Package 'wq'

July 2, 2014

Type Package

Title Exploring water quality monitoring data

Version 0.4-1

Date 2014-05-05

Author Alan D. Jassby and James E. Cloern

Maintainer Anthony Malkassian <anthonym@sfei.org>

Description Functions to assist in the processing and exploration of data from environmental monitoring programs. The name ``wq" stands for ``water quality" and reflects the original focus on time series data for physical and chemical properties of water, as well as the plankton. The package is intended for programs that sample approximately monthly 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 methods, R (>= 3.0.0), zoo

Suggests ggplot2 (>= 0.9), grid, reshape2

License GPL-2

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-05-15 01:03:53

R topics documented:

`wq-package` *Exploring water quality monitoring data*

Description

Functions to assist in the processing and exploration of data from environmental monitoring programs. The name "wq" stands for "water quality" and reflects the original focus on time series data for physical and chemical properties of water, as well as the plankton. The package is intended for programs that sample approximately monthly 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

Package:	wq
Type:	Package
Version:	0.4-1
Date:	2014-05-05
Depends:	methods, R (>= 3.0.0), zoo
Suggests:	ggplot2 (>= 0.9), grid, reshape2
License:	GPL-2
LazyData:	yes

Index:

Further information is available in the following vignettes:

wq-package wq: exploring water quality monitoring data (source, pdf)

Author(s)

Alan D. Jassby and James E. Cloern

Maintainer: Anthony Malkassian <anthonym@sfei.org>

---

ammFrac *Un-ionized ammonia*

Description

Calculates fraction of total ammonium in un-ionized form.

Usage

```
ammFrac(pH, t, S, pHscale = c('total', 'free'))
```

Arguments

pH	pH value
t	temperature, degrees Celsius
S	salinity, Practical Salinity Scale
pHscale	scale of pH measurement

Details

The stoichiometric dissociation constant K_a^* of the ammonium ion is estimated using the equations of Clegg and Whitfield (1995). There are separate equations for use with pH measurements made on the total and free scales. Equations should be valid in the range of 0 to 40 salinity and -2 to 40 C. Accuracy is probably better than 5% at all temperatures and salinities. The dissociation constant is then used to calculate the mole fraction from $1/(1 + 10^{-pH}/K_a^*)$. If input vectors are not of the same length, shorter ones will be recycled as necessary without warning.

Value

Fraction of total ammonium occurring as un-ionized ammonium.

Note

There are several approaches available to make these estimates, depending on the underlying theory, experimental data and fitting methods. See package **seacarb** for an algorithm based on the work of Millero (1995), which includes pressure (and thus depth) corrections.

References

Clegg, S. and Whitfield, M. (1995) A chemical model of seawater including dissolved ammonia and the stoichiometric dissociation constant of ammonia in estuarine water and seawater from -2 to 40 C. *Geochimica et Cosmochimica Acta* **59(12),** 2403–2421.

Millero, F. (1995) Thermodynamics of the carbon dioxide system in the oceans. *Geochimica et Cosmochimica Acta* **59(4),** 661–677.

Examples

```
## Examine different combinations of environmental variables:
ph = c(8, 8, 8, 7.8)
temp = c(10 , 25, 25, 25)
sal = c(0, 0, 35, 35)
round(ammFrac(ph, temp, sal), 4)  # 0.0183 0.0539 0.0556 0.0358
round(ammFrac(ph, temp, sal, 'free'), 4)  # 0.0183 0.0539 0.0440 0.0282
```

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.

Methods

No methods defined with class "DateTime" in the signature.

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.

Usage

```
decompTs(x, startyr, endyr, event = TRUE, type = c("mult", "add"))
```

Arguments

`x`	a monthly time series vector
`startyr`	the desired starting year for the analysis
`endyr`	the ending year
`event`	whether or not an "events" component should be determined
`type`	the kind of decomposition, either multiplicative ("mult") or additive ("add")

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. Aside from the long-term mean, each component of a multiplicative decomposition will average 1, whereas each component of an additive decomposition will average 0.

