Package 'wql'

August 6, 2022

Type Package

Title Exploring Water Quality Monitoring Data

Version 1.0.0

Maintainer Jemma Stachelek <jemma.stachelek@gmail.com>

URL https://github.com/jsta/wql

BugReports https://github.com/jsta/wql/issues

Description Functions to assist in the processing and exploration of data from environmental monitoring programs. The package name stands for ``water quality'' and reflects the original focus on time series data for physical and chemical properties of water, as well as the biota. Intended for programs that sample approximately monthly, quarterly or annually at discrete stations, a feature of many legacy data sets. Most of the functions should be useful for analysis of similar-frequency time series regardless of the subject matter.

Depends R (>= 3.0.0)

Imports graphics, grDevices, methods, stats, ggplot2 (>= 1.0), reshape2, zoo

LazyData yes

License GPL-2

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.2.0

Suggests knitr, rmarkdown

R topics documented:

wql-package	*c("\Sexpr[results=rd,stage=build]tools:::Rd_package_title(\"#1\")", "wql")Exploring Water Quality Monitoring Data*

Description

c("\Sexpr[results=rd,stage=build]tools:::Rd_package_description(\"#1\")", "wql")Functions to assist in the processing and exploration of data from environmental monitoring programs. The package name stands for "water quality" and reflects the original focus on time series data for physical and chemical properties of water, as well as the biota. Intended for programs that sample approximately monthly, quarterly or annually at discrete stations, a feature of many legacy data sets. Most of the functions should be useful for analysis of similar-frequency time series regardless of the subject matter.

Details

The main purpose of **wql** is to explore seasonal time series through plots and nonparametric trend tests. It was created originally to examine water quality data sets (hence, "wql") but is suitable as a more general purpose set of tools for looking at annual or seasonal time series.

One of the more tedious tasks in exploring environmental data sets is creating usable time series from the original complex data sets, especially when you want many series at will that group data in different ways. So **wql** also provides a way of transforming data sets to a common format that then allows a diversity of time series to be created quickly. A few functions are specific to the fields of limnology and oceanography.

The plots are designed for easy use, not for publication-quality graphs. Nonetheless, extensive customization is possible by passing options through `...{}`, adding annotations in the case of base graphics, and adding layers in the case of **ggplot2** objects.

Two functions are used mainly for preparing the times series:

- a function that transforms incoming data to a common data structure in the form of the `WqData` class
- a function that easily prepares time series objects from this class

The `WqData` class can be easily adapted to non-aquatic data. Obviously, the `depth` field can be used for elevation in atmospheric studies. But more generally, the `site` and `depth` fields can be used for many two-way classifications and don't need to refer to spatial location.

Some of the time series functions include:

- a variety of plots to examine changes in seasonal patterns
- nonparametric trend tests
- time series interpolation and related manipulations
- a simple decomposition of a series into different time scales
- phenological analyses
- the use of empirical orthogonal functions to detect multiple independent mechanisms underlying temporal change

A few functions are specialized for the aquatic sciences:

- converting between oxygen concentrations and percent saturation
- converting between salinity and conductivity

The capabilities of **wql** are more fully explained in the accompanying vignette: "wql: Exploring environmental monitoring data".

Author(s)

c("\Sexpr[results=rd,stage=build]tools:::Rd_package_author(\"#1\")", "wql")NA

Maintainer: c("\Sexpr[results=rd,stage=build]tools:::Rd_package_maintainer(\"#1\")", "wql")Jemma Stachelek <jemma.stachelek@gmail.com>

`date2decyear` *date2decyear*

Description

date2decyear

Usage

```
date2decyear(w)
```

Arguments

`w`	date

Author(s)

Alan Jassby, James Cloern

`DateTime-class` *Class "DateTime"*

Description

A class union of `"Date"` and `"POSIXct"` classes.

Objects from the Class

A virtual Class: No objects may be created from it.

See Also

`WqData-class`

Examples

```
showClass("DateTime")
```

`decompTs` *Decompose a time series*

Description

The function decomposes a time series into a long-term mean, annual, seasonal and "events" component. The decomposition can be multiplicative or additive, and based on median or mean centering.

Usage

```
decompTs(
  x,
  event = TRUE,
  type = c("mult", "add"),
  center = c("median", "mean")
)
```

Arguments

`x`	a monthly time series vector
`event`	whether or not an "events" component should be determined
`type`	the type of decomposition, either multiplicative ("mult") or additive ("add")
`center`	the method of centering, either median or mean

Details

The rationale for this simple approach to decomposing a time series, with examples of its application, is given by Cloern and Jassby (2010). It is motivated by the observation that many important events for estuaries (e.g., persistent dry periods, species invasions) start or stop suddenly. Smoothing to extract the annualized term, which can disguise the timing of these events and make analysis of them unnecessarily difficult, is not used.

A multiplicative decomposition will typically be useful for a biological community- or population-related variable (e.g., chlorophyll-a) that experiences exponential changes in time and is approximately lognormal, whereas an additive decomposition is more suitable for a normal variable. The default centering method is the median, especially appropriate for series that have large, infrequent events.

If `event = TRUE`, the seasonal component represents a recurring monthly pattern and the events component a residual series. Otherwise, the seasonal component becomes the residual series. The latter is appropriate when seasonal patterns change systematically over time. You can use `plotSeason` and `seasonTrend` to investigate the way seasonality changes.

Value

A monthly time series matrix with the following individual time series:

`original`	original time series
`annual`	annual mean series
`seasonal`	repeating seasonal component
`events`	optionally, the residual or "events" series

Author(s)

Alan Jassby, James Cloern

References

Cloern, J.E. and Jassby, A.D. (2010) Patterns and scales of phytoplankton variability in estuarine-coastal ecosystems. *Estuaries and Coasts* **33,** 230–241.

See Also

`plotSeason`, `seasonTrend`

Examples

```
# Apply the function to a single series (Station 27) and plot it:
y <- decompTs(sfbayChla[, 's27'])
y
plot(y, nc=1, main="")
```

`decyear2date` *decyear2date*

Description

decyear2date

Usage

```
decyear2date(x)
```

Arguments

`x`	date

Author(s)

Alan Jassby, James Cloern

`ec2pss` *Convert conductivity to salinity*

Description

Electrical conductivity data are converted to salinity using the Practical Salinity Scale and an extension for salinities below 2.

Usage

```
ec2pss(ec, t, p = 0)
```

Arguments

`ec`	conductivity, mS/cm
`t`	temperature, Celsius
`p`	gauge pressure, decibar

Details

`ec2pss` converts electrical conductivity data to salinity using the Practical Salinity Scale 1978 in the range of 2-42 (Fofonoff and Millard 1983). Salinities below 2 are calculated using the extension of the Practical Salinity Scale (Hill et al. 1986).

`R2pss` is the same function, except that conductivity ratios rather than conductivities are used as input.

Value

`ec2pss` and `R2pss` both return salinity values on the Practical Salinity Scale.

Note

Input pressures are not absolute pressures but rather gauge pressures. Gauge pressures are measured relative to 1 standard atmosphere, so the gauge pressure at the surface is 0.

Author(s)

Alan Jassby, James Cloern

References

Fofonoff N.P. and Millard Jr R.C. (1983) *Algorithms for Computation of Fundamental Properties of Seawater.* UNESCO Technical Papers in Marine Science 44. UNESCO, Paris, 53 p.

Hill K.D., Dauphinee T.M. and Woods D.J. (1986) The extension of the Practical Salinity Scale 1978 to low salinities. *IEEE Journal of Oceanic Engineering* **11,** 109-112.

