.data=NULL)
```

Arguments

x	a matrix, a data frame or a multiple time-series
max.dist	the maximum distance to calculate. By default, it is the third of the number of observations (that is, the number of rows in the matrix)
plotit	If `plotit=TRUE` then the graph of the distogram is plotted
disto.data	data coming from a previous call to `disto()`. Call the function again with these data to plot the corresponding graph

Value

A data frame containing distance and distogram values

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Dauvin, J.C. & F. Ibanez, 1986. *Variations à long-terme (1977-1985) du peuplement des sables fins de la Pierre Noire (baie de Morlaix, Manche Occidentale): analyse statistique de l'évolution structurale.* Hydrobiologia, 142:171-186.

Ibanez, F. & J.C. Dauvin, 1988. *Long-term changes (1977-1987) in a muddy fine sand Abra alba - Melinna palmate community from the Western English Channel: multivariate time-series analysis.* Mar. Ecol. Prog. Ser., 49:65-81.

Mackas, D.L., 1984. *Spatial autocorrelation of plankton community composition in a continental shelf ecosystem.* Limnol. Ecol., 20:451-471.

See Also

vario

Examples

```
data(bnr)
disto(bnr)
```

`escouf` *Choose variables using the Escoufier's equivalent vectors method*

Description

Calculate equivalent vectors sensu Escoufier, that is, most significant variables from a multivariate data frame according to a principal component analysis (variables that are most correlated with the principal axes). This method is useful mainly for physical or chemical data where simply summarizing them with a PCA does not always gives easily interpretable principal axes.

Usage

```
escouf(x, level=1, verbose=TRUE)
## S3 method for class 'escouf':
summary(esc)
## S3 method for class 'escouf':
plot(esc, level=x$level, lhorz=TRUE, lvert=TRUE, lvars=TRUE,
        lcol=2, llty=2, diff=TRUE, dlab="RV' (units not shown)", dcol=4,
        dlty=par("lty"), dpos=0.8, ...)
## S3 method for class 'escouf':
lines(esc, level=x$level, lhorz=TRUE, lvert=TRUE, lvars=TRUE,
        lcol=2, llty=2, ...)
## S3 method for class 'escouf':
identify(esc, lhorz=TRUE, lvert=TRUE, lvars=TRUE, lcol=2,
        llty=2, ...)
## S3 method for class 'escouf':
extract(esc, n=NULL, level=e$level)
```

Arguments

x	A data frame containing the variables to sort according to the Escoufier's method
level	The level of correlation at which to stop calculation. By default level=1, the highest value, and all variables are sorted. Specify a value lower than one to speed up calculation. If you specify a too low values you will not be able to extract all significant variables (extraction level must be lower than calculation level). We advise you keep 0.95 < level < 1
verbose	Print calculation steps. This allows to control the percentage of calculation already achieved when computation takes a long time (that is, with many variables to sort)
esc	An 'escouf' object returned by escouf
lhorz	If TRUE then an horizontal line indicating the extraction level is drawn
lvert	If TRUE then a vertical line separate the n extracted variables at left from the rest
lvars	If TRUE then the x-axis labels of the n extracted variables at left are printed in a different color to emphasize them
lcol	The color to use to draw the lines (lhorz=TRUE and lvert=TRUE) and the variables labels (lvars=TRUE) of the n extracted variables. By default, color 2 is used
llty	The style used to draw the lines (lhorz=TRUE and lvert=TRUE). By default, lines are dashed
diff	If TRUE then the RV' curve is also plotted (by default)
dlab	The label to use for the RV' curve. By default: "RV' (units not shown)"
dcol	The color to use for the RV' curve (by default, color 4 is used)
dlty	The style for the RV' curve
dpos	The relative horizontal position of the label for the RV' curve. The default value of 0.8 means that the label is placed at 80% of the horizontal axis.Vertical position of the label is automatically determined
...	additional graph parameters
n	The number of variables to extract. If a value is given, it has the priority on level

Value

An object of type 'escouf' is returned. It has methods print(), summary(), plot(), lines(), identify(), extract().

WARNING

Since a large number of iterations is done, this function is slow with a large number of variables (more than 25-30)!

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Benjamin Planque (⟨Benjamin.Planque@ifremer.fr⟩), Jean-Marc Fromentin (⟨Jean.Marc.Fromentin@ifremer.fr⟩)

References

Cambon, J., 1974. *Vecteur équivalent à un autre au sens des composantes principales.* Application hydrologique. DEA de Mathématiques Appliquées, Université de Montpellier.

Escoufier, Y., 1970. *Echantillonnage dans une population de variables aléatoires réelles.* Pub. Inst. Stat. Univ. Paris, 19:1-47.

Jabaud, A., 1996. *Cadre climatique et hydrobiologique du lac Léman.* DEA d'Océanologie Biologique Paris.

See Also

abund

Examples

```
data(marbio)
marbio.esc <- escouf(marbio)
summary(marbio.esc)
plot(marbio.esc)
# The x-axis has short labels. For more info., enter:
marbio.esc$vr
# Define a level at which to extract most significant variables
marbio.esc$level <- 0.90
# Show it on the graph
lines(marbio.esc)
# This can also be done interactively on the plot using:
# marbio.esc$level <- identify(marbio.esc)
# Finally, extract most significant variables
marbio2 <- extract(marbio.esc)
names(marbio2)
```

---

extract *Extract a subset of the original dataset*

Description

'extract' is a generic function for extracting a part of the original dataset according to an analysis...)

Usage

```
extract(e, n, ...)
```

Arguments

e	An object from where extraction is performed
n	The number of elements to extract
...	Additional arguments affecting the extraction

Value

A subset of the original dataset contained in the extracted object

See Also

abund, regul, tsd, turnogram, turnpoints

---

first *Get the first element of a vector*

Description

Extract the first element of a vector. Useful for the turnogram() function

Usage

```
first(x, na.rm=FALSE)
```

Arguments

x	a numerical vector
na.rm	if na.rm=TRUE, then the first non-missing value (if any) is returned. By default, it is FALSE and the first element (whatever its value) is returned

Value

a numerical value

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

last, turnogram

Examples

```
a <- c(NA, 1, 2, NA, 3, 4, NA)
first(a)
first(a, na.rm=TRUE)
```

GetUnitText *Format a nice time units for labels in graphs*

Description

This is an internal function called by some `plot()` methods. Considering the time series 'units' attribute and the frequency of the observations in the series, the function returns a string with a pertinent time unit. For instance, if the unit is 'years' and the frequency is 12, then data are monthly sampled and `GetUnitText()` returns the string "months"

Usage

```
GetUnitText(series)
```

Arguments

`series`	a regular time series (a 'rts' object in Splus, or a 'ts' object in R)

Value

a string with the best time unit for graphs

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

`daystoyears`, `yearstodays`

Examples

```
timeser <- ts(1:24, frequency=12)           # 12 observations per year
if (exists("is.R") && is.function(is.R) && is.R()) {  # We are in R
attr(timeser, "units") <- "years"           # time in years for 'ts' object
} else {                                              # We are in Splus
attr(attr(timeser, "tspar"), "units") <- "years" # Idem for Splus 'rts' object
}
GetUnitText(timeser)                        # formats unit (1/12 year=months)
```

is.tseries *Is this object a time series?*

Description

This is equivalent to `is.rts()` in Splus and to `is.ts()` in R. `is.tseries()` recognizes both 'rts' and 'ts' objects whatever the environment (Splus or R)

Usage

```
is.tseries(x)
```

Arguments

`x`	an object

Value

a boolean value

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

`tseries`

Examples

```
tser <- ts(sin((1:100)/6*pi)+rnorm(100, sd=0.5), start=c(1998, 4), frequency=12)
is.tseries(tser)      # TRUE
not.a.ts <- c(1,2,3)
is.tseries(not.a.ts)  # FALSE
```

last *Get the last element of a vector*

Description

Extract the last element of a vector. Useful for the `turnogram()` function

Usage

```
last(x, na.rm=FALSE)
```

Arguments

x	a numerical vector
na.rm	if na.rm=TRUE, then the last non-missing value (if any) is returned. By default, it is FALSE and the last element (whatever its value) is returned

Value

a numerical value

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

first, turnogram

Examples

```
a <- c(NA, 1, 2, NA, 3, 4, NA)
last(a)
last(a, na.rm=TRUE)
```

---

local.trend *Calculate local trends using cumsum*

Description

A simple method using cumulated sums that allows to detect changes in the tendency in a time series

Usage

```
local.trend(x, k=mean(x), plotit=TRUE, ...)
## S3 method for class 'local.trend':
identify(loctrd)
```

Arguments

x	a regular time series (a 'rts' object under S+ or a 'ts' object under R)
k	the reference value to substract from cumulated sums. By default, it is the mean of all observations in the series
plotit	if plotit=TRUE (by default), a graph with the cumsum curve superposed to the original series is plotted
...	additional arguments for the graph
loctrd	a 'local.trend' object, as returned by the function local.trend()

Details

With `local.trend()`, you can:

- detect changes in the mean value of a time series

- determine the date of occurence of such changes

- estimate the mean values on homogeneous intervals

Value

a 'local.trend' object is returned. It has the method `identify()`

Note

Once transitions are identified with this method, you can use `stat.slide()` to get more detailed information on each phase. A smoothing of the series using running medians (see `decmedian()`) allows also to detect various levels in a time series, but according to the median statistic. Under R, see also the 'strucchange' package for a more complete, but more complex, implementation of cumsum applied to time series.

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Ibanez, F., J.M. Fromentin & J. Castel, 1993. *Application de la méthode des sommes cumulées à l'analyse des séries chronologiques océanographiques.* C. R. Acad. Sci. Paris, Life Sciences, 316:745-748.

See Also

`trend.test`, `stat.slide`, `decmedian`

Examples