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
`grandmean`	constant series equal to the long-term mean
`annual`	annual mean series
`seasonal`	repeating seasonal component
`events`	optionally, the residual or "events" series

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 matrix time series,
# producing a list of decompositions
ans <- vector('list', ncol(sfbayChla))
names(ans) <- colnames(sfbayChla)
for(i in seq(along = names(ans))) {
  ans[[i]] <- decompTs(sfbayChla[, i])
}

# A quick plot for a time series decomposiiton
plot(ans[[7]], nc = 1, main = paste(names(ans)[7], "Chl-a decomposition"))
```

---

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

Arguments

ec	conductivity, mS/cm
t	temperature, Celsius
p	gauge pressure, decibar
R	conductivity ratio, dimensionless

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.

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 functions*

Description

Finds and rotates empirical orthogonal functions (EOFs, or principal components).

Usage

```
eof(x, n)
```

Arguments

x	a numeric object of class "matrix" or "data.frame", with no NA values
n	number of EOFs to retain for rotation

Details

This function is basically just a wrapper for `prcomp` followed by `promax` and is meant to facilitate routine examination of variability modes in collections of time series. For the usage intended in this package, each row of `x` usually will be an observation for a specific time or time period. Each column will be a specific location—the most common usage—but can also be seasons of the year (Jassby et al. 1999) or even a combination of seasons and depth layers (Jassby et al. 1990). First, eigenvalues are found by singular value decomposition of the *correlation* matrix. `eofNum` can be used to help choose `n`. The first `n` EOFs are then rotated using the `promax` method with power `m = 2` (Richman 1986). Hannachi et al. (2007) give a detailed discussion of this exploratory approach with emphasis on meteorological data.

Value

A list with the following members:

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

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 *Assess significance of eigenvalues*

Description

Plots the eigenvalue spectrum (scree plot) for the correlation matrix of a space-time field of observations. Useful for deciding how many empirical orthogonal functions (EOFs, or principal components) to retain for rotation.

Usage

```
eofNum(x, distr = c("normal", "lognormal"), n = nrow(x), reps = 10000)

ruleN(n, p, type = c("normal", "lognormal"), reps = 10000)
```

Arguments

x	an object of class "data.frame" or "matrix", with no missing values
distr, type	suspected distribution of data
n	number of independent observations in the sample, also known as effective sample size (n <= nrow(x))
reps	number of repetitions to use for rule N
p	number of columns in data

Details

Computes singular values of the correlation matrix for a space-time field represented as a data frame or matrix of observations x locations. The eigenvalue variances are plotted against eigenvalue number, 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. The significance of each eigenvalue is also estimated using *rule N* (Overland and Preisendorfer 1982), which repeatedly computes eigenvalues of the correlation matrix for an n x p matrix of a random variable and returns the 0.95 quantiles. The dist determines the distribution for the random variable. ruleN is not normally called directly.

Both North's rule-of-thumb and rule N as calculated here by default ignore any autocorrelation in the data and are therefore lenient in accepting the significance of eigenvalues. Their findings should therefore be taken as upper limits to the number of significant eigenvalues. 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. The default is to assume that the effective and actual sample sizes are the same.

Value

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

References

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.

Overland, J.E. and Preisendorfer, R.W. (1982). A significance test of principal components applied to a cyclone climatology. *Monthly Weather Review* **110,** 1–4.

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
chla1 <- aggregate(sfbayChla, 1, mean, na.rm = TRUE)  # average over each year
chla1 <- chla1[, 1:12]  # remove stations with missing years
eofNum(chla1, distr = 'lognormal', reps = 2000)
# These stations appear to act as one with respect to chlorophyll
# variability on the annual scale.
```

---

eofPlot *Plot results of an EOF analysis*

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	if TRUE, coefficients and amplitudes are multiplied by -1
ord	if TRUE, coefficients are 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.

The vertical guide lines at 0.2 and 0.35 are meant to reflect the Monte Carlo studies of Richman and Gong (1999): "For investigators who do not wish to reproduce a part of the Monte Carlo analysis for their research, the most general rule of thumb is that if there is a sufficient sample size, a correlation-based PC loading value of [plus or minus] 0.2-0.35 will likely suffice to separate the hyperplane out..." 'Sufficient sample size' in this case refers to `n > 50`.

Value

A plot of the EOF coefficients or amplitudes (and corresponding object of class "ggplot").