Examples

```
# Check values from Fofonoff and Millard (1983):
R = c(1, 1.2, 0.65)
t = c(15, 20, 5)
p = c(0, 2000, 1500)
R2pss(R, t, p)  # 35.000 37.246 27.995
# Repeat calculation with equivalent conductivity values by setting
# ec <- R * C(35, 15, 0):
ec = c(1, 1.2, 0.65) * 42.9140
ec2pss(ec, t, p)  # same results
```

`eof` *Empirical orthogonal function analysis*

Description

Finds and rotates empirical orthogonal functions (EOFs).

Usage

```
eof(x, n, scale. = TRUE)
```

Arguments

`x`	a data frame or matrix, with no missing values
`n`	number of EOFs to retain for rotation
`scale.`	logical indicating whether the (centered) variables should be scaled to have unit variance

Details

EOF analysis is used to study patterns of variability ("modes") in a matrix time series and how these patterns change with time ("amplitude time series"). Hannachi et al. (2007) give a detailed discussion of this exploratory approach with emphasis on meteorological data. In oceanography and climatology, the time series represent observations at different spatial locations (columns) over time (rows). But columns can also be seasons of the year (Jassby et al. 1999) or even a combination of seasons and depth layers (Jassby et al. 1990). EOF analysis uses the same techniques as principal component analysis, but the time series are observations of the same variable in the same units. Scaling the data is optional, but it is the default here.

Eigenvectors (unscaled EOFs) and corresponding eigenvalues (amount of explained variance) are found by singular value decomposition of the centered and (optionally) scaled data matrix using `prcomp`. In order to facilitate a physical interpretation of the variability modes, a subset consisting of the `n` most important EOFs is rotated (Richman 1986). `eofNum` can be used to help choose `n`. Hannachi et al. (2007) recommend orthogonal rotation of EOFs scaled by the square root of the corresponding eigenvalues to avoid possible computation problems and reduce sensitivity to the

choice of n. We follow this recommendation here, using the varimax method for the orthogonal rotation.

Note that the signs of the EOFs are arbitrary.

Value

A list with the following members:

REOF	a matrix with rotated EOFs
amplitude	a matrix with amplitude time series of REOFs
eigen.pct	all eigenvalues of correlation matrix as percent of total variance
variance	variance explained by retained EOFs

Author(s)

Alan Jassby, James Cloern

References

Hannachi, A., Jolliffe, I.T., and Stephenson, D.B. (2007) Empirical orthogonal functions and related techniques in atmospheric science: A review. *International Journal of Climatology* **27,** 1119–1152.

Jassby, A.D., Powell, T.M., and Goldman, C.R. (1990) Interannual fluctuations in primary production: Direct physical effects and the trophic cascade at Castle Lake, California (USA). *Limnology and Oceanography* **35,** 1021–1038.

Jassby, A.D., Goldman, C.R., Reuter, J.E., and Richards, R.C. (1999) Origins and scale dependence of temporal variability in the transparency of Lake Tahoe, California-Nevada. *Limnology and Oceanography* **44,** 282–294.

Richman, M. (1986) Rotation of principal components. *Journal of Climatology* **6,** 293–335.

See Also

eofNum, eofPlot, monthCor, ts2df

Examples

```
# Create an annual matrix time series
chla1 <- aggregate(sfbayChla, 1, mean, na.rm = TRUE)
chla1 <- chla1[, 1:12]  # remove stations with missing years
# eofNum (see examples) suggests n = 1
eof(chla1, 1)
```

eofNum *Plot EOF percent variance*

Description

Plots the variances associated with empirical orthogonal functions (EOF). Useful for deciding how many EOFs to retain for rotation.

Usage

```
eofNum(x, n = nrow(x), scale. = TRUE)
```

Arguments

x	a data frame or matrix, with no missing values
n	effective sample size
scale.	logical indicating whether the (centered) variables should be scaled to have unit variance

Details

Calculates the eigenvalues from an EOF analysis, as described in eof. The eigenvalues are plotted against eigenvalue number (sometimes called a "scree plot"), and the cumulative variance as % of total is plotted over each eigenvalue. The approximate 0.95 confidence limits are depicted for each eigenvalue using North et al.'s (1982) rule-of-thumb, which ignores any autocorrelation in the data. If the autocorrelation structure is assessed separately and can be expressed in terms of effective sample size (e.g., Thiebaux and Zwiers 1984), then n can be set equal to this number.

There is no universal rule for deciding how many of the EOFs should be retained for rotation (Hannachi et al. 2007). In practice, the number is chosen by requiring a minimum cumulative variance, looking for a sharp break in the spectrum, requiring that confidence limits not overlap, various Monte Carlo methods, or many other techniques. The plot produced here enables the first three methods.

Value

A plot of the eigenvectors.

Author(s)

Alan Jassby, James Cloern

References

Hannachi, A., Jolliffe, I.T., and Stephenson, D.B. (2007) Empirical orthogonal functions and related techniques in atmospheric science: A review. *International Journal of Climatology* **27,** 1119–1152.

North, G., Bell, T., Cahalan, R., and Moeng, F. (1982) Sampling errors in the estimation of empirical orthogonal functions. *Monthly Weather Review* **110,** 699–706.

Thiebaux H.J. and Zwiers F.W. (1984) The interpretation and estimation of effective sample sizes. *Journal of Climate and Applied Meteorology* **23,** 800–811.

See Also

eof, interpTs, monthCor, eofPlot

Examples

```
# Create an annual time series data matrix from sfbay chlorophyll data
# Average over each year
chla1 <- aggregate(sfbayChla, 1, mean, na.rm = TRUE)
chla1 <- chla1[, 1:12]  # remove stations with missing years
eofNum(chla1)
# These stations appear to act as one with respect to chlorophyll
# variability on the annual scale because there's one dominant EOF.
```

---

eofPlot *Plot EOF analysis results*

Description

Plots the rotated empirical orthogonal functions or amplitude time series resulting from eof.

Usage

```
eofPlot(x, type = c("coef", "amp"), rev = FALSE, ord = FALSE)
```

Arguments

x	result of the function eof
type	whether the EOF coefficients or amplitudes should be plotted
rev	logical indicating whether coefficients and amplitudes should be multiplied by -1
ord	logical indicating whether coefficients should be ordered by size

Details

When the columns of the original data have a natural order, such as stations along a transect or months of the year, there may be no need to reorder the EOF coefficients. But if there is no natural order, such as when columns represents disparate sites around the world, the plot can be more informative if coefficients are ordered by size (`ord = TRUE`).

Coefficients and amplitudes for a given EOF may be more easily interpreted if `rev = TRUE`, because the sign of the first coefficient is arbitrarily determined and all the other signs follow from that choice.

Value

A plot of the EOF coefficients or amplitudes.

Author(s)

Alan Jassby, James Cloern

See Also

`eof`

Examples

```
# Create an annual matrix time series
chla1 <- aggregate(sfbayChla, 1, mean, na.rm = TRUE)
chla1 <- chla1[, 1:12]  # remove stations with missing years

# eofNum (see examples) suggests n = 1
e1 <- eof(chla1, n = 1)
eofPlot(e1, type = 'coef')
eofPlot(e1, type = 'amp')
```

---

interpTs *Interpolate or substitute missing time series values*

Description

Imterpolates or substitutes missing data in a time series for gaps up to a specified size.

Usage

```
interpTs(
  x,
 type = c("linear", "series.median", "series.mean", "cycle.median", "cycle.mean"),
  gap = NULL
)
```

Arguments

x	object of class "ts" or "mts"
type	method of interpolation or substitution
gap	maximum gap to be replaced

Details

When type = "linear", the function performs linear interpolation of any NA runs of length smaller than or equal to gap. When gap = NULL, gaps of any size will be replaced. Does not change leading or trailing NA runs. This interpolation approach is best for periods of low biological activity when sampling is routinely suspended.