```
data(bnr)
# Calculate and plot cumsum for the 8th series
bnr8.lt <- local.trend(bnr[,8])
# To identify local trends, use:
# identify(bnr8.lt)
# and click points between which you want to compute local linear trends...
```

marbio *Several zooplankton taxa measured across a transect*

Description

The marbio data frame has 68 rows (stations) and 24 columns (taxonomic zooplankton groups). Abundances are expressed in no per cubic meter of seawater.

Usage

```
data(marbio)
```

Format

This data frame contains the following columns giving abundance of:

Acartia Acartia clausi - herbivorous

AdultsOfCalanus Calanus helgolandicus (adults) - herbivorous

Copepodits1 Idem (copepodits 1) - omnivorous

Copepodits2 Idem (copepodits 2) - omnivorous

Copepodits3 Idem (copepodits 3) - omnivorous

Copepodits4 Idem (copepodits 4) - omnivorous

Copepodits5 Idem (copepodits 5) - omnivorous

ClausocalanusA Clausocalanus size class A - herbivorous

ClausocalanusB Clausocalanus size class B - herbivorous

ClausocalanusC Clausocalanus size class C - herbivorous

AdultsOfCentropages Centropages tipicus (adults) - omnivorous

JuvenilesOfCentropages Centropages typicus (juv.) - omnivorous

Nauplii Nauplii of copepods - filter feeders

Oithona Oithona sp. - carnivorous

Acanthaires Various species of acanthaires - misc

Cladocerans Various species of cladocerans - carnivorous

EchinodermsLarvae Larvae of echinoderms - filter feeders

DecapodsLarvae Larvae of decapods - omnivorous

GasteropodsLarvae Larvae of gastropods - filter feeders

EggsOfCrustaceans Eggs of crustaceans - misc

Ostracods Various species of ostracods - omnivorous

Pteropods Various species of pteropods - herbivorous

Siphonophores Various species of siphonophores - carnivorous

BellsOfCalycophores Bells of calycophores - misc

Details

This dataset corresponds to a single transect sampled with a plankton net across the Ligurian Sea front in the Mediterranean Sea, between Nice (France) and Calvi (Corsica). The transect extends from the Cap Ferrat (close to Nice) to about 65 km offshore in the direction of Calvi (along a bearing of 123°). 68 regularly spaced samples where collected on this transect. For more information about the water masses and their physico-chemical characteristics, see the marphy dataset.

Source

Boucher, J., F. Ibanez & L. Prieur (1987) *Daily and seasonal variations in the spatial distribution of zooplankton populations in relation to the physical structure in the Ligurian Sea Front.* Journal of Marine Research, 45:133-173.

References

Fromentin, J.-M., F. Ibanez & P. Legendre (1993) *A phytosociological method for interpreting plankton data.* Marine Ecology Progress Series, 93:285-306.

See Also

marphy

---

marphy *Physico-chemical records at the same stations as for marbio*

Description

The marphy data frame has 68 rows (stations) and 4 columns. They are seawater measurements at a deep of 3 to 7 m at the same 68 stations as marbio.

Usage

```
data(marphy)
```

Format

This data frame contains the following columns:

Temperature Seawater temperature in °C

Salinity Salinity in g/kg

Fluorescence Fluorescence of chlorophyll a

Density Excess of volumic mass of the seawater in g/l

Details

This dataset corresponds to a single transect sampled across the Ligurian Sea front in the Mediterranean Sea, between Nice (France) and Calvi (Corsica). The transect extends from the Cap Ferrat (close to Nice) to about 65 km offshore in the direction of Calvi (along a bearing of 123°). 68 regularly spaced measurements where recorded. They correspond to the stations where zooplankton was collected in the `marbio` dataset. Water masses in the transect across the front where identified as:

Stations 1 to 17 Peripheral zone

Stations 17 to 25 D1 (divergence) zone

Stations 25 to 30 C (convergence) zone

Stations 30 to 41 Frontal zone

Stations 41 to 46 D2 (divergence) zone

Stations 46 to 68 Central zone

Source

Boucher, J., F. Ibanez & L. Prieur (1987) *Daily and seasonal variations in the spatial distribution of zooplankton populations in relation to the physical structure in the Ligurian Sea Front.* Journal of Marine Research, 45:133-173.

References

Fromentin, J.-M., F. Ibanez & P. Legendre (1993) *A phytosociological method for interpreting plankton data.* Marine Ecology Progress Series, 93:285-306.

See Also

`marbio`

---

match.tol *Determine matching observation with a tolerance in time-scale*

Description

Determine which observations in a regular time series match observation in an original irregular one, accepting a given tolerance window in the time-scale. This function is internally used for regulation (functions `regul()`, `regul.screen()` and `regul.adj()`

Usage

```
match.tol(x, table, nomatch=NA, tol.type="both", tol=0)
```

Arguments

x	a numerical vector containing seeked values (time-scale of the regular series)
table	a numerical vector containing initial times to look for match in x
nomatch	the symbol tu use to flag an entry where no match is found. By default, nomatch=NA
tol.type	the type of adjustment to use for the time-tolerance: "left", "right", "both" (by default) or "none". If tol.type="left", corresponding x values are seeked in a window]table-tol, table]. If tol.type="right", they are seeked in the window [table, table+tol[. If tol.type="both", then they are seeked in the window]table-tol, table+tol]. If several observations are in this window, the closest one is used. Finally, if tol.type="none", then the function returns the nomatch symbol for all entries
tol	the tolerance in the time-scale to determine if a value in x matches a value in table. If tol=0, observations in each respective series must match exactly, otherwise observations in the regulated series are interpolated. tol must be a round fraction of the interval between observations in x (x[i+1] - x[i], (x[i+1] - x[i])/2, (x[i+1] - x[i])/3, etc...), and cannot be larger than it, otherwise, tol is automatically adjusted to the closest allowed value. By default, tol=NULL. This is equivalent to tol=0. Warning!

Value

a vector of the same length of x, indicating the position of the matching observations in table

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

regul, regul.screen, regul.adj

Examples

```
X <- 1:5
Table <- c(1, 3.1, 3.8, 4.4, 5.1, 6)
match.tol(X, Table)
match.tol(X, Table, tol=0.2)
match.tol(X, Table, tol=0.55)
```

---

pennington *Calculate Pennington statistics*

Description

Calculate the mean, the variance and the variance of the mean for a single series, according to Pennington (correction for zero/missing values for abundance of species collected with a net)

Usage

```
pennington(x, calc="all", na.rm=FALSE)
```

Arguments

x	a numerical vector
calc	indicate which estimator(s) should be calculated. Use: "mean", "var", "mean.var" or "all" (by default) for the mean, the variance, the variance of the mean or all these three statitics, respectively
na.rm	if na.rm=TRUE, missing data are eliminated first. If it is FALSE (by default), any missing value in the series leads to NA as the result for the statistic

Details

A complex problem in marine ecology is the notion of zero. In fact, the random sampling of a fish population often leads to a table with many null values. Do these null values indicate that the fish was absent or do we have to consider these null values as missing data, that is, that the fish was rare but present and was not caught by the net? For instance, for 100 net trails giving 80 null values, how to estimate the mean? Do we have to divide the sum of fishes caught by 100 or by 20?

Pennington (1983) applied in this case the probabilistic theory of Aitchison (1955). The later established a method to estimate mean and variance of a random variable with a non-null probability to present zero values and whose the conditional distribution corresponding to its non-null values follows a Gaussian curve. In practice, a lognormal distribution is found most of the time for non-null values in fishes population. It is also often the case for the plankton, after our own experience. pennington() calculates thus statistics for such conditional lognormal distributions.

Value

a single value, or a vector with "mean", "var" and "mean.var" if the argument calc="all"

Note

For multiple series, or for getting more statistics on the series, use stat.pen(). Use stat.slide() for obtaining statistics calculated separately for various intervals along the time for a time series

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Aitchison, J., 1955. *On the distribution of a positive random variable having a discrete probability mass at the origin.* J. Amer. Stat. Ass., 50:901-908.

Pennington, M., 1983. *Efficient estimations of abundance for fish and plankton surveys.* Biometrics, 39:281-286.

See Also

stat.pen, stat.slide

Examples

```
data(marbio)
pennington(marbio[, "Copepodits2"])
pennington(marbio[, "Copepodits2"], calc="mean", na.rm=TRUE)
```

---

pgleissberg *Gleissberg distribution probability*

Description

The Gleissberg distribution gives the probability to have k extrema in a series of n observations. This distribution is used in the turnogram to determine if monotony indices are significant (see `turnogram()`)

Usage

```
pgleissberg(n, k, lower.tail=TRUE, two.tailed=FALSE)
```

Arguments

`n`	the number of observations in the series
`k`	the number of extrema in the series, as calculated by `turnpoints()`
`lower.tail`	if `lower.tail=TRUE` (by default) and `two.tailed=FALSE`, the left-side probability is returned. If it is FALSE, the right-side probability is returned
`two.tailed`	if `two.tailed=TRUE`, the two-sided probability is returned. By default, it is FALSE and a one-sided probability is returned (left or right, depending on the value of `lower.tail`

Value

a value giving the probability to have `k` extrema in a series of `n` observations

Note

The Gleissberg distribution is asymptotically normal. For `n` > 50, the distribution is approximated by a Gaussian curve. For lower `n` values, the exact probability is returned (using data in the variable `.gleissberg.table`

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Dallot, S. & M. Etienne, 1990. *Une méthode non paramétrique d'analyse des séries en océanographie biologique: les tournogrammes.* Biométrie et océanographie - Société de biométrie, 6, Lille, 26-28 mai 1986. IFREMER, Actes de colloques, 10:13-31.

Johnson, N.L. & Kotz, S., 1969. *Discrete distributions.* J. Wiley & sons, New York, 328 pp.

See Also

.gleissberg.table, turnpoints, turnogram

Examples

```
# Until n=50, the exact probability is returned
pgleissberg(20, 10, lower.tail=TRUE, two.tailed=FALSE)
# For higher n values, it is approximated by a normal distribution
pgleissberg(60, 33, lower.tail=TRUE, two.tailed=FALSE)
```

---

regarea *Regulate a series using the area method*

Description

Transform an irregular time series in a regular time series, or fill gaps in regular time series using the area method

Usage