References

Richman, M. and Gong, X. (1999) Relationships between the definition of the hyperplane width to the fidelity of principal component loading patterns. *Journal of Climate* **12,** 1557–1576.

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.

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)")
```

`layOut` *Arrange a group of saved plots*

Description

Lays out plots on a grid, with flexible choice of the relative dimensions for each plot. Individual plots must be objects of class "ggplot".

Usage

```
layOut(...)
```

Arguments

`...`	A series of 3-element lists, each containing the name of a saved plot of class "ggplot", the sequence of grid numbers of the rows, and the sequence of grid numbers of the columns that will be occupied by that plot.

Details

The position of each plot is determined by the beginning row and column numbers. The relative size of each plot is determined by the sequence lengths of row and column numbers. The total grid size of the graph is determined automatically by the grid numbers used for individual plots. Some manual adjustment of the graphics window may be necessary to get proper aspect ratios and prevent text from overlapping.

Value

A graph containing all the given plots.

Examples

```
chl27 = sfbayChla[, 's27']
g1 <- plotTsTile(chl27, legend.title = 'Chl log-anomaly',
    square=FALSE)
g2 <- seasonTrend(chl27, plot = TRUE, legend = TRUE)
g3 <- plotSeason(chl27, num.era = 3,
    ylab = expression(paste('Chl-', italic(a), ', ', mu*g~L^{-1})))
## quartz("", 10, 6)  # e.g., in mac os x, or:
## grid.newpage()  # to re-use existing plot window
layOut(list(g1, 1:2, 1:6), list(g2, 1:2, 7:10), list(g3, 3:5, 1:8))
```

mannKen *Mann-Kendall 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", "pct", "tau"), order = FALSE)
```

Arguments

x	A time series vector or matrix.
plot	Should the trends be plotted when x is of class "mts"?
type	Type of trend to be plotted
order	Should the plotted trends be ordered by size?

Details

The Sen slope (alternately, Theil or Theil-Sen slope)—the median slope joining all pairs of observations—is expressed both by quantity per unit time and percent of the mean 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. Other results are used for further analysis by other functions.

If `plot = TRUE`, then either the Sen slope, the Sen slope as a percent of the mean, or Kendall's tau are plotted, along with an indication of *p*-value and fraction of missing slopes joining the first and last fifths of the data. Only the last two types make sense when the variables in `x` have different units.

Value

A list with the following members:

sen.slope	Sen slope.
sen.slope.pct	Sen slope as percent of mean.
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`.

Note

The `varS` calculation is based on `kensen` in the USGS library for S-PLUS (Slack et al. 2003).

References

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

Slack, J.R., Lorenz, D.L., and others (2003) *USGS library for S-PLUS for Windows.* Open-File Report 03-357, 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)
mannKen(y2, plot=TRUE, type='pct')
mannKen(y2, plot=TRUE, type='tau', order=TRUE)
```

---

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 annnualized 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

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.

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)
```

---

phenoAmp *Phenological amplitude*

Description

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

Usage

```
## S4 method for signature 'ts'
phenoAmp(x, mon.range = c(1, 12))

## S4 method for signature 'zoo'
phenoAmp(x, mon.range = c(1, 12))
```

Arguments

x	A monthly time series, or a class "zoo" object.
mon.range	A vector of two numbers specifying the month range to be considered.

Details

phenoAmp gives three measures of the amplitude of a seasonal cycle: the range, the range divided by the mean, and the standard deviation divided by the mean, i.e., the coefficient of variation.

These measures can be restricted to a subset of the year by giving the desired range of month 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, 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.

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

Value

A data frame.

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']

p2 <- phenoAmp(y)
p2
apply(p2, 2, mean, na.rm=TRUE)

phenoAmp(y, c(1, 6))
```

---

phenoAmp-methods *Methods for Function phenoAmp*

Description

Finds various measures of the amplitude of the annual cycle.

Methods

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

`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.

Usage

```
## S4 method for signature 'ts'
phenoPhase(x, mon.range = c(1, 12), ...)