When type = "series.median" or "series.mean", missing values are replaced by the overall median or mean, respectively. This may be desirable when missing values are not allowed but one wants, for example, to avoid spurious enhancement of trends.

When type = "cycle.median" or type = "cycle.mean", missing values are replaced by the median or mean, respectively, for the same cycle position (i.e., same month, quarter, etc., depending on the frequency). This may give more realistic series than using the overall mean or median.

Intended for time series but first three types will work with any vector or matrix. Matrices will be interpolated by column.

Value

The time series with some or all missing values replaced.

Author(s)

Alan Jassby, James Cloern

See Also

decompTs

Examples

```
### Interpolate a vector time series and highlight the imputed data
chl27 <- sfbayChla[, 's27']
x1 <- interpTs(chl27, gap = 3)
plot(x1, col = 'red')
lines(chl27, col = 'blue')
x2 <- interpTs(chl27, type = "series.median", gap = 3)
plot(x2, col = 'red')
lines(chl27, col = 'blue')

### Interpolate a matrix time series and plot results
x3 <- interpTs(sfbayChla, type = "cycle.mean", gap = 1)
plot(x3[, 1:10], main = "SF Bay Chl-a\n(gaps of 1 month replaced)")
```

`layerMean` *layerMean*

Description

Acts on a matrix or data frame with depth in the first column and observations for different variables (or different sites, or different times) in each of the remaining columns. The trapezoidal mean over the given depths is calculated for each of the variables. Replicate depths are averaged, and missing values or data with only one unique depth are handled. Data are not extrapolated to cover missing values at the top or bottom of the layer. The result can differ markedly from the simple mean even for equal spacing of depths, because the top and bottom values are weighted by 0.5 in a trapezoidal mean.

Usage

```
layerMean(d)
```

Arguments

d	data.frame

Author(s)

Alan Jassby, James Cloern

`leapYear` *leapYear*

Description

`TRUE` if `x` is a leap year, `FALSE` otherwise.

Usage

```
leapYear(x)
```

Arguments

x	integer year

Author(s)

Alan Jassby, James Cloern

mannKen *Mann-Kendall trend test and the Sen slope*

Description

Applies Kendall's tau test for the significance of a monotonic time series trend (Mann 1945). Also calculates the Sen slope as an estimate of this trend.

Usage

```
mannKen(
  x,
  plot = FALSE,
  type = c("slope", "relative"),
  order = FALSE,
  pval = 0.05,
  pchs = c(19, 21),
  ...
)
```

Arguments

x	A numeric vector, matrix or data frame
plot	Should the trends be plotted when x is a matrix or data frame?
type	Type of trend to be plotted, actual or relative
order	Should the plotted trends be ordered by size?
pval	p-value for significance
pchs	Plot symbols for significant and not significant trend estimates, respectively
...	Other arguments to pass to plotting function

Details

The Sen slope (alternately, Theil or Theil-Sen slope)—the median slope joining all pairs of observations—is expressed by quantity per unit time. The fraction of missing slopes involving the first and last fifths of the data are provided so that the appropriateness of the slope estimate can be assessed and results flagged. Schertz et al. [1991] discuss this and related decisions about missing data. Other results are used for further analysis by other functions. Serial correlation is ignored, so the interval between points should be long enough to avoid strong serial correlation.

For the relative slope, the slope joining each pair of observations is divided by the first of the pair before the overall median is taken. The relative slope makes sense only as long as the measurement scale is non-negative (not, e.g., temperature on the Celsius scale). Comparing relative slopes is useful when the variables in x have different units.

If `plot = TRUE`, then either the Sen slope (`type = "slope"`) or the relative Sen slope (`type = "relative"`) are plotted. The plot symbols indicate, respectively, that the trend is significant or not significant. The plot can be customized by passing any arguments used by `dotchart` such as `xlab` or `xlim`, as well as graphical parameters described in `par`.

Value

A list of the following if x is a vector:

sen.slope	Sen slope
sen.slope.rel	Relative Sen slope
p.value	Significance of slope
S	Kendall's S
varS	Variance of S
miss	Fraction of missing slopes connecting first and last fifths of x

or a matrix with corresponding columns if x is a matrix or data frame.

Note

Approximate p-values with corrections for ties and continuity are used if $n > 10$ or if there are any ties. Otherwise, exact p-values based on Table B8 of Helsel and Hirsch (2002) are used. In the latter case, $p = 0.0001$ should be interpreted as $p < 0.0002$.

Author(s)

Alan Jassby, James Cloern

References

Mann, H.B. (1945) Nonparametric tests against trend. *Econometrica* **13,** 245–259.

Helsel, D.R. and Hirsch, R.M. (2002) *Statistical methods in water resources.* Techniques of Water Resources Investigations, Book 4, chapter A3. U.S. Geological Survey. 522 pages. http://pubs.usgs.gov/twri/twri4a3/

Schertz, T.L., Alexander, R.B., and Ohe, D.J. (1991) *The computer program EStimate TREND (ESTREND), a system for the detection of trends in water-quality data.* Water-Resources Investigations Report 91-4040, U.S. Geological Survey.

See Also

seaKen, seasonTrend, tsSub

Examples

```
tsp(Nile)  # an annual time series
mannKen(Nile)

y <- sfbayChla
y1 <- interpTs(y, gap=1)  # interpolate single-month gaps only
y2 <- aggregate(y1, 1, mean, na.rm=FALSE)
mannKen(y2)
mannKen(y2, plot=TRUE) # missing data means missing trend estimates
mannKen(y2, plot=TRUE, xlim = c(0.1, 0.25))
mannKen(y2, plot=TRUE, type='relative', order = TRUE, pval = .001,
```

```
  xlab = "Relative trend")
legend("topleft", legend = "p < 0.001", pch = 19, bty="n")
```

---

meanSub *meanSub*

Description

meanSub

Usage

```
meanSub(x, sub, na.rm = FALSE)
```

Arguments

x	numeric vector
sub	integer index
na.rm	logical

Author(s)

Alan Jassby, James Cloern

---

monthCor *monthCor*

Description

monthCor

Usage

```
monthCor(x)
```

Arguments

x	ts

Author(s)

Alan Jassby, James Cloern

monthNum *monthNum*

Description

Converts dates to the corresponding numeric month.

Usage

```
monthNum(y)
```

Arguments

y	date

Author(s)

Alan Jassby, James Cloern

mts2ts *Converts matrix to vector time series for various analyses*

Description

First aggregates multivariate matrix time series by year. Then converts to a vector time series in which "seasons" correspond to these annualized values for the original variables.

Usage

```
mts2ts(x, seas = 1:frequency(x), na.rm = FALSE)
```

Arguments

x	An object of class "mts"
seas	Numeric vector of seasons to aggregate in original time series.
na.rm	Should missing data be ignored when aggregating?

Details

The seas parameter enables focusing the subsequent analysis on seasons of special interest, or to ignore seasons where there are too many missing data. The function can be used in conjunction with seaKen to conduct a Regional Kendall trend analysis. Sometimes just plotting the resulting function can be useful for exploring a spatial transect over time.

Value

A vector time series

Author(s)

Alan Jassby, James Cloern

See Also

seaKen

Examples

```
## Quick plot a spatial transect of chlorophyll a during the
## spring bloom period (Feb-Apr) for each year.
y <- mts2ts(sfbayChla, seas = 2:4)
plot(y, type = 'n')
abline(v = 1978:2010, col = 'lightgrey')
lines(y, type = 'h')
```

---

oxySol *Dissolved oxygen at saturation*

Description

Finds dissolved oxygen concentration in equilibrium with water-saturated air.

Usage

```
oxySol(t, S, P = NULL)
```

Arguments

t	temperature, degrees C
S	salinity, on the Practical Salinity Scale
P	pressure, atm

Details

Calculations are based on the approach of Benson and Krause (1984), using Green and Carritt's (1967) equation for dependence of water vapor partial pressure on t and S. Equations are valid for temperature in the range 0-40 C and salinity in the range 0-40.