```
regarea(x, y=NULL, xmin=min(x), n=length(x),
        deltat=(max(x) - min(x))/(n - 1), rule=1,
        window=deltat, interp=FALSE, split=100)
```

Arguments

x	a vector with time in the irregular series. Missing values are allowed
y	a vector of same length as x and holding observations at corresponding times
xmin	allows to respecify the origin of time in the calculated regular time series. By default, the origin is not redefined and it is equivalent to the smallest value in x
n	the number of observations in the regular time series. By default, it is the same number than in the original irregular time series (i.e., length(x)
deltat	the time interval between two observations in the regulated time series
rule	the rule to use for extrapolated values (outside of the range in the initial irregular time series) in the regular time series. With rule=1 (by default), these entries are not calculated and get NA; with rule=2, these entries are extrapolated
window	size of the window to use for interpolation. By default, adjacent windows are used (window=deltat)
interp	indicates if matching observations in both series must be calculated (interp=TRUE), or if initial observations are used "as is" in the final regular series (interp=FALSE, by default)
split	a parameter to optimise calculation time and to avoid saturation of the memory. Very long time series are splitted into smaller subunits. This is transparent for the user. The default value of split=100 should be rarely changed. Give a lower value if the program fails and reports a memory problem during calculation

Details

This is a linear interpolation method described by Fox (1972). It takes into account all observations located in a given time window around the missing observation. On the contrary to many other interpolations (constant, linear, spline), the interpolated curve does not pass by the initial observations. Indeed, a smoothing is also operated simultaneously by this method. The importance of the smoothing is dependent on the size of the window (the largest it is, the smoothest will be the calculated regular time series)

Value

An object of type 'regul' is returned. It has methods `print()`, `summary()`, `plot()`, `lines()`, `identify()`, `hist()`, `extract()` and `specs()`.

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

References

Fox, W.T. & J.A. Brown, 1965. *The use of time-trend analysis for environmental interpretation of limestones.* J. Geol., 73:510-518.

Ibanez, F., 1991. *Treatment of the data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis.* In: B. Keegan (ed). Space and Time Series Data Analysis in Coastal Benthic Ecology. Pp 5-43.

Ibanez, F. & J.C. Dauvin, 1988. *Long-term changes (1977-1987) on a muddy fine sand Abra alba - Melinna palmata population community from the Western English Channel.* J. Mar. Ecol. Prog. Ser., 49:65-81.

See Also

`regul`, `regconst`, `reglin`, `regspline`, `regul.screen`, `regul.adj`, `tseries`, `is.tseries`

Examples

```
data(releve)
# The 'Melosul' series is regulated with a 25-days window
reg <- regarea(releve$Day, releve$Melosul, window=25)
# Then with a 50-days window
reg2 <- regarea(releve$Day, releve$Melosul, window=50)
# The initial and both regulated series are shown on the graph for comparison
plot(releve$Day, releve$Melosul, type="l")
lines(reg$x, reg$y, col=2)
lines(reg2$x, reg2$y, col=4)
```

`regconst` *Regulate a series using the constant value method*

Description

Transform an irregular time series in a regular time series, or fill gaps in regular time series using the constant value method

Usage

```
regconst(x, y=NULL, xmin=min(x), n=length(x),
        deltat=(max(x) - min(x))/(n - 1), rule=1, f=0)
```

Arguments

`x`	a vector with time in the irregular series. Missing values are allowed
`y`	a vector of same length as `x` and holding observations at corresponding times
`xmin`	allows to respecify the origin of time in the calculated regular time series. By default, the origin is not redefined and it is equivalent to the smallest value in `x`
`n`	the number of observations in the regular time series. By default, it is the same number than in the original irregular time series (i.e., `length(x)`
`deltat`	the time interval between two observations in the regulated time series
`rule`	the rule to use for extrapolated values (outside of the range in the initial irregular time series) in the regular time series. With `rule=1` (by default), these entries are not calculated and get `NA`; with `rule=2`, these entries are extrapolated
`f`	coefficient giving more weight to the left value (`f=0`, by default), to the right value (`f=`) or to a combination of these two observations (0 < f <1)

Details

This is the simplest, but the less powerful regulation method. Interpolated values are calculated according to existing observations at left and at right as: x[reg] = x[right]*f + x[left]*(f-1), with 0 < f < 1.

Value

An object of type 'regul' is returned. It has methods `print()`, `summary()`, `plot()`, `lines()`, `identify()`, `hist()`, `extract()` and `specs()`.

Note

This function uses `approx()` for internal calculations

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

regul, regarea, reglin, regspline, regul.screen, regul.adj, tseries, is.tseries, approxfun

Examples

```
data(releve)
reg <- regconst(releve$Day, releve$Melosul)
plot(releve$Day, releve$Melosul, type="l")
lines(reg$x, reg$y, col=2)
```

---

reglin *Regulation of a series using a linear interpolation*

Description

Transform an irregular time series in a regular time series, or fill gaps in regular time series using a linear interpolation

Usage

```
reglin(x, y=NULL, xmin=min(x), n=length(x),
        deltat=(max(x) - min(x))/(n - 1), rule=1)
```

Arguments

x	a vector with time in the irregular series. Missing values are allowed
y	a vector of same length as x and holding observations at corresponding times
xmin	allows to respecify the origin of time in the calculated regular time series. By default, the origin is not redefined and it is equivalent to the smallest value in x
n	the number of observations in the regular time series. By default, it is the same number than in the original irregular time series (i.e., length(x)
deltat	the time interval between two observations in the regulated time series
rule	the rule to use for extrapolated values (outside of the range in the initial irregular time series) in the regular time series. With rule=1 (by default), these entries are not calculated and get NA; with rule=2, these entries are extrapolated

Details

Observed values are connected by lines and interpolated values are obtained from this "polyline".

Value

An object of type 'regul' is returned. It has methods print(), summary(), plot(), lines(), identify(), hist(), extract() and specs().

Note

This function uses `approx()` for internal calculations

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

`regul`, `regarea`, `regconst`, `regspline`, `regul.screen`, `regul.adj`, `tseries`, `is.tseries`, `approxfun`

Examples

```
data(releve)
reg <- reglin(releve$Day, releve$Melosul)
plot(releve$Day, releve$Melosul, type="l")
lines(reg$x, reg$y, col=2)
```

---

regspline *Regulation of a time series using splines*

Description

Transform an irregular time series in a regular time series, or fill gaps in regular time series using splines

Usage

```
regspline(x, y=NULL, xmin=min(x), n=length(x),
        deltat=(max(x) - min(x))/(n - 1), rule=1, periodic=FALSE)
```

Arguments

`x`	a vector with time in the irregular series. Missing values are allowed
`y`	a vector of same length as `x` and holding observations at corresponding times
`xmin`	allows to respecify the origin of time in the calculated regular time series. By default, the origin is not redefined and it is equivalent to the smallest value in `x`
`n`	the number of observations in the regular time series. By default, it is the same number than in the original irregular time series (i.e., `length(x)`
`deltat`	the time interval between two observations in the regulated time series
`rule`	the rule to use for extrapolated values (outside of the range in the initial irregular time series) in the regular time series. With `rule=1` (by default), these entries are not calculated and get `NA`; with `rule=2`, these entries are extrapolated

`periodic` indicates if the time series should be considered as periodic (`periodic=TRUE`, first value must be equal to the last one). If this is the case, first and second derivates used to calculate spline segments around first and last observations use data in the other extreme of the series. In the other case (`periodic=FALSE` (by default), derivates for extremes observations are considered to be equal to zero

Details

Missing values are interpolated using cubic splines between observed values.

Value

An object of type 'regul' is returned. It has methods `print()`, `summary()`, `plot()`, `lines()`, `identify()`, `hist()`, `extract()` and `specs()`.

Note

This function uses `spline()` for internal calculations. However, interpolated values are not allowed to be higher than the largest initial observation or lower than the smallest one.

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Lancaster, P. & K. Salkauskas, 1986. *Curve and surface fitting.* Academic Press, England, 280 pp.

See Also

`regul`, `regarea`, `regconst`, `reglin`, `regul.screen`, `regul.adj`, `tseries`, `is.tseries`, `splinefun`

Examples

```
data(releve)
reg <- regspline(releve$Day, releve$Melosul)
plot(releve$Day, releve$Melosul, type="l")
lines(reg$x, reg$y, col=2)
```

`regul` *Regulation of one or several time series using various methods*

Description

Regulate irregular time series or regular time series with gaps. Create a `regul` object from whose one or several regular time series can be extracted using `extract()` or `tseries()`. This is the function to apply most of the time to create regular time series ('rts' objects in Splus or 'ts' objects in R) that will be further analyzed by other functions that apply to regular time series.

Usage