## S4 method for signature 'zoo'
phenoPhase(x, mon.range = c(1, 12), out = c('date', 'doy', 'julian'), ...)
```

Arguments

x	A monthly time series, or a class "zoo" object.
mon.range	A vector of two numbers specifying the month 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 month (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 month 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, 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 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. In the case of monthly time series, the results are all given as decimal months 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.mon > 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

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

`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)

A. Jassby and A. Malkassian

See Also

decompTs, seasonTrend

Examples

```
chl27 <- sfbayChla[, 's27']
plotSeason(chl27, num.era = c(1978, 1988, 1998, 2008), ylab = 'Stn 27 Chl-a')
plotSeason(chl27, num.era = 3, same.plot = FALSE, ylab = 'Stn 27 Chl-a')
plotSeason(chl27, "by.month", ylab = 'Stn 27 Chl-a')
```

---

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, xlab, ylab, dot.size = 1, plot.order = colnames(x),
  strip.labels = colnames(x), ...)
```

Arguments

x	matrix or vector time series
xlab	optional x-axis label
ylab	optional y-axis label
dot.size	size of dots representing isolated data points
plot.order	column names of matrix time series, possibly in a different order
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` avoids both extremes, plotting 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".

See Also

`plotTsAnom`

Examples

```
### Chlorophyll at 6 stations in SF Bay
chl <- sfbayChla[, 1:6]
plotTs(chl, ylab = 'Chl-a', strip.labels = paste('Station',
  substring(colnames(chl), 2, 3)), ncol = 2)
```

---

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, ylab, plot.order = colnames(x),
  strip.labels = colnames(x), ...)
```

Arguments

x	matrix or vector time series
xlab	optional x-axis label
ylab	optional y-axis label
plot.order	column names of matrix time series, possibly in a different order
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".

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')
```

---

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("mean", "median"))
```

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").

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)
```

---

seaKen *Seasonal and Regional Kendall test*

Description

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

Usage

```
seaKen(x)

seaRoll(x, w = 5, rule = 2, plot = FALSE, ylab = NULL, legend = FALSE)
```

Arguments

x	A time series vector.
w	The window width for "rolling" estimates of slope.
rule	The rule number for excluding windows with excessive missing data.
plot	Indicates if a plot should be drawn.
ylab	An optional y-axis label.
legend	Indicates if the legend is drawn.

Details

The Seasonal Kendall tests were introduced by Hirsch et al. (1982) and are further described by Helsel and Hirsch (2002). The *p*-values provided here are the raw values, not the ones corrected for serial correlation among seasons. In any case, the raw values are recommended for series lengths less than 10 years.

The function `seaRoll` applies `seaKen` to rolling time windows of width `w`. A minimum `w` of five years is required. The only rules currently implemented are: (1) ignore missing data; and (2) report a result only if more than half the seasons are each missing less than half the possible comparisons between the first and last 20% of the years (Schertz et al. [1991] discuss these and related decisions about missing data).

If `plot = TRUE` in the latter function, a point plot will be drawn with the Sen slope plotted at the leading year of the trend window. Filled circles indicate p-value < 0.01, and a legend will be drawn if requested.

Both functions 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

`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

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

References

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/

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.

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

`mts2ts`, `trendHomog`

Examples

```
chl27 <- sfbayChla[, 's27']
seaKen(chl27)
seaRoll(chl27)
seaRoll(chl27, plot = TRUE, legend = TRUE)
chl <-sfbayChla
seaKen(mts2ts(chl))  # too much missing data
seaKen(mts2ts(chl, seas = 2:4))  # better when just Feb-Apr, spring
    # bloom period, but last 4 stations still missing too much data.
seaKen(mts2ts(chl[, 1:12], 2:4))  # more reliable result
```

seasonTrend *Determine seasonal trends*

Description

Finds the trend for each season of a time series (matrix or vector) and indicates statistical significance.