Value

Dissolved oxygen concentration in mg/L at 100% saturation. If P = NULL, saturation values at 1 atm are calculated.

Author(s)

Alan Jassby, James Cloern

References

Benson, B.B. and Krause, D. (1984) The concentration and isotopic fractionation of oxygen dissolved in fresh-water and seawater in equilibrium with the atmosphere. *Limnology and Oceanography* **29,** 620-632.

Green, E.J. and Carritt, D.E. (1967) New tables for oxygen saturation of seawater. *Journal of Marine Research* **25,** 140-147.

Examples

```
# Convert DO into % saturation for 1-m depth at Station 32.
# Use convention of expressing saturation at 1 atm.
sfb1 <- subset(sfbay, depth == 1 & stn == 32)
dox.pct <- with(sfb1, 100 * dox/oxySol(temp, sal))
summary(dox.pct)
```

---

pett *Nonparametric Change-Point Detection*

Description

Locates a single change-point in an annual series based on the Pettitt test.

Usage

```
pett(x, plot = FALSE, order = FALSE, pval = 0.05, pchs = c(19, 21), ...)
```

Arguments

x	a numeric vector, matrix or data frame with no missing interior values
plot	Should the trends be plotted when x is a matrix?
order	Should the plotted trends be ordered by size?
pval	p-value for significance
pchs	Plot symbols for significant and not significant trend estimates, respectively
...	Other arguments to pass to plotting function

Details

Pettitt's (1979) method is a rank-based nonparametric test for abrupt changes in a time series. It uses the Mann-Whitney statistic for testing that two samples (before and after the change-point) come from the same distribution, choosing the change-point that maximizes the statistic. The p-value is approximate but accurate to 0.01 for $p \leq 0.5$. Serial correlation is ignored, so the interval between points should be long enough to avoid strong serial correlation. The size of the change is estimated as the median difference between all pairs of observations in which the first one is after the change-point and the second is up to the change-point.

Missing values are allowed at the beginning or end of each variable but interior missing values will produce an NA. Otherwise the change-point might not be meaningful.

If `plot = TRUE`, a dot plot of `change.times` is shown. If `sort = TRUE`, the dots are sorted by `change.time`. The plot symbols indicate, respectively, that the trend is significant or not significant. The plot can be customized by passing any arguments used by `dotchart` such as `xlab`, as well as graphical parameters described in `par`.

Value

A list of the following if `x` is a vector:

`pettitt.K`	Pettitt's statistic
`p.value`	significance probability for statistic
`change.point`	last position preceding change to new level
`change.time`	if available, time of change.point position
`change.size`	median of all differences between points after and up to change.point

or a matrix with corresponding columns if `x` is a matrix or data frame.

Note

The `change.point` returned by these functions is the last position before the series actually changes, for consistency with the original Pettitt test. But for reporting purposes, the following position might be more appropriate to call the "change-point".

The Pettitt test produces a supposed change-point, even when the trend is smooth, or when the abrupt change is smaller than the long-term smooth change. Remove any smooth, long-term trend before applying this test.

Author(s)

Alan Jassby, James Cloern

References

Pettitt, A. N. (1979) A non-parametric approach to the change-point problem. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **28(2),** 126–135.

Examples

```
# data from Pettitt (1979, Table 1):
y <- c(-1.05, 0.96, 1.22, 0.58, -0.98, -0.03, -1.54, -0.71, -0.35, 0.66,
0.44, 0.91, -0.02, -1.42, 1.26, -1.02, -0.81, 1.66, 1.05, 0.97, 2.14, 1.22,
-0.24, 1.60, 0.72, -0.12, 0.44, 0.03, 0.66, 0.56, 1.37, 1.66, 0.10, 0.80,
1.29, 0.49, -0.07, 1.18, 3.29, 1.84)
pett(y) # K=232, p=0.0146, change-point=17, the same results as Pettitt
# identify the year of a change-point in an annual time series:
pett(Nile)
# apply to a matrix time series:
y <- ts.intersect(Nile, LakeHuron)
pett(y)
pett(y, plot = TRUE, xlab = "Change-point")
legend("topleft", legend = "p < 0.05", pch = 19, bty="n")
# note how a smooth trend can disguise a change-point:
# smooth trend with change-point at 75
y <- 1:100 + c(rep(0, 75), rep(10, 25))
pett(y) # gives 50, erroneously
pett(residuals(lm(y~I(1:100)))) # removing trend gives 75, correctly
```

---

phenoAmp *Phenological amplitude*

Description

Finds various measures of the amplitude of the annual cycle, or of some specified season range.

Arguments

x	A seasonal time series, or a class zoo object.
season.range	A vector of two numbers specifying the season range to be considered.

Details

phenoAmp gives three measures of the amplitude of a seasonal cycle: the range, the variance, and the median absolute deviation, along with the mean and median to allow calculation of other statistics as well.

These measures can be restricted to a subset of the year by giving the desired range of season numbers. This can be useful for isolating measures of, say, the spring and autumn phytoplankton blooms in temperate waters. In the case of a monthly time series, for example, a non-missing value is required for every month or the result will be NA, so using a period shorter than one year can also help avoid any months that are typically not covered by the sampling program. Similarly, in the case of dated observations, a shorter period can help avoid times of sparse data. The method for time series allows for other than monthly frequencies, but season.range is always interpreted as months for zoo objects.

Note that the amplitude is sensitive to the number of samples for small numbers. This could be a problem for zoo objects if the sample number is changing greatly from year to year, depending on

the amplitude measure and the underlying data distribution. So use ts objects or make sure that the sample number stays more or less the same over time.

tsMake can be used to produce ts and zoo objects suitable as arguments to this function.

Value

A matrix of class ts or zoo with individual series for the range, variance, median absolute deviation, mean, median and – in the case of zoo objects – number of samples.

References

Cloern, J.E. and Jassby, A.D. (2008) Complex seasonal patterns of primary producers at the land-sea interface. *Ecology Letters* **11,** 1294–1303.

See Also

phenoPhase, tsMake

Examples

```
y <- sfbayChla[, "s27"]
phenoAmp(y) # entire year
# i.e., Jan-Jun only, which yields results for more years
phenoAmp(y, c(1, 6))
```

---

phenoAmp-methods *Methods for Function phenoAmp*

Description

Finds various measures of the amplitude of the annual cycle.

Methods

list("signature(x = \"ts\")") See phenoAmp,ts-method

list("signature(x = \"zoo\")") See phenoAmp,zoo-method

phenoPhase *Phenological phase*

Description

Finds various measures of the phase of the annual cycle, or of some specified month range.

Arguments

x	A seasonal time series, or a class zoo object.
season.range	A vector of two numbers specifying the season range to be considered.
out	The form of the output.
...	Additional arguments to be passed for changing integration defaults.

Details

phenoPhase gives three measures of the phasing of a seasonal cycle: the time of the maximum (Cloern and Jassby 2008), the *fulcrum* or center of gravity, and the weighted mean season (Colebrook 1979). The latter has sometimes been referred to in the literature as "centre of gravity", but it is not actually the same. These measures differ in their sensitivity to changes in the seasonal pattern, and therefore also in their susceptibility to sampling variability. The time of maximum is the most sensitive, the weighted mean the least.

These measures can be restricted to a subset of the year by giving the desired range of seasons. This can be useful for isolating measures of, say, the spring and autumn phytoplankton blooms in temperate waters. In the case of a seasonal time series, a non-missing value is required for every season or the result will be NA, so using a period shorter than one year can also help avoid any seasons that are typically not covered by the sampling program. Similarly, in the case of dated observations, a shorter period can help avoid times of sparse data. The method for time series allows for other than monthly frequencies, but season.range is always interpreted as months for zoo objects. The method for time series requires data for all seasons in season.range. The method for zoo objects will provide a result regardless of number of sampling days, so make sure that data are sufficient for a meaningful result.