```
regul(x, y=NULL, xmin=min(x), n=length(x), units="days", frequency=NULL,
        deltat=1/frequency, datemin=NULL, dateformat="m/d/Y", tol=NULL,
        tol.type="both", methods="linear", rule=1, f=0, periodic=FALSE,
        window=(max(x) - min(x))/(n - 1), split=100, specs=NULL)
## S3 method for class 'regul':
summary(reg)
## S3 method for class 'regul':
plot(reg, series=1, col=c(1, 2), lty, plot.pts=TRUE,
        leg=FALSE, llab=c("initial", x$specs$methods[series]),
        lpos=c(1.5, 10), ...)
## S3 method for class 'regul':
lines(reg, series=1, col=3, lty=1, plot.pts=TRUE, ...)
## S3 method for class 'regul':
identify(reg, series=1, col=3, label="#", ...)
## S3 method for class 'regul':
hist(reg, nclass=30, col=c(4, 5, 2), plotit=TRUE, ...)
## S3 method for class 'regul':
extract(reg, n=ncol(reg$y), series=NULL)
## S3 method for class 'regul':
specs(reg)
```

Arguments

`x`	a vector containing times at which observations are sampled in the initial irregular time series. It can be expressed in any unit ("years", "days", "weeks", "hours", "min", "sec",...) as defined by the argument `units`. It is often expressed in "days" and the decimal part represents the part of the day, that is the time in hour:min:sec (dates coming from Excel, or even standard dates in S+ or R are expressed like that)
`y`	a vector (single series) or a matrix/data frame whose columns correspond to the various irregular time series to regulate. Rows are observations made at corresponding times in `x`. The number of rows must thus match the length of vector `x`
`xmin`	allows to respecify the origin of time in `x`. By default, the origin is not redefined and thus, the smallest value in `x` is used

`n` the number of observations in the regular time series. By default, it is the same number than in the original irregular time series (i.e., `length(x)`

`units` the time unit for the `x` vector. By default `units="days"`. A special value, `units="daystoyears"` indicates that `x` is expressed in "days" (1 unit = 1 day) but that we want to obtain the final regular time series expressed in "years" (1 unit = 1 year). Give a correct value for `datemin` to make sure the right fraction of the year is computed for each observation (see example hereunder)

`frequency` the frequency of the regulated time series in the corresponding time unit. For instance, `frequency=12` with `units="years"` means montly sampled observations. Warning! When using `units="daystoyears"`, specify `frequency` (or `deltat`) in years!

`deltat` the interval between two observations in the regulated time series. It is the inverse of `frequency`. If both `frequency` and `deltat` are provided, then `frequency` supersedes `deltat`

`datemin` this is mostly useful for converting "days" in "years" time-scales (`units="daystoyears"`). If the `x` vector contains: 1, 3, 6,... (day #1, day #3, day #6,... of the experiment), one can give here the exact date of the first observation, allowing to define a correct origin in the "years" time scale. Provide a string in a format compatible with `dateformat`. For instance, if day #1 is the 21th March 1998, give `datemin="03/21/1998"` with `dateformat="m/d/Y"`

`dateformat` indicate how `datemin` is formated. For instance: `"d/m/Y"`, or `m/d/Y"` (by default), see `daystoyears()` for more info on date formatting

`tol` the tolerance in the time-scale to determine if a measured value is used to approximate a regulated value. If `tol=0`, observations in each respective series must match exactly, otherwise observations in the regulated series are interpolated. `tol` must be a round fraction of `deltat` (deltat, deltat/2, deltat/3, etc...), and cannot be larger than it, otherwise, `tol` is automatically adjusted to the closest allowed value. By default, `tol=NULL`. This is equivalent to `tol=0`. Warning! In the particular case of `units="daystoyears"`, `tol` must be expressed in the original time-scale, that is "days", while `deltat` must be expressed in the fimal time-scale, that is "years"!

`tol.type` the type of adjustment to use for the time-tolerance: `"left"`, `"right"`, `"both"` (by default) or `"none"`. If `tol.type="left"`, corresponding `x` values are seeked in a window]xregul-tol, xregul]. If `tol.type="right"`, they are seeked in the window [xregul, xregul+tol[. If `tol.type="both"`, then they are seeked in the window]xregul-tol, xregul+tol]. If several observations are in this window, the closest one is used. Finally, if `tol.type="none"`, then *all* observations in the regulated time series are interpolated (even if exactly matching observations exist!)

`methods` the method(s) to use to regulate the time series. Currently, it can be: `"constant"`, `"linear"`, `"spline"` or `"area"` (or a unique abbreviation of them). If several time series are provided (`y` is a matrix or a data frame), it is possible to define methods individually for each series. For instance, `methods=c("l", "a", "s")` defines the "linear" method for the first series, the "area" method for the second one, the "spline" method for the third one,... and again the "linear" for the fourth, the "area" for the fifth one, etc. (recycling rule). By default, the "linear" method is selected for all series

rule — the rule to use for extrapolated values (observations in the final regular time series that are outside the range of observed values in the initial time series). With rule=1 (by default), these entries are not calculated and get NA; with rule=2, these entries are extrapolated (avoid using this option, or use with extreme care!!!)

f — parameter for the "constant" regulation method. Coefficient giving more weight to the observation at left (f=0, by default), to the observation at right (f=1), or give an intermediate weight to both of these observations (0 < f < 1) during the interpolation (see reglin())

periodic — parameter for the "spline" regulation method. Indicate if the time series should be considered as periodic (periodic=TRUE, first value must be equal to the last one). If this is the case, first and second derivates used to calculate spline segments around first and last observations use data in the other extreme of the series. In the other case (periodic=FALSE, by default), derivates for extremes observations are considered to be equal to zero

window — parameter for the "area" regulation method. Size of the window to consider (see regarea()). By default, the mean interval between observations in the initial irregular time series is used. Give the same value as for deltat for working with adjacent windows

split — other parameter for the "area" method. To optimise calculation time and to avoid to saturate memory, very long time series are splitted into smaller subunits (see regarea()). This is transparent for the user. The default value of split=100 should be rarely changed. Give a lower value if the program fails and reports a memory problem during calculation

specs — a specs.regul object returned by the function specs() applied to a regul object. Allows to collect parameterization of the regul() function and to apply them to another regulation

reg — A regul object as obtained after using the regul() function

series — the series to plot. By default, series=1, corresponding to the first (or possibly the unique) series in the regul object

col — (1) for plot(): the two colors to use to draw respectively the initial irregular series and the final regulated series. col=c(1,2) by default. (2) for hist(): the three colors to use to represent respectively the fist bar (exact coincidence), the middle bars (coincidence in a certain tolerance window) and the last bar (values always interpolated). By default, col=c(4,5,2)

lty — the style to use to draw lines for the initial series and the regulated series, respectively. The default style is used for both lines if this argument is not provided

plot.pts — if plot.pts=TRUE (by default) then points are also drawn for the regulated series (+). Those points that match observations in the initial irregular series, and are not interpolated, are further marked with a circle

leg — do we add a legend to the graph? By default, leg=FALSE, no legend is added

llab — the labels to use for the initial irregular and the final regulated series, respectively. By default, it is "initial" for the first one and the name of the regulation method used for the second one (see methods argument)

lpos	the position of the top-left corner of the legend box (x,y), in the graph coordinates
...	additional graph parameters
label	the character to use to mark points interactively selected on the graph. By default, label="#"
nclass	the number of classes to calculate in the histogram. This is indicative and this value is automatically adjusted to obtain a nicely-formatted histogram. By default, nclass=30
plotit	If plotit=TRUE then the histogram is plotted. Otherwise, it is only calculated

Details

Several irregular time series (for instance, contained in a data frame) can be treated at once. Specify a vector with "constant", "linear", "spline" or "area" for the argument methods to use a different regulation method for each series. See corresponding fonctions (regconst(), reglin(), regspline() and regarea()), respectively, for more details on these methods. Arguments can be saved in a specs object and reused for other similar regulation processes. Functions regul.screen() and regul.adj() are useful to chose best time interval in the computed regular time series. If you want to work on seasonal effects in the time series, you will better use a "years" time-scale (1 unit = 1 year), or convert into such a scale. If initial time unit is "days" (1 unit = 1 day), a conversion can be operated at the same time as the regulation by specifying units="daystoyears".

Value

An object of type 'regul' is returned. It has methods print(), summary(), plot(), lines(), identify(), hist(), extract() and specs().

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Lancaster, P. & K. Salkauskas, 1986. *Curve and surface fitting.* Academic Press, England, 280 pp.

Fox, W.T. & J.A. Brown, 1965. *The use of time-trend analysis for environmental interpretation of limestones.* J. Geol., 73:510-518.

Ibanez, F., 1991. *Treatment of the data deriving from the COST 647 project on coastal benthic ecology: The within-site analysis.* In: B. Keegan (ed). Space and Time Series Data Analysis in Coastal Benthic Ecology. Pp 5-43.

Ibanez, F. & J.C. Dauvin, 1988. *Long-term changes (1977-1987) on a muddy fine sand Abra alba - Melinna palmata population community from the Western English Channel.* J. Mar. Ecol. Prog. Ser., 49:65-81.

See Also

regul.screen, regul.adj, tseries, is.tseries, regconst, reglin, regspline, regarea, daystoyears

Examples

```
data(releve)
# The series in this data frame are very irregularly sampled in time:
releve$Day
length(releve$Day)
intervals <- releve$Day[2:61]-releve$Day[1:60]
intervals
range(intervals)
mean(intervals)
# The series must be regulated to be converted in a 'rts' or 'ts object
rel.reg <- regul(releve$Day, releve[3:8], xmin=9, n=63, deltat=21,
        tol=1.05, methods=c("s","c","l","a","s","a"), window=21)
rel.reg
plot(rel.reg, 5)
specs(rel.reg)
# Now we can extract one or several regular time series
melo.ts <- extract(rel.reg, series="Melosul")
is.tseries(melo.ts)

# One can convert time-scale from "days" to "years" during regulation
# This is most useful for analyzing seasonal cycles in a second step
melo.regy <- regul(releve$Day, releve$Melosul, xmin=6, n=87,
        units="daystoyears", frequency=24, tol=2.2, methods="linear",
        datemin="21/03/1989", dateformat="d/m/Y")
melo.regy
plot(melo.regy, main="Regulation of Melosul")
# In this case, we have only one series in 'melo.regy'
# We can use also 'tseries()' instead of 'extract()'
melo.tsy <- tseries(melo.regy)
is.tseries(melo.tsy)
```

---

regul.adj *Adjust regulation parameters*

Description

Calculate and plot an histogram of the distances between interpolated observations in a regulated time series and closest observations in the initial irregular time series. This allows to optimise the `tol` parameter

Usage