The measures are annum-centric, i.e., they reflect the use of calendar year as the annum, which may not be appropriate for cases in which important features occur in winter and span two calendar years. Such cases can be handled by lagging the time series by an appropriate number of months, or by subtracting an appropriate number of days from the individual dates.

tsMake can be used to produce ts and zoo objects suitable as arguments to this function.

The default parameters used for the integrate function in phenoPhase may fail for certain datasets. Try increasing the number of subdivisions above its default of 100 by adding, for example, subdivisions = 1000 to the arguments of phenoPhase.

Value

A data frame with columns year, time of the maximum, fulcrum, weighted mean time and – in the case of zoo objects – number of observations. In the case of seasonal time series, the results are all given as decimal seasons of the year. In the case of dated observations, the results can be dates, day of the year, or julian day with an origin of 1970-01-01, depending on the option out.

References

Cloern, J.E. and Jassby, A.D. (2008) Complex seasonal patterns of primary producers at the land-sea interface. *Ecology Letters* **11,** 1294–1303.

Colebrook, J.M. (1979) Continuous plankton records - seasonal cycles of phytoplankton and copepods in the North Atlantic ocean and the North Sea. *Marine Biology* **51,** 23–32.

See Also

phenoAmp, tsMake

Examples

```
# ts example
y <- sfbayChla[, "s27"]
p1 <- phenoPhase(y)
p1
apply(p1, 2, sd, na.rm = TRUE)  # max.time > fulcrum > mean.wt
phenoPhase(y, c(3, 10))

# zoo example
sfb <- wqData(sfbay, c(1, 3, 4), 5:12, site.order = TRUE, type = "wide",
  time.format = "%m/%d/%Y")
y <- tsMake(sfb, focus = "chl", layer = c(0, 5), type = "zoo")
phenoPhase(y[, "s27"])
```

---

phenoPhase-methods *Methods for Function phenoPhase*

Description

Finds various measures of the phase of the annual cycle.

Methods

list("signature(x = \"ts\")") See phenoPhase,ts-method

list("signature(x = \"zoo\")") See phenoPhase,zoo-method

`plotSeason` *Plots seasonal patterns for a time series*

Description

Divides the time range for a monthly time series into different eras and plots composites of seasonal pattern. Can also plot each month separately for the entire record.

Usage

```
plotSeason(
  x,
  type = c("by.era", "by.month"),
  num.era = 4,
  same.plot = TRUE,
  ylab = NULL,
  num.col = 3
)
```

Arguments

`x`	Monthly time series
`type`	Plot seasonal pattern by era, or each month for the entire record
`num.era`	Integer number of eras, or vector of era year breaks
`same.plot`	Should eras be plotted by month?
`ylab`	Optional character string label for y-axis
`num.col`	Number of columns when plotted `"by.month"`

Details

If `num.era` is an integer, the time range is divided into that many equal eras; otherwise, the time range is divided into eras determined by the `num.era` vector of years. When plotted `"by.era"` and `same.plot = FALSE`, the composite patterns are plotted in a horizontal row for easier comparison, which limits the number of periods that can be examined. Boxes based on fewer than half of the maximum possible years available are outlined in red. If `same.plot = TRUE`, a single plot is produced with era boxplots arranged by month. When plotted `"by.month"`, values for each month are first converted to standardized anomalies, i.e., by subtraction of long-term mean and division by standard deviation. As always, and especially with these plots, experiment with the device aspect ratio and size to get the clearest information.

Value

A plot (and the corresponding object of class `"ggplot"`).

Author(s)

Alan Jassby, James Cloern

See Also

decompTs, seasonTrend

Examples

```
chl27 <- sfbayChla[, "s27"]
plotSeason(chl27, num.era = c(1978, 1988, 1998, 2008), ylab = "Stn 27 Chl-a")
## Not run:
plotSeason(chl27, num.era = 3, same.plot = FALSE, ylab = "Stn 27 Chl-a")
plotSeason(chl27, "by.month", ylab = "Stn 27 Chl-a")

## End(Not run)
```

---

plotTs *Time series plot*

Description

Creates line plot of vector or matrix time series, including any data surrounded by NAs as additional points.

Usage

```
plotTs(
  x,
  dot.size = 1,
  xlab = NULL,
  ylab = NULL,
  strip.labels = colnames(x),
  ...
)
```

Arguments

x	matrix or vector time series
dot.size	size of dots representing isolated data points
xlab	optional x-axis label
ylab	optional y-axis label
strip.labels	labels for individual time series plots
...	additional options

Details

The basic time series line plot ignores data points that are adjacent to missing data, i.e., not directly connected to other observations. This can lead to an uninformative plot when there are many missing data. If one includes both a point and line plot, the resulting graph can be cluttered and difficult to decipher. `plotTs` plots only isolated points as well as lines joining adjacent observations.

Options are passed to the underlying `facet_wrap` function in **ggplot2**. The main ones of interest are `ncol` for setting the number of plotting columns and `scales = "free_y"` for allowing the y scales of the different plots to be independent.

Value

A plot or plots and corresponding object of class "ggplot".

Author(s)

Alan Jassby, James Cloern

See Also

`plotTsAnom`

Examples

```
# Chlorophyll at 4 stations in SF Bay
chl <- sfbayChla[, 1:4]
plotTs(chl, dot.size = 1.5, ylab = 'Chl-a', strip.labels = paste('Station',
  substring(colnames(chl), 2, 3)), ncol = 1, scales = "free_y")
```

`plotTsAnom` *Anomaly plot of time series*

Description

Series are illustrated by vertical lines extending from individual data values to the long-term mean. The axes are not scaled in any way. Anomaly plots are useful for visualizing shifts in time series levels.

Usage

```
plotTsAnom(x, xlab = NULL, ylab = NULL, strip.labels = colnames(x), ...)
```

Arguments

`x`	matrix or vector time series
`xlab`	optional x-axis label
`ylab`	optional y-axis label
`strip.labels`	labels for individual time series plots
`...`	additional options

Details

Options are passed to the underlying `facet_wrap` function in **ggplot2**. The main ones of interest are `ncol` for setting the number of plotting columns and `scales = "free_y"` for allowing the y scales of the different plots to be independent.

Value

A plot and corresponding object of class "ggplot".

Author(s)

Alan Jassby, James Cloern

See Also

`plotTs`

Examples

```
# Spring bloom size for 6 stations in SF Bay
bloom <- aggregate(sfbayChla[, 1:6], 1, meanSub, sub=3:5)
plotTsAnom(bloom, ylab = 'Chl-a', strip.labels = paste('Station',
  substring(colnames(bloom), 2, 3)), ncol = 2, scales = "free_y")
```

---

plotTsTile *Image plot of monthly time series*

Description

Monthly values are transformed into deciles or other bins, and corresponding colors are plotted in a month by year matrix.

Usage

```
plotTsTile(
  x,
  plot.title = NULL,
  legend.title = NULL,
  four = TRUE,
  loganom = TRUE,
  square = TRUE,
  legend = TRUE,
  trim = TRUE,
  overall = TRUE,
  stat = c("median", "mean")
)
```

Arguments

x	monthly time series.
plot.title	plot title.
legend.title	legend title.
four	logical indicating if data should be binned into 4 special groups or into deciles.
loganom	logical indicating if data should be transformed into log-anomalies.
square	logical indicating if tiles should be square.
legend	logical indicating if a legend should be included.
trim	logical indicating if leading and trailing NA values should be removed.
overall	determines whether anomalies are calculated with respect to overall mean or to long-term mean for the same month.
stat	determines whether anomalies are calculated and binned using mean or median.