```
regul.adj(x, xmin=min(x), frequency=NULL, deltat, tol=deltat,
        tol.type="both", nclass=50, col=c(4, 5, 2), plotit=TRUE, ...)
```

Arguments

`x`	a vector with times corresponding to the observations in the irregular initial time series

xmin	the time corresponding to the first observation in the regular time series
frequency	the frequency of observations in the regular time series
deltat	the interval between two successive observations in the regular time series. This is the inverse of frequency. Only one of both parameters need to be given. If both are provided, frequency supersedes deltat
tol	the tolerance in the difference between two matching observations (in the original irregular series and in the regulated series). If tol=0 both values must be strictly identical; a higher value for tol allows some fuzzy matching. tol must be a round fraction of deltat and cannot be higher than it, otherwise, it is adjusted to the closest acceptable value. By default, tol=deltat
tol.type	the type of window to use for the time-tolerance: "left", "right", "both" (by default) or "none". If tol.type="left", corresponding x values are seeked in a window]xregul-tol, xregul]. If tol.type="right", they are seeked in the window [xregul, xregul+tol[. If tol.type="both", then they are seeked in the window]xregul-tol, xregul+tol]. If several observations are in this window, the closest one is used. Finally, if tol.type="none", then *all* observations in the regulated time series are interpolated (even if exactly matching observations exist!)
nclass	the number of classes to compute in the histogram. This is indicative, and will be adjusted by the algorithm to produce a nicely-formatted histogram. The default value is nclass=50. It is acceptable in many cases, but if the histogram is not correct, try a larger value
col	the three colors to use to represent respectively the fist bar (exact coincidence), the middle bars (coincidence in a certain tolerance window) and the last bar (values always interpolated). By default, col=c(4,5,2)
plotit	if plotit=TRUE then the histogram is plotted. Otherwise, it is only calculated
...	additional graph parameters for the histogram

Details

This function is complementary to regul.screen(). While the later look for the best combination of the number of observations, the interval between observations and the position of the first observation on the time-scale for the regular time series, regul.adj() look for the optimal value for tol, the tolerance window.

Value

A list with components:

params	the parameters used for the regular time-scale
match	the number of matching observations in the tolerance window
exact.match	the number of exact matching observations
match.counts	a vector with the number of matching observations for increasing values of tol

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

regul.screen, regul

Examples

```
# This example follows the example for regul.screen()
# where we determined that xmin=9, deltat=21, n=63, with tol=1.05
# is a good choice to regulate the irregular time series in 'releve'
data(releve)
regul.adj(releve$Day, xmin=9, deltat=21)
# The histogram indicates that it is not useful to increase tol
# more than 1.05, because few observations will be added
# except if we increase it to 5-7, but this value could be
# considered to be too large in comparison with deltat=22
# On the other hand, with tol <= 1, the number of matching
# observations will be almost divided by two!
```

---

regul.screen *Test various regulation parameters*

Description

Seek for the best combination of the number of observation, the interval between two successive observation and the position of the first observation in the regulated time series to match as much observations of the initial series as possible

Usage

```
regul.screen(x, weight=NULL, xmin=min(x), frequency=NULL, deltat,
        tol=deltat/5, tol.type="both")
```

Arguments

x	a vector with times corresponding to the observations in the irregular initial time series
weight	a vector of the same length as x, with the weight to give to each observation. A value of 0 indicates to ignore an observation. A value of 1 gives a normal weight to an observation. A higher value gives more importance to the corresponding observation. You can increase weight of observations around major peaks and pits, to make sure they are not lost in the regulated time series. If weight=NULL (by default), then a weight of 1 is used for all observations
xmin	a vector with all time values for the first observation in the regulated time series to be tested
frequency	a vector with all the frequencies to be screened
deltat	a vector with all time intervals to screen. deltat is the inverse of frequency. Only one of these two arguments is required. If both are provided, frequency supersedes deltat

`tol` it is possible to tolerate some differences in the time between two matching observations (in the original irregular series and in the regulated series). If `tol=0` both values must be strictly identical; a higher value allows some fuzzy matching. `tol` must be a round fraction of `deltat` and cannot be higher than it, otherwise, it is adjusted to the closest acceptable value. By default, `tol=deltat/5`

`tol.type` the type of window to use for the time-tolerance: `"left"`, `"right"`, `"both"` (by default) or `"none"`. If `tol.type="left"`, corresponding `x` values are seeked in a window]xregul-tol, xregul]. If `tol.type="right"`, they are seeked in the window [xregul, xregul+tol[. If `tol.type="both"`, then they are seeked in the window]xregul-tol, xregul+tol]. If several observations are in this window, the closest one is used. Finally, if `tol.type="none"`, then *all* observations in the regulated time series are interpolated (even if exactly matching observations exist!)

Details

Whatever the efficiency of the interpolation procedure used to regulate an irregular time series, a matching, non-interpolated observation is always better than an interpolated one! With very irregular time series, it is often difficult to decide which is the better regular time-scale in order to interpolate as less observations as possible. `regul.screen()` tests various combinations of number of observation, interval between two observations and position of the first observation and allows to choose the combination that best matches the original irregular time series. To choose also an optimal value for `tol`, use `regul.adj()` concurrently.

Value

A list containing:

`tol` a vector with the adjusted values of `tol` for the various values of `deltat`

`n` a table indicating the maximum value of `n` for all combinations of `deltat` and `xmin` to avoid any extrapolation

`nbr.match` a table indicating the number of matching observations (in the tolerance window) for all combinations of `deltat` and `xmin`

`nbr.exact.match` a table indicating the number of exactly matching observations (with a tolerance window equal to zero) for all combinations of `deltat` and `xmin`

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

`regul.adj`, `regul`

Examples

```
data(releve)
# This series is very irregular, and it is difficult
# to choose the best regular time-scale
releve$Day
length(releve$Day)
intervals <- releve$Day[2:61]-releve$Day[1:60]
intervals
range(intervals)
mean(intervals)
# A combination of xmin=1, deltat=22 and n=61 seems correct
# But is it the best one?
regul.screen(releve$Day, xmin=0:11, deltat=16:27, tol=1.05)
# Now we can tell that xmin=9, deltat=21, n=63, with tol=1.05
# is a much better choice!
```

---

releve *A data frame of six phytoplankton taxa followed in time at one station*

Description

The releve data frame has 61 rows and 8 columns. Several phytoplankton taxa were numbered in a single station from 1989 to 1992, but at irregular times.

Usage

```
data(releve)
```

Format

This data frame contains the following columns:

Day days number, first observation being day 1

Date strings indicating the date of the observations in "dd/mm/yyyy" format

Astegla the abundance of Asterionella glacialis

Chae the abundance of Chaetoceros sp

Dity the abundance of Ditylum sp

Gymn the abundance of Gymnodinium sp

Melosul the abundance of Melosira sulcata + Paralia sulcata

Navi the abundance of Navicula sp

Source

Belin, C. & B. Raffin, 1998. *Les espèces phytoplanctoniques toxiques et nuisibles sur le littoral français de 1984 à 1995, résultats du REPHY (réseau de surveillance du phytoplancton et des phycotoxines)*. Rapport IFREMER, RST.DEL/MP-AO-98-16. IFREMER, France.

specs — *Collect parameters ("specifications") from one object to use them in another analysis*

Description

'specs' is a generic function for reusing specifications included in an object and applying them in another similar analysis

Usage

```
specs(x, ...)
```

Arguments

x	An object that has "specs" data
...	Additional arguments (redefinition of one or several parameters)

Value

A 'specs' object that has the `print` method and that can be entered as an argument to functions using similar "specifications"

See Also

`regul`, `tsd`

stat.desc — *Descriptive statistics on a data frame or time series*

Description

Compute a table giving various descriptive statistics about the series in a data frame or in a single/multiple time series

Usage

```
stat.desc(x, basic=TRUE, desc=TRUE, norm=FALSE, p=0.95)
```

Arguments

x	a data frame or a time series
basic	do we have to return basic statistics (by default, it is TRUE)? These are: the number of values (nbr.val), the number of null values (nbr.null), the number of missing values (nbr.na), the minimal value (min), the maximal value (max), the range (range, that is, max-min) and the sum of all non-missing values (sum)
desc	do we have to return various descriptive statistics (by default, it is TRUE)? These are: the median (median), the mean (mean), the standard error on the mean (SE.mean), the confidence interval of the mean (CI.mean) at the p level, the variance (var), the standard deviation (std.dev) and the variation coefficient (coef.var) defined as the standard deviation divided by the mean
norm	do we have to return normal distribution statistics (by default, it is FALSE)? the skewness coefficient g1 (skewness), its significant criterium (skew.2SE, that is, g1/2.SEg1; if skew.2SE > 1, then skewness is significantly different than zero), kurtosis coefficient g2 (kurtosis), its significant criterium (kurt.2SE, same remark than for skew.2SE), the statistic of a Shapiro-Wilk test of normality (normtest.W) and its associated probability (normtest.p)
p	the probability level to use to calculate the confidence interval on the mean (CI.mean). By default, p=0.95

Value

a data frame with the various statistics in rows and with each column correponding to a variable in the data frame, or to a separate time series

Note

The Shapiro-Wilk test of normality is not available yet in Splus and it returns 'NA' in this environment. If you prefer to get separate statistics for various time intervals in your time series, use stat.slide(). If your data are fish or plankton sampled with a net, consider using the Pennington statistics (see stat.pen())

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

stat.slide, stat.pen

Examples

```
data(marbio)
stat.desc(marbio[,13:16], basic=TRUE, desc=TRUE, norm=TRUE, p=0.95)
```

`stat.pen` *Pennington statistics on a data frame or time series*

Description

Compute a table giving various descriptive statistics, including Pennington's estimators of the mean, the variance and the variance of the mean, about the series in a data frame or in a single/multiple time series

Usage

```
stat.pen(x, basic=FALSE, desc=FALSE)
```

Arguments

`x`	a data frame or a time series
`basic`	do we have to return also basic statistics (by default, it is FALSE)? These are: the number of values (nbr.val), the number of null values (nbr.null), the number of missing values (nbr.na), the minimal value (min), the maximal value (max), the range (range, that is, max-min) and the sum of all non-missing values (sum)
`desc`	do we have to return also various descriptive statistics (by default, it is FALSE)? These are: the median (median), the mean (mean), the standard error on the mean (SE.mean), the confidence interval of the mean (CI.mean) at the `p` level, the variance (var), the standard deviation (std.dev) and the variation coefficient (coef.var) defined as the standard deviation divided by the mean

Value

a data frame with the various statistics in rows and with each column correponding to a variable in the data frame, or to a separate time series

Note

If you prefer to get separate statistics for various time intervals in your time series, use `stat.slide()`. Various other descriptive statistics, including test of the normal distribution are also available in `stat.desc()`

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Aitchison, J., 1955. *On the distribution of a positive random variable having a discrete probability mass at the origin.* J. Amer. Stat. Ass., 50:901-908.

Pennington, M., 1983. *Efficient estimations of abundance for fish and plankton surveys.* Biometrics, 39:281-286.

See Also

stat.slide, stat.desc

Examples

```
data(marbio)
stat.pen(marbio[,c(4, 14:16)], basic=TRUE, desc=TRUE)
```

---

stat.slide *Sliding statistics*

Description

Statistical parameters are not constant along a time series: mean or variance can vary each year, or during particular intervals (radical or smooth changes due to a pollution, a very cold winter, a shift in the system behaviour, etc. Sliding statistics offer the potential to describe series on successive blocs defined along the space-time axis

Usage

```
stat.slide(x, y, xcut=NULL, xmin=min(x), n=NULL, frequency=NULL,
        deltat=1/frequency, basic=FALSE, desc=FALSE, norm=FALSE,
        pen=FALSE, p=0.95)
## S3 method for class 'stat.slide':
plot(statsl, stat="mean", col=c(1, 2), lty=c(par("lty"), par("lty")),
        leg=FALSE, llab=c("series", stat), lpos=c(1.5, 10), ...)
## S3 method for class 'stat.slide':
lines(statsl, stat="mean", col=3, lty=1, ...)
```

Arguments

x	a vector with time data
y	a vector with observation at corresponding times
xcut	a vector with the position in time of the breaks between successive blocs. xcut=NULL by default. In the later case, a vector with equally spaced blocs is constructed using xmin, n and frequency or deltat. If a value is provided for xcut, then it supersedes all these other parameters
xmin	the minimal value in the time-scale to use for constructing a vector of equally spaced breaks
n	the number of breaks to use
frequency	the frequency of the breaks in the time-scale
deltat	the bloc interval touse for constructing an equally-spaced break vector. deltat is 1/frequency

basic	do we have to return basic statistics (by default, it is FALSE)? These are: the number of values (nbr.val), the number of null values (nbr.null), the number of missing values (nbr.na), the minimal value (min), the maximal value (max), the range (range, that is, max-min) and the sum of all non-missing values (sum)
desc	do we have to return descriptive statistics (by default, it is FALSE)? These are: the median (median), the mean (mean), the standard error on the mean (SE.mean), the confidence interval of the mean (CI.mean) at the p level, the variance (var), the standard deviation (std.dev) and the variation coefficient (coef.var) defined as the standard deviation divided by the mean
norm	do we have to return normal distribution statistics (by default, it is FALSE)? the skewness coefficient g1 (skewness), its significant criterium (skew.2SE, that is, g1/2.SEg1; if skew.2SE > 1, then skewness is significantly different than zero), kurtosis coefficient g2 (kurtosis), its significant criterium (kurt.2SE, same remark than for skew.2SE), the statistic of a Shapiro-Wilk test of normality (normtest.W) and its associated probability (normtest.p)
pen	do we have to return Pennington and other associated statistics (by default, it is FALSE)? pos.median, pos.mean, pos.var, pos.std.dev, respectively the median, the mean, the standard deviation and the variance, considering only non-null values; geo.mean, the geometric mean that is, the exponential of the mean of the logarithm of the observations, excluding null values. pen.mean, pen.var, pen.std.dev, pen.mean.var, respectively the mean, the variance, the standard deviation and the variance of the mean after Pennington's estimators (see `pennington()`)
p	the probability level to use to calculate the confidence interval on the mean (CI.mean). By default, `p=0.95`
statsl	a 'stat.slide' object, as returned by the function `stat.slide()`
stat	the statistic to plot on the graph. You can use "min", "max", "median", "mean" (by default), "pos.median", "pos.mean", "geo.mean" and "pen.mean". The other statistics cannot be superposed on the graph of the series in the current version of the function
col	the colors to use to plot the initial series and the statistics, respectively. By default, `col=c(1,2)`
lty	the style to use to draw the original series and the statistics. The default style is used if this argument is not provided
leg	if `leg=TRUE`, a legend box is drawn on the graph
llab	the labels to use for the legend. By default, it is "series" and the corresponding statistics provided in `stat`, respectively
lpos	the position of the top-left corner (x,y) of the legend box in the graph coordinates. By default `lpos=c(1.5,10)`
...	additional graph parameters

Details

Available statistics are the same as for `stat.desc()` and `stat.pen()`. The Shapiro-Wilk test of normality is not available yet in Splus and it returns 'NA' in this environment. If not a priori known, successive blocs can be identified using either `local.trend()` or `decmedian()` (see respective functions for further details)

Value

An object of type 'stat.slide' is returned. It has methods `print()`, `plot()` and `lines()`.

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

`stat.desc`, `stat.pen`, `pennington`, `local.trend`, `decmedian`

Examples

```
data(marbio)
# Sliding statistics with fixed-length blocs
statsl <- stat.slide(1:68, marbio[, "ClausocalanusA"], xmin=0, n=7, deltat=10)
statsl
plot(statsl, stat="mean", leg=TRUE, lpos=c(55, 2500), xlab="Station",
        ylab="ClausocalanusA")

# More information on the series, with predefined blocs
statsl2 <- stat.slide(1:68, marbio[, "ClausocalanusA"],
        xcut=c(0, 17, 25, 30, 41, 46, 70), basic=TRUE, desc=TRUE, norm=TRUE,
        pen=TRUE, p=0.95)
statsl2
plot(statsl2, stat="median", xlab="Stations", ylab="Counts",
        main="Clausocalanus A")              # Median
lines(statsl2, stat="min")                   # Minimum
lines(statsl2, stat="max")                   # Maximum
lines(c(17, 17), c(-50, 2600), col=4, lty=2) # Cuts
lines(c(25, 25), c(-50, 2600), col=4, lty=2)
lines(c(30, 30), c(-50, 2600), col=4, lty=2)
lines(c(41, 41), c(-50, 2600), col=4, lty=2)
lines(c(46, 46), c(-50, 2600), col=4, lty=2)
text(c(8.5, 21, 27.5, 35, 43.5, 57), 2300, labels=c("Peripheral Zone", "D1",
        "C", "Front", "D2", "Central Zone")) # Labels
legend(0, 1900, c("series", "median", "range"), col=1:3, lty=1)
# Get cuts back from the object
statsl2$xcut
```

`trend.test` *Test if an increasing or decreasing trend exists in a time series*

Description

Test if the series has an increasing or decreasing trend, using a non-parametric Spearman test between the observations and time

Usage

```
trend.test(tseries, R=1)
```

Arguments

`tseries`	a univariate or multivariate time series (a 'rts' object in Splus or a 'ts' object in R)
`R`	The number of time the series is/are resampled for a bootstrap test. If `R1` (by default), an usual Spearman test is performed. If `R` > 1 then a bootstrap test is run

Value

A 'htest' object if `R=1`, a 'boot' object with an added `boot$p.value` item otherwise

Note

In both cases (normal test with `R=1` and bootstrap test), the p-value can be obtained from `obj$p.value` (see examples)

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Siegel, S. & N.J. Castellan, 1988. *Non-parametric statistics.* McGraw-Hill, New York. 399 pp.

See Also

`local.trend`

Examples

```
data(marbio)
trend.test(marbio[, 8])
# Run a bootstrap test on the same series
marbio8.trend.test <- trend.test(marbio[, 8], R=99)
# R=999 is a better value... but it is very slow!
marbio8.trend.test
plot(marbio8.trend.test)
marbio8.trend.test$p.value
```

`tsd` *Decomposition of one or several regular time series using various methods*

Description

Use a decomposition method to split the series into two or more components. Decomposition methods are either series filtering/smoothing (difference, average, median, evf), deseasoning (loess) or model-based decomposition (reg, i.e., regression).

Usage