Details

If four = TRUE, then x is first divided into a positive and negative bin. Each bin is then further divided into two bins by its mean, yielding a total of four bins. If four=FALSE, then x is simply divided into deciles. In either case, each bin has its own assigned color, with colors ranging from dark blue (smallest numbers) through light blue and pink to red.

Although four = TRUE can be useful for any data in which 0 represents a value with special significance, it is especially so for data converted into log-anomalies, i.e., log10(x/xbar) where xbar = mean(x,na.rm=TRUE). The mean month then has value 0, and a value of -1, for example, indicates original data equal to one-tenth the mean. Log-anomaly transforms can be particularly appropriate for biological populations, in which variability is often approximately proportional to the mean.

When loganom = TRUE, the anomalies are calculated with respect to the overall mean month. This differs from, for example, the log-anomaly zooplankton plot of O'Brien et al. (2008), in which a monthly anomaly is calculated with respect to the mean value of the same month. To get the latter behavior, set overall = FALSE. A further option is to set stat = "median" rather than the default stat = "mean", in which case xbar = median(x, na.rm = TRUE), and the positive and negative bins are each divided into two bins by their median instead of mean. Using combinations of these different options can reveal complementary information.

You may want to set square = FALSE and then adjust the plot window manually if you plan to use the plot in a subsequent layout or if there is too much white space.

Value

An image plot of monthly values classified into either deciles or into four bins as described above (and corresponding object of class "ggplot").

Author(s)

Alan Jassby, James Cloern

References

O'Brien T., Lopez-Urrutia A., Wiebe P.H., Hay S. (editors) (2008) *ICES Zooplankton Status Report 2006/2007*. ICES Cooperative Research Report 292, International Council for the Exploration of the Sea, Copenhagen, 168 p.

Examples

```
# plot log-anomalies in four bins
chl27 = sfbayChla[, 's27']
plotTsTile(chl27, legend.title = 'Chl log-anomaly')

# plot deciles
plotTsTile(chl27, plot.title = 'SF Bay station 27', legend.title =
'chlorophyll', four = FALSE, loganom = FALSE, square = FALSE)
```

`R2pss` *R2pss*

Description

R2pss

Usage

```
R2pss(R, t, p = 0)
```

Arguments

`R`	conductivity ratio, dimensionless
`t`	temperature, Celsius
`p`	gauge pressure, decibar

Author(s)

Alan Jassby, James Cloern

seaKen *Seasonal and Regional Kendall trend test*

Description

Calculates the Seasonal or Regional Kendall test of trend significance, including an estimate of the Sen slope.

Usage

```
seaKen(
  x,
  plot = FALSE,
  type = c("slope", "relative"),
  order = FALSE,
  pval = 0.05,
  mval = 0.5,
  pchs = c(19, 21),
  ...
)
```

Arguments

x	A numeric vector, matrix or data frame made up of seasonal time series
plot	Should the trends be plotted when x is a matrix or data frame?
type	Type of trend to be plotted, actual or relative to series median
order	Should the plotted trends be ordered by size?
pval	p-value for significance
mval	Minimum fraction of seasons needed with non-missing slope estimates
pchs	Plot symbols for significant and not significant trend estimates, respectively
...	Other arguments to pass to plotting function

Details

The Seasonal Kendall test (Hirsch et al. 1982) is based on the Mann-Kendall tests for the individual seasons (see mannKen for additional details). *p*-values provided here are not corrected for serial correlation among seasons.

If plot = TRUE, then either the Sen slope in units per year (type = "slope") or the relative slope in fraction per year (type = "relative") is plotted. The relative slope is defined identically to the Sen slope except that each slope is divided by the first of the two values that describe the slope. Plotting the relative slope is useful when the variables in x are always positive and have different units.

The plot symbols indicate, respectively, that the trend is statistically significant or not. The plot can be customized by passing any arguments used by dotchart such as xlab, as well as graphical parameters described in par.

If mval or more of the seasonal slope estimates are missing, then that trend is considered to be missing. The seasonal slope estimate (mannKen), in turn, is missing if half or more of the possible comparisons between the first and last 20% of the years are missing.

The function can be used in conjunction with mts2ts to calculate a Regional Kendall test of significance for annualized data, along with a regional estimate of trend (Helsel and Frans 2006). See the examples below.

Value

A list of the following if x is a vector: seaKen returns a list with the following members:

sen.slope	Sen slope
sen.slope.pct	Sen slope as percent of mean
p.value	significance of slope
miss	for each season, the fraction missing of slopes connecting first and last 20% of the years

or a matrix with corresponding columns if x is a matrix or data frame.

Author(s)

Alan Jassby, James Cloern

References

Helsel, D.R. and Frans, L. (2006) Regional Kendall test for trend. *Environmental Science and Technology* **40(13),** 4066-4073.

Hirsch, R.M., Slack, J.R., and Smith, R.A. (1982) Techniques of trend analysis for monthly water quality data. *Water Resources Research* **18,** 107-121.

See Also

mannKen, mts2ts, trendHomog

Examples

```
# Seasonal Kendall test:
chl <- sfbayChla # monthly chlorophyll at 16 stations in San Francisco Bay
seaKen(sfbayChla[, 's27']) # results for a single series at station 27
seaKen(sfbayChla) # results for all stations
seaKen(sfbayChla, plot=TRUE, type="relative", order=TRUE)

# Regional Kendall test:
# Use mts2ts to change 16 series into a single series with 16 "seasons"
seaKen(mts2ts(chl))  # too many missing data
# better when just Feb-Apr, spring bloom period,
# but last 4 stations still missing too much.
seaKen(mts2ts(chl, seas = 2:4))
seaKen(mts2ts(chl[, 1:12], 2:4)) # more reliable result
```

seaRoll *Rolling Seasonal Kendall trend test*

Description

Calculates the Seasonal Kendall test of significance, including an estimate of the Sen slope, for rolling windows over a time series.

Usage

```
seaRoll(
  x,
  w = 10,
  plot = FALSE,
  pval = 0.05,
  mval = 0.5,
  pchs = c(19, 21),
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

x	A seasonal time series vector.
w	The window width for "rolling" estimates of slope.
plot	Indicates if a plot should be drawn
pval	p-value for significance
mval	Minimum fraction of seasons needed with non-missing slope estimates
pchs	Plot symbols for significant and not significant trend estimates, respectively
xlab	Optional label for x-axis
ylab	Optional label for y-axis
...	Other arguments to pass to plotting function

Details

The function seaRoll applies seaKen to rolling time windows of width w. A minimum w of five years is required. For any window, a season is considered missing if half or more of the possible comparisons between the first and last 20% of the years is missing. If mval or more of the seasons are missing, then that windowed trend is considered to be missing.

If plot = TRUE, a point plot will be drawn with the Sen slope plotted at the leading year of the trend window. The plot symbols indicate, respectively, that the trend is significant or not significant. The plot can be customized by passing any arguments used by plot.default, as well as graphical parameters described in par.

Value

seaRoll returns a matrix with one row per time window containing the Sen slope, the relative Sen slope, and the p-value. Rows are labelled with the leading year of the window.

Author(s)

Alan Jassby, James Cloern

See Also

seaKen

Examples

```
chl27 <- sfbayChla[, 's27']
seaRoll(chl27)
seaRoll(chl27, plot = TRUE)
```

---

seasonTrend *Determine seasonal trends*

Description

Finds the trend for each season and each variable in a time series.

Usage