```
tsd(x, specs=NULL, method="loess", type="additive", ...)
## S3 method for class 'tsd':
summary(tsdobj)
## S3 method for class 'tsd':
plot(tsdobj, series=1, stack=TRUE, resid=TRUE,
        labels, leg=TRUE, lpos=c(0, 0), ...)
## S3 method for class 'tsd':
extract(tsdobj, n, series=NULL, components=NULL)
## S3 method for class 'tsd':
specs(tsdobj)
```

Arguments

`x`	an univariate or multivariate regular time series ('rts' in Splus or 'ts' in R) to be decomposed
`specs`	specifications are collected from a 'tsd' object, using the `specs` method. This allows for reusing parameters issued from a previous similar analysis
`method`	the method to use to decompose the time series. Currently, possible values are: `"diff"`, `"average"`, `"median"`, `"evf"`, `"reg"`, `"loess"` (by default) or `"census"`. The corresponding function `decXXXX()` is applied to each of the series in `x`
`type`	the type of model to use: either `"additive"` (by default) or `"multiplicative"`. In the additive model, all components must be added to reconstruct the initial series. In the multiplicative model, they must be multiplied (one components has the same unit as the original series, and the other ones are dimensionless multiplicative factors)
`...`	(1) for `tsd()`: further arguments to pass to the corresponding `decXXXX()` function. (2) for `plot()`: further graphical arguments
`tsdobj`	a 'tsd' object as returned by the function `tsd()`, or any of the `decXXXX()` functions
`series`	(1) for `plot()`: the series to plot. By default, `series=1`, the first (or possibly unique) series in the 'tsd' object is plotted. (2) for `extract`: the name or the index of the series to extract. If `series` is provided, then `n` is ignored. By

default, `series=NULL`. It is also possible to use negative indices. In this case, all series are extracted, except those ones

`stack` graphs of each component are either stacked (`stack=TRUE`, by default), or superposed on the same graph `stack=FALSE`

`resid` do we have to plot also the "residuals" components (`resid=TRUE`, by default) or not? Usually, in a stacked graph, you would like to plot the residuals, while in a superposed graph, you would not

`labels` the labels to use for all y-axes in a stacked graph, or in the legend for a superposed graph. By default, the names of the components ("trend", "seasonal", "deseasoned", "filtered", "residuals", ...) are used

`leg` only used when `stack=FALSE`. Do we plot a legend (`leg=TRUE` or not?

`lpos` position of the upper-left corner of the legend box in the graph coordinates (x,y). By default, `leg=c(0,0)`

`n` the number of series to extract (from series 1 to series n). By default, n equals the number of series in the 'tsd' object. If both `series` and `components` arguments are NULL, all series and components are extracted and this method has exactly the same effect as `tseries`

`components` the names or indices of the components to extract. If `components=NULL` (by default), then all components of the selected series are extracted. It is also possible to specify negative indices. In this case, all components are extracted, except those ones

Details

To eliminate trend from a series, use "diff" or use "loess" with `trend=TRUE`. If you know the shape of the trend (linear, exponential, periodic, etc.), you can also use it with the "reg" (regression) method. To eliminate or extract seasonal components, you can use "loess" if the seasonal component is additive, or "census" if it is multiplicative. You can also use "average" with argument `order="periodic"` and with either an additive or a multiplicative model, although the later method is often less powerful than "loess" or "census". If you want to extract a seasonal cycle with a given shape (for instance, a sinusoid), use the "reg" method with a fitted sinusoidal equation. If you want to identify levels in the series, use the "median" method. To smooth the series, you can use preferably the "evf" (eigenvector filtering), or the "average" methods, but you can also use "median". To extract most important components from the series (no matter if they are cycles -seasonal or not-, or long-term trends), you should use the "evf" method. For more information on each of these methods, see online help of the corresponding `decXXXX()` functions.

Value

An object of type 'tsd' is returned. It has methods `print()`, `summary()`, `plot()`, `extract()` and `specs()`.

Note

If you have to decompose a single time series, you could also use the corresponding `decXXXX()` function directly. In the case of a multivariate regular time series, `tsd()` is more convenient because it decompose all times series of a set at once!

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Kendall, M., 1976. *Time-series.* Charles Griffin & Co Ltd. 197 pp.

Laloire, J.C., 1972. *Méthodes du traitement des chroniques.* Dunod, Paris, 194 pp.

Legendre, L. & P. Legendre, 1984. *Ecologie numérique. Tome 2: La structure des données écologiques.* Masson, Paris. 335 pp.

Malinvaud, E., 1978. *Méthodes statistiques de l'économétrie.* Dunod, Paris. 846 pp.

Philips, L. & R. Blomme, 1973. *Analyse chronologique.* Université Catholique de Louvain. Vander ed. 339 pp.

See Also

tseries, decdiff, decaverage, decmedian, decevf, decreg, decloess, deccensus

Examples

```
data(releve)
# Regulate the series and extract them as a time series object
rel.regy <- regul(releve$Day, releve[3:8], xmin=6, n=87, units="daystoyears",
        frequency=24, tol=2.2, methods="linear", datemin="21/03/1989",
        dateformat="d/m/Y")
rel.ts <- tseries(rel.regy)

# Decompose all series in the set with the "loess" method
rel.dec <- tsd(rel.ts, method="loess", s.window=13, trend=FALSE)
rel.dec
plot(rel.dec, series=5, col=1:3)    # An plot series 5

# Extract "deseasoned" components
rel.des <- extract(rel.dec, series=3:6, components="deseasoned")
rel.des[1:10,]

# Further decompose these components with a moving average
rel.des.dec <- tsd(rel.des, method="average", order=2, times=10)
plot(rel.des.dec, series=3, col=c(2, 4, 6))
# In this case, a superposed graph is more appropriate:
plot(rel.des.dec, series=3, col=c(2,4), stack=FALSE, resid=FALSE,
        labels=c("without season cycle", "trend"), lpos=c(0, 55000))
# Extract residuals from the latter decomposition
rel.res2 <- extract(rel.des.dec, components="residuals")
```

`tseries` *Convert a 'regul' or a 'tsd' object into a time series*

Description

Regulated series contained in a 'regul' object or components issued from a time series decomposition with 'tsd' are extracted from their respective object and converted into uni- or multivariate regular time series ('rts' objects in Splus and 'ts' objects in R)

Usage

```
tseries(x)
```

Arguments

`x`	A 'regul' or 'tsd' object

Value

an uni- or multivariate regular time series

Note

To extract some of the time series contained in the 'regul' or 'tsd' objects, use the `extract()` method

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩), Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩)

See Also

`is.tseries`, `regul`, `tsd`

Examples

```
data(releve)
rel.regy <- regul(releve$Day, releve[3:8], xmin=6, n=87, units="daystoyears",
        frequency=24, tol=2.2, methods="linear", datemin="21/03/1989",
        dateformat="d/m/Y")
# This object is not a time series
is.tseries(rel.regy)     # FALSE
# Extract all time series contained in the 'regul' object
rel.ts <- tseries(rel.regy)
# Now this is a time series
is.tseries(rel.ts)       # TRUE
```

`turnogram` *Calculate and plot a turnogram for a regular time series*

Description

The turnogram is the variation of a monotony index with the observation scale (the number of data per time unit). A monotony index indicates if the series has more or less erratic variations than a pure random succession of independent observations. Since a time series almost always has autocorrelation, it is expected to be more monotonous than a purely random series. The monotony index is a way to quantify the density of information beared by a time series. The turnogram determines at which observation scale this density of information is maximum. It is also the scale that optimize the sampling effort (best compromise between less samples versus more information).

Usage

```
turnogram(series, intervals=c(1, length(series)/5), step=1, complete=FALSE,
        two.tailed=TRUE, FUN=mean, plotit=TRUE, level=0.05, lhorz=TRUE,
        lvert=FALSE, xlog=TRUE)
## S3 method for class 'turnogram':
summary(turno)
## S3 method for class 'turnogram':
plot(turno, level=0.05, lhorz=TRUE, lvert=TRUE, lcol=2,
        llty=2, xlog=TRUE, ...)
## S3 method for class 'turnogram':
identify(turno, lvert=TRUE, ...)
## S3 method for class 'turnogram':
extract(turno, n, level=turno$level, FUN=turno$fun, drop=0)
```

Arguments

`series`	a single regular time series ('rts' object in Splus or 'ts' object in R)
`intervals`	the range (mini, maxi) of the intervals to calculate, i.e., to take one obervation every 'interval' one. By default, `intervals` ranges from 1 to the fifth of the total number of observations
`step`	the increment used for the intervals. By defaults `step=1`. To limit calculation or for a first screenning with a large range in the intervals, use a higher value for `step`
`complete`	if `complete=TRUE`, a complete turnogram is calculated, showing mean, minimal and maximal curves. If it is FALSE (by default), only a simple turnogram always starting from the first observation is calculated
`two.tailed`	if `two.tailed=TRUE` (by default), the monotony index is tested with a bilateral test, otherwise, a left-sided test is used
`FUN`	a function to apply to aggregate data in the intervals. It is a function of the type `FUN(x, na.rm, ...)`. The most used function is `mean()` (by default), but it is also possible to keep only the first value with `first()`, the last value

with `last()`, the median or the sum of values in the interval. The later function is useful for cumulative observations, like pluviometry. It should be noted that the turnograms with `FUN=mean` and with `FUN=sum` are the same, but that extraction of final series are different for levels > 1

`plotit` if `plotit=TRUE` (by default), the graph of the turnogram is also plotted

`level` the significant level to draw on the graph. By default `level=0.05`, corresponding to a test with P = 5%

`lhorz` if `lhorz=TRUE` (by default) then one (left-sided test), or two (two-sided test) horizontal lines are drawn on the graph, indicating the significant level of the test given by the argument `level`. Any point above the single line, or outside the interval defined by the two lines is significant

`lvert` if `lvert=TRUE` (by default, except for `turnogram()` function), a vertical line is drawn, indicating the time interval that corresponds to the maximum information and it is also the automatic level of extraction unless this value is changed

`lcol` the color to use to draw supplemental lines: the horizontal line indicating where the test is significant (if `lhorz=TRUE`) and the vertical line indicating the extraction level (if `lvert=TRUE`). By default, color 2 is used

`llty` the style for the supplemental lines. By default, style 2 is used (dashed lines)

`xlog` if `xlog=TRUE` (by default), then the x-axis is expressed in logarithms. Otherwise, a linear scale is used

`...` additional optional graphic arguments

`turno` a 'turnogram' object as returned by the function `turnogram()`

`n` the number of observations to take into account in the initial series. Use `n=NULL` (by default) to use all observations of the series

`drop` the number of observations to drop at the beginning of the series before proceeding with the aggregation of the data for the extracted series. By default, `drop=0`: no observations are dropped

Details

The turnogram is a generalisation of the information theory (see `turnpoints()`). If a series has a lot of erratic peaks and pits that alternate with a high frequency, it is more difficult to interpret than a more monotonous series. These erratic fluctuations can be eliminated by changing the scale of observation (keeping one observation every two, three, four,... from the original series). The turnogram resample the original series this way, and calculate a monotony index for each resampled subseries. This monotony index quantifies the number of peaks and pits presents in the series, compared to the total number of observations. The Gleissberg distribution (see `pgleissberg()`) indicates the probability to have such a number of extrema in a series given it is purely random. It is possible to test monotony indices: is it a random series or not (two-sided test), or is more monotonous than a random series (left-sided test) thanks to a Chi-2 test proposed by Wallis & Moore (1941).

There are various turnograms depending on the way the observations are aggregated inside each time interval. For instance, if one consider one observation every three from the original series, each group of three observations can be aggregated in several different ways. One can take the

mean of the three observation, or the median value, or the sum,... One can also decide not to aggregate observations, but to drop some of them. Hence, one can take only the first or the last observation of the group. All these options can be choosen by defining the argument `FUN=...`. A simple turnogram correspond to the change of the monotony index with the scale of observation, stating always from the first observation. One could also decide to start from the second, or the third observation for an aggregation of the observations three by three... and result could be somewhat different. A complete turnogram investigates all possibles combinations (observation scale versus starting point for the aggregation) and trace the maximal, minimal and mean curves for the change of the monotony index. It is thus more informative than the simple turnogram. However, it takes much more time to compute.

The most obvious use of the turnogram is for the pretreatment of continuously sampled data. It helps in deciding which is the optimal sampling interval for the series to bear as most information as possible while keeping the dataset as small as possible. It is also interesting to compare the turnogram with other functions like the variogram (see `vario()`) or the spectrogram (see `spectrum()`).

Value

An object of type 'turnogram' is returned. It has methods `print()`, `summary()`, `plot()`, `identify()` and `extract()`.

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Dallot, S. & M. Etienne, 1990. *Une méthode non paramétrique d'analyse des series en océanographie biologique: les tournogrammes.* Biométrie et océanographie - Société de biométrie, 6, Lille, 26-28 mai 1986. IFREMER, Actes de colloques, 10:13-31.

Johnson, N.L. & Kotz, S., 1969. *Discrete distributions.* J. Wiley & sons, New York, 328 pp.

Kendall, M.G., 1976. *Time-series, 2nd ed.* Charles Griffin & co, London.

Wallis, W.A. & G.H. Moore, 1941. *A significance test for time series.* National Bureau of Economic Research, tech. paper n°1.

See Also

`pgleissberg`, `turnpoints`, `first`, `last`, `vario`, `spectrum`

Examples