```
seasonTrend(x, plot = FALSE, type = c("slope", "relative"), pval = 0.05, ...)
```

Arguments

x	Time series vector, or time series matrix with column names
plot	Should the results be plotted?
type	Type of trend to be plotted, actual or relative to series median
pval	p-value for significance
...	Further options to pass to plotting function

Details

The Mann-Kendall test is applied for each season and series (in the case of a matrix). The actual and relative Sen slope (actual divided by median for that specific season and series); the p-value for the trend; and the fraction of missing slopes involving the first and last fifths of the data are calculated (see mannKen).

If plot = TRUE, each season for each series is represented by a bar showing the trend. The fill colour indicates whether $p < 0.05$ or not. If the fraction of missing slopes is 0.5 or more, the corresponding trends are omitted.

Parameters can be passed to the plotting function, in particular, to `facet_wrap` in **ggplot2**. The most useful parameters here are `ncol` (or `nrow`), which determines the number of columns (or rows) of plots, and `scales`, which can be set to `"free_y"` to allow the y-axis to change for each time series. Like all **ggplot2** objects, the plot output can also be customized extensively by modifying and adding layers.

Value

A data frame with the following fields:

`series`	series names
`season`	season number
`sen.slope`	Sen slope in original units per year
`sen.slope.rel`	Sen slope divided by median for that specific season and series
`p`	p-value for the trend according to the Mann-Kendall test.
`missing`	Proportion of slopes joining first and last fifths of the data that are missing

Author(s)

Alan Jassby, James Cloern

See Also

`mannKen`, `plotSeason`, `facet_wrap`

Examples

```
x <- sfbayChla
seasonTrend(x)
seasonTrend(x, plot = TRUE, ncol = 4)
```

---

sfbay *San Francisco Bay water quality data*

Description

Selected observations and variables from U.S. Geological Survey water quality stations in south San Francisco Bay. Data include CTD and nutrient measurements.

Format

`sfbay` is a data frame with 23207 observations (rows) of 12 variables (columns):

`[, 1]`	`date`	date
`[, 2]`	`time`	time
`[, 3]`	`stn`	station code
`[, 4]`	`depth`	measurement depth
`[, 5]`	`chl`	chlorophyll *a*
`[, 6]`	`dox.pct`	dissolved oxygen
`[, 7]`	`spm`	suspended particulate matter
`[, 8]`	`ext`	extinction coefficient
`[, 9]`	`sal`	salinity
`[, 10]`	`temp`	water temperature
`[, 11]`	`nox`	nitrate + nitrite
`[, 12]`	`nhx`	ammonium

`sfbayStns` is a data frame with 16 observations of 6 variables:

`[, 1]`	`site`	station code
`[, 2]`	`description`	station description
`[, 3]`	`lat`	latitude
`[, 4]`	`long`	longitude
`[, 5]`	`depthMax`	maximum depth, in m
`[, 6]`	`distFrom36`	distance from station 36, in km

`sfbayVars` is a data frame with 7 observations of 3 variables:

`[, 1]`	`variable`	water quality variable code
`[, 2]`	`description`	description
`[, 3]`	`units`	measurement units

`sfbayChla` is a time series matrix (380 months x 16 stations) of average 0-5 m chlorophyll *a* concentrations calculated from the data in `sfbay`.

Details

The original downloaded dataset was modified by taking a subset of six well-sampled stations and the period 1985–2004. Variable names were also simplified. The data frames `sfbayStns` and `sfbayVars` describe the stations and water quality variables in more detail; they were created from information at the same web site. Note that the station numbers in `sfbayStns` have been prefixed with `s` to make station codes into legal variable names. `sfbayChla` was constructed from the entire downloaded sfbay dataset and encompasses the period 1969–2009.

Source

Downloaded from https://sfbay.wr.usgs.gov/water-quality-database/ on 2009-11-17.

Examples

```
data(sfbay)
str(sfbay)
str(sfbayStns)
str(sfbayVars)
plot(sfbayChla[, 1:10], main = "SF Bay Chl-a")
```

`trendHomog` *Trend homogeneity test*

Description

Tests for homogeneity of seasonal trends using method proposed by van Belle and Hughes (1984). Seasons with insufficient data as defined in `mannKen` are ignored.

Usage

```
trendHomog(x)
```

Arguments

`x`	A vector time series with frequency > 1

Value

`chisq.trend`	"Trend" chi-square.
`chisq.homog`	"Homogeneous" chi-square.
`p.value`	For null hypothesis that trends are homogeneous.
`n`	Number of seasons used.

Author(s)

Alan Jassby, James Cloern

References

van Belle, G. and Hughes, J.P. (1984) Nonparametric tests for trend in water quality. *Water Resources Research* **20,** 127-136.

See Also

`seaKen`

Examples

```
## Apply to a monthly vector time series to test homogeneity
## of seasonal trends.
x <- sfbayChla[, 's27']
trendHomog(x)
```

---

ts2df *Convert time series to data frame*

Description

Convert monthly time series vector to a year x month data frame for several possible subsequent analyses. Leading and trailing empty rows are removed.

Usage

```
ts2df(x, mon1 = 1, addYr = FALSE, omit = FALSE)
```

Arguments

x	monthly time series vector
mon1	starting month number, i.e., first column of the data frame
addYr	rows are normally labelled with the year of the starting month, but addYr = TRUE will add 1 to this year number
omit	if TRUE, then rows with any NA will be removed.

Details

Our main use of ts2df is to convert a single monthly time series into a year x month data frame for EOF analysis of interannual variability.

monthCor finds the month-to-month correlations in a monthly time series x. It is useful for deciding where to start the 12-month period for an EOF analysis (mon1 in ts2df), namely, at a time of low serial correlation in x.

Value

An n x 12 data frame, where n is the number of years.

Author(s)

Alan Jassby, James Cloern

References

Craddock, J. (1965) A meteorological application of principal component analysis. *Statistician* **15,** 143–156.

See Also

eof

Examples

```
# San Francisco Bay station 27 chlorophyll has the lowest serial
# correlation in Oct-Nov, with Sep-Oct a close second
chl27 <- sfbayChla[, 's27']
monthCor(chl27)

# Convert to a data frame with October, the first month of the
# local "water year", in the first column
tsp(chl27)
chl27 <- round(chl27, 1)
ts2df(chl27, mon1 = 10, addYr = TRUE)
ts2df(chl27, mon1 = 10, addYr = TRUE, omit = TRUE)
```

---

tsMake *Create time series from water quality data*

Description

Creates a matrix time series object from an object of class "WqData", either all variables for a single site or all sites for a single variable.

Arguments

object	Object of class "WqData".
focus	Name of a site or water quality variable.
layer	Number specifying a single depth; a numeric vector of length 2 specifying top and bottom depths of layer; a list specifying multiple depths and/or layers; or just the string "max.depths".
type	ts.mon to get a monthly time series, zoo to get an object of class "zoo" with individual observation dates.
qprob	quantile probability, a number between 0 and 1.

Details

When qprob = NULL, the function averages all included depths for each day, the implicit assumption being that the layer is well-mixed and/or the samples are evenly distributed with depth in the layer. If layer = "max.depths", then only the value at the maximum depth for each time, site and variable combination will be used. If no layer is specified, all depths will be used.

The function produces a matrix time series of all variables for the specified site or all sites for the specified variable. If type = "ts.mon", available daily data are averaged to produce a monthly time series, from which a quarterly or annual series can be created if needed. If you want values for the actual dates of observation, then set type = "zoo".

When qprob is a number from 0 to 1, it is interpreted as a probability and the corresponding quantile is used to aggregate observations within the specified layer. So to get the maximum, for example, use qprob = 1. If type = "ts.mon", the same quantile is used to aggregate all the available daily values.

Value

A matrix of class "mts" or "zoo".

Note

The layer list is allowed to include negative numbers, which may have been used in the WqData object to denote variables that apply to the water column as a whole, such as, say, -1 for light attenuation coefficient. This enables focus = 's27' and layer = list(-1, c(0, 5)) to produce a time series matrix for station 27 that includes both attenuation coefficient and chlorophyll averaged over the top 5 m. Negative numbers may also have been used in the WqData object to identify qualitative depths such as "near bottom", which is not uncommon in historical data sets. So data from such depths can be aggregated easily with other data to make these time series.

See Also

WqData-class

Examples