```
data(bnr)
# Let's transform series 4 into a time series (supposing it is regular)
bnr4 <- as.ts(bnr[, 4])
plot(bnr4, type="l", main="bnr4: raw data", xlab="Time")
# A simple turnogram is calculated
bnr4.turno <- turnogram(bnr4)
summary(bnr4.turno)
# A complete turnogram confirms that "level=3" is a good value:
turnogram(bnr4, complete=TRUE)
```

```
# Data with maximum info. are extracted (thus taking 1 every 3 observations)
bnr4.interv3 <- extract(bnr4.turno)
plot(bnr4, type="l", lty=2, xlab="Time")
lines(bnr4.interv3, col=2)
title("Original bnr4 (dotted) versus max. info. curve (plain)")
# Choose another level (for instance, 6) and extract the corresponding series
bnr4.turno$level <- 6
bnr4.interv6 <- extract(bnr4.turno)
# plot both extracted series on top of the original one
plot(bnr4, type="l", lty=2, xlab="Time")
lines(bnr4.interv3, col=2)
lines(bnr4.interv6, col=3)
legend(70, 580, c("original", "interval=3", "interval=6"), col=1:3, lty=c(2, 1, 1))
# It is hard to tell on the graph which series contains more information
# The turnogram shows us that it is the "interval=3" one!
```

`turnpoints` *Analyze turning points (peaks or pits)*

Description

Determine the number and the position of extrema (turning points, either peaks or pits) in a regular time series. Calculate the quantity of information associated to the observations in this series, according to Kendall's information theory

Usage

```
turnpoints(x)
## S3 method for class 'turnpoints':
summary(turnp)
## S3 method for class 'turnpoints':
plot(turnp, level=0.05, lhorz=TRUE, lcol=2, llty=2, ...)
## S3 method for class 'turnpoints':
lines(turnp, max=TRUE, min=TRUE, median=TRUE,
        col=c(4, 4, 2), lty=c(2, 2, 1), ...)
## S3 method for class 'turnpoints':
extract(turnp, n, no.tp=0, peak=1, pit=-1)
```

Arguments

`x`	a vector or a time series
`turnp`	a 'turnpoints' object, as returned by the function `turnpoints()`
`level`	the significant level to draw on the graph if `lhorz=TRUE`. By default, `level=0.05`, which corresponds to a 5% p-value for the test
`lhorz`	if `lhorz=TRUE` (by default), an horizontal line indicating significant level is drawn on the graph
`lcol`	the color to use to draw the significant level line, by default, color 2 is used

`llty`	the style to use for the significant level line. By default, style 2 is used (dashed line)
`...`	Additional graph parameters
`max`	do we plot the maximum envelope line (by default, yes)
`min`	do we plot the minimum envelope line (by default, yes)
`median`	do we plot the median line inside the envelope (by default, yes)
`col`	a vector of three values for the color of the max, min, median lines, respectively. By default `col=c(4,4,2)`
`lty`	a vector of three values for the style of the max, min, median lines, respectively. By default `lty=c(2,2,1)`, that is: dashed, dashed and plain lines
`n`	the number of points to extract. By default `n=length(turnp)`, all points are extracted
`no.tp`	extract gives a vector representing the position of extrema in the original series. `no.tp` represents the code to use for points that are not an extremum, by default '0'
`peak`	the code to use to flag a peak, by default '1'
`pit`	the code to use to flag a pit, by default '-1'

Details

This function tests if the time series is purely random or not. Kendall (1976) proposed a series of tests for this. Moreover, graphical methods using the position of the turning points to draw automatically envelopes around the data are implemented, and also the drawing of median points between these envelopes.

With a purely random time series, one expect to find, on average, a turning point (peak or pit that is, an observation that is preceeded and followed by, respectively, lower or higher observations) every 1.5 observation. Given it is impossible to determine if first and last observation are turning point, it gives:

$$E(p) = 2/3 * (n - 2)$$

with p, the number of observed turning points and n the number of observations. The variance of p is:

$$var(p) = (16 * n - 29)/90$$

Ibanez (1982) demonstrated that P(t), the probability to observe a turning point at time t is:

$$P(t) = 2 * (1/n(t-1)! * (n-1)!)$$

where P is the probability to observe a turning point at time t under the null hypothesis that the time series is purely random, and thus, the distribution of turning points follows a normal distribution.

The quantity of information I associated with this probability is:

$$I = -log2P(t)$$

It can be interpreted as follows. If I is larger, there are less turning points than expected in a purely random series. There are, thus, longer sequence of increasing or decreasing values along the time scale. This is considered to be more informative.

As you can easily imagine, from this point on, it is straightforward to construct a test to determine if the series is random (regarding the distribution of the turning points), more or less monotonic (more or less turning points than expected).

Value

An object of type 'turnpoints' is returned. It has methods `print()`, `summary()`, `plot()`, `lines()` and `extract()`. Regarding your specific question, 'info' is the quantity of information I associated with the turning points:

`data`	The dataset to which the calculation is done
`n`	The number of observations
`points`	The value of the points in the series, after elimination of ex-aequos
`pos`	The position of the points on the time scale in the series (including ex-aequos)
`exaequos`	Location of exaequos (1), or not (0)
`nturns`	Total number of tunring points in the whole time series
`firstispeak`	Is the first turning point a peak (`TRUE`), or not (`FALSE`)
`peaks`	Logical vector. Location of the peaks in the time series without ex-aequos
`pits`	Logical vector. Location of the pits in the time series without ex-aequos
`tppos`	Position of the turning points in the initial series (with ex-aequos)
`proba`	Probability to find a turning point at this location (see details)
`info`	Quantity of information associated with this point (see details)

WARNING

the `lines()` method should be used to draw lines on the graph of the *original* dataset (`plot(data, type="l")` for instance), *not* on the graph of turning points (`plot(turnp)`)!

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

Ibanez, F., 1982. *Sur une nouvelle application de la theorie de l'information à la description des series chronologiques planctoniques.* J. Exp. Mar. Biol. Ecol., 4:619-632

Kendall, M.G., 1976. *Time-series, 2nd ed.* Charles Griffin & Co, London.

See Also

turnogram, stat.slide

Examples

```
data(marbio)
plot(marbio[, "Nauplii"], type="l")
# Calculate turning points for this series
Nauplii.tp <- turnpoints(marbio[, "Nauplii"])
summary(Nauplii.tp)
plot(Nauplii.tp)
# Add envelope and median line to original data
plot(marbio[, "Nauplii"], type="l")
lines(Nauplii.tp)
# Note that lines() applies to the graph of original dataset!!!
title("Raw data, envelope maxi., mini. and median line")
```

---

vario *Compute and plot a semi-variogram*

Description

Compute a classical semi-variogram for a single regular time series

Usage

```
vario(x, max.dist=length(x)/3, plotit=TRUE, vario.data=NULL)
```

Arguments

x	a vector or an univariate time series
max.dist	the maximum distance to calculate. By default, it is the third of the number of observations
plotit	If plotit=TRUE then the graph of the semi-variogram is plotted
vario.data	data coming from a previous call to vario(). Call the function again with these data to plot the corresponding graph

Value

A data frame containing distance and semi-variogram values

Author(s)

Frédéric Ibanez (⟨ibanez@obs-vlfr.fr⟩), Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

References

David, M., 1977. *Developments in geomathematics. Tome 2: Geostatistical or reserve estimation.* Elsevier Scientific, Amsterdam. 364 pp.

Delhomme, J.P., 1978. *Applications de la théorie des variables régionalisées dans les sciences de l'eau.* Bull. BRGM, section 3 n°4:341-375.

Matheron, G., 1971. *La théorie des variables régionalisées et ses applications.* Cahiers du Centre de Morphologie Mathématique de Fontainebleau. Fasc. 5 ENSMP, Paris. 212 pp.

See Also

disto

Examples

```
data(bnr)
vario(bnr[, 4])
```

Index