```
# Create new WqData object
sfb <- wqData(sfbay, c(1, 3:4), 5:12, site.order = TRUE,
  time.format = "%m/%d/%Y", type = "wide")

# Find means in the 0-10 m layer
y <- tsMake(sfb, focus = "s27", layer = c(0, 10))
plot(y, main = "Station 27")
# Or select medians in the same layer
y1 <- tsMake(sfb, focus = "s27", layer = c(0, 10), qprob = 0.5)
plot(y1, main = "Station 27")
# Compare means:medians
apply(y / y1, 2, mean, na.rm = TRUE)

# Combine a layer with a single additional depth
y <- tsMake(sfb, focus = "chl", layer = list(c(0, 2), 5))
plot(y, main = "Chlorophyll a, ug/L")

# Use values from the deepest samples
y <- tsMake(sfb, focus = "dox", layer = "max.depths", type = "zoo")
head(y)
plot(y, type = "h", main = "'Bottom' DO, mg/L")
```

tsMake-methods *Methods for Function tsMake*

Description

Creates a matrix of observations indexed by time.

Methods

list("signature(x = \"WqData\")") See tsMake,WqData-method

tsSub *tsSub*

Description

tsSub

Usage

```
tsSub(x1, seas = 1:frequency(x1))
```

Arguments

x1	ts
seas	numeric

Author(s)

Alan Jassby, James Cloern

wqData *Construct an object of class "WqData"*

Description

wqData is a constructor for the "WqData" class that is often more convenient to use than new. It converts a data.frame containing water quality data in "long" or "wide" format to a "WqData" object. In "long" format, observations are all in one column and a second column is used to designate the variable being observed. In "wide" format, observations for each variable are in a separate column.

Usage

```
wqData(
  data,
  locus,
  wqdata,
  site.order,
  time.format = "%Y-%m-%d",
  type = c("long", "wide")
)
```

Arguments

data	Data frame containing water quality data.
locus	Character or numeric vector designating column names or numbers, respectively, in data that correspond to time, site and depth.
wqdata	In the case of "long" data, character or numeric vector designating column names or numbers, respectively, in data that correspond to variable and value. In the case of "wide" data, character or numeric vector designating column names or numbers, respectively, in data that denote water quality variable data.
site.order	If TRUE, site factor levels will be ordered in alphanumeric order.
time.format	Conversion specification for time defined by ISO C/POSIX standard (see strptime).
type	Either "long" or "wide" data.

Details

If the data are already in long format, the function has little to do but rename the data fields. If in wide format, the **reshape2** package is called to melt the data. The function also removes NA observations, converts site to (possibly ordered) factors with valid variable names, and converts time to class "Date" or "POSIXct" and ISO 8601 format, depending on time.format.

Value

An object of class "WqData".

Author(s)

Alan Jassby, James Cloern

References

International Organization for Standardization (2004) ISO 8601. Data elements and interchange formats - Information interchange - Representation of dates and times.

See Also

as.Date, strptime, WqData-class

Examples

```
## Not run:

# Create new WqData object from sfbay data. First combine date and time
# into a single string after making sure that all times have 4 digits.
sfb <- within(sfbay, time <- substring(10000 + time, 2, 5))
sfb <- within(sfb, time <- paste(date, time, sep = ' '))
sfb <- wqData(sfb, 2:4, 5:12, site.order = TRUE, type = "wide",
    time.format = "%m/%d/%Y %H%M")

head(sfb)
tail(sfb)

# If time of day were not required, then the following would suffice:
sfb <- wqData(sfbay, c(1,3,4), 5:12, site.order = TRUE, type = "wide",
  time.format = "%m/%d/%Y")

## End(Not run)
```

WqData-class *Class WqData*

Description

A simple extension or subclass of the "data.frame" class for typical "discrete" water quality monitoring programs that examine phenomena on a time scale of days or longer. It requires water quality data to be in a specific "long" format, although a generating function wqData can be used for different forms of data.

Objects from the Class

Objects can be created by calls of the form new("WqData", d), where d is a data.frame. d should have columns named time, site, depth, variable, value of class "DateTime", "factor", "numeric", "factor", "numeric", respectively.

See Also

DateTime-class, tsMake,WqData-method, wqData

Examples

```
showClass("WqData")
# Construct the WqData object sfb as shown in the wqData examples.
sfb <- wqData(sfbay, c(1, 3, 4), 5:12, site.order = TRUE, type = "wide",
  time.format = "%m/%d/%Y")
# Summarize the data
summary(sfb)
# Create boxplot summary of data
```

```
plot(sfb, vars = c("chl", "dox", "spm"), num.col = 2)
# Extract some of the data as a WqData object
sfb[1:10, ]  # first 10 observations
sfb[sfb$depth == 20, ]  # all observations at 20 m
```

---

years *Miscellaneous utility functions*

Description

A variety of small utilities used in other functions.

Arguments

d	A numeric matrix or data frame with depth in the first column and observations for some variable in each of the remaining columns.
na.rm	Should missing data be removed?
seas	An integer vector of seasons to be retained.
sub	An integer vector.
w	A vector of class "Date".
x	A numeric vector.
x1	A matrix or vector time series.
y	A vector of class "Date" or "POSIX" date-time.

Details

date2decyear: Converts object of class "Date" to decimal year assuming time of day is noon.

decyear2date: Converts decimal year to object of class "Date".

layerMean: Acts on a matrix or data frame with depth in the first column and observations for different variables (or different sites, or different times) in each of the remaining columns. The trapezoidal mean over the given depths is calculated for each of the variables. Replicate depths are averaged, and missing values or data with only one unique depth are handled. Data are not extrapolated to cover missing values at the top or bottom of the layer. The result can differ markedly from the simple mean even for equal spacing of depths, because the top and bottom values are weighted by 0.5 in a trapezoidal mean.

leapYear: TRUE if x is a leap year, FALSE otherwise.

meanSub: Mean of a subset of a vector.

monthNum: Converts dates to the corresponding numeric month.

tsSub: Drops seasons from a matrix or vector time series.

years: Converts dates to the corresponding numeric years.

Examples

```
dates <- as.Date(c("1996-01-01", "1999-12-31", "2004-02-29", "2005-03-01"))
date2decyear(dates)

decyear2date(c(1996.0014, 1999.9986, 2004.1626, 2005.1630))

z <- c(1, 2, 3, 5, 10)  # 5 depths
x <- matrix(rnorm(30), nrow = 5)  # 6 variables at 5 depths
layerMean(cbind(z, x))

leapYear(seq(1500, 2000, 100))
leapYear(c(1996.9, 1997))

## Aggregate monthly time series over Feb-Apr only.
aggregate(sfbayChla, 1, meanSub, sub = 2:4)

monthNum(as.Date(c("2007-03-17", "2003-06-01")))

## Ignore certain seasons in a Seasonal Kendall test.
c27 <- sfbayChla[, "s27"]
seaKen(tsSub(c27))  # Aug and Dec missing the most key data
seaKen(tsSub(c27, seas = c(1:7, 9:11)))

y <- Sys.time()
years(y)
```

`zoo-class` *Class "zoo"*

Description

Registration of S3 class `"zoo"` as a formally defined class. Used here to allow the `"zoo"` class to appear in method signatures.

Objects from the Class

A virtual Class: No objects may be created from it.

See Also

phenoAmp, phenoPhase

Examples

```
showClass("zoo")
```

Index