Usage

```
seasonTrend(x, first, last, type = c("slope", "slope.pct"),
    method = c("mk", "lin"), plot = FALSE, xlab = NULL,
    ylab = NULL, miss = FALSE, legend = FALSE, ...)
```

Arguments

x	Time series vector, or time series matrix with column names.
first	First year of desired time interval. Will be adjusted if less than start of series.
last	Last year of desired time interval. Will be adjusted if more than end of series.
type	Type of trend, either in original units per year, or percent per year.
method	'mk' for Mann-Kendall (Theil-Sen slope), and 'lin' for linear regression (linear slope).
plot	If TRUE, a plot is generated; otherwise the results are listed.
xlab	Optional x-axis label.
ylab	Optional y-axis label.
miss	If TRUE, trends with insufficient data.
legend	If TRUE, a legend is included.
...	Further parameters to pass to the plotting function.

Details

The slope estimate and its significance are calculated for each season and time series. If `type = "slope.pct"`, the slopes are multiplied by 100 and divided by the overall mean (not the median, which can be zero even in cases where a trend estimate is useful). If `method = 'mk'`. the Theil-Sen slope is calculated with the Mann-Kendall test of significance. Otherwise, linear regression is used to determine the slope and significance.

If `plot = TRUE`, each time series is represented by a box plot showing the trend for each season. The fill colour of the box indicates whether the trend is significant or not (the legend is optional). When `method = 'mk'`, the proportion of slopes joining the first and last fifths of the data is calculated. If this value is 0.5 or more, the corresponding trends can be omitted by setting `miss = TRUE`; the trend results may not be a good representation of the entire period and a different time window should be considered.

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_x"` to allow the x-axis to change for each time series.

Value

A data frame with the following fields:

`trend`	Theil-Sen slope in original units per year, or percent per year.
`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.
`season`	Season number.
`tsname`	Name of time series.

See Also

`mannKen`, `plotSeason`, `facet_wrap`

Examples

```
x <- sfbayChla
seasonTrend(x, first=1978, last=2009, ncol = 4, plot = TRUE, legend = TRUE)
seasonTrend(x, type = 'slope.pct')
```

`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.

Usage

```
sfbay
```

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 R variable names. `sfbayChla` was constructed from the entire downloaded sfbay dataset and encompasses the period 1969–2009.

Source

Downloaded from http://sfbay.wr.usgs.gov/access/wqdata 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).

Usage

```
trendHomog(x)
```

Arguments

x	A vector time series.

Value

chisq.trend	"Trend" chi-square.
chisq.homog	"Homogeneous" chi-square.
p.value	For null hypothesis that trends are homogeneous.

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)

monthCor(x)
```

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.

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.

Usage

```
## S4 method for signature 'WqData'
tsMake(object, focus, layer,
    type = c("ts.mon", "zoo"), qprob = NULL)
```

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

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

utilities *Miscellaneous utility functions*

Description

A variety of small utilities used in other functions.

`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.

Usage

```
date2decyear(w)

decyear2date(x)

layerMean(d)

leapYear(x)

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

monthNum(y)

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

years(y)
```

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.

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)
```

---

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".

References

International Organization for Standardization (2004) ISO 8601. Data elements and interchange formats - Information interchange - Representation of dates and times. http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html

See Also

as.Date, strptime, WqData-class

Examples

```
# 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")
```

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", respectively.

Slots

.Data: Object of class "data.frame" as described above

Automatically created slots:

names: Object of class "character" containing the names of the original data.frame.

row.names: Object of class "data.frameRowLabels" containing the row names of the original data.frame.

.S3Class: Object of class "character" describing which S3 class is extended by this class.

Extends

Class "data.frame", directly. Class "list", by class "data.frame", distance 2. Class "oldClass", by class "data.frame", distance 2. Class "vector", by class "data.frame", distance 3.

Methods

[signature(x = "WqData"):

x[i,]

Extracts a subset of the data, based on either row numbers or a logical expression involving the columns (time, site, depth, variable, value). The returned value is also a WqData object.

plot signature(x = "WqData"):

plot(x, vars, num.col = NULL)

Multiple plots, each containing boxplots of data at each site for the same variable. An argument vars can be given specifying a character string of variables to be plotted. Otherwise, only the first 10 variables will be plotted if the number of variables is more than 10. The number of columns plotted is determined automatically unless num.col is given an integer value.

summary `signature(x = "WqData")`:

summary(x)

Date range, number of observations for each combination of site and variable, and quartiles of variable values.

tsMake `signature(x = "WqData")`: see `tsMake,WqData-method`

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
```

`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.

Slots

`.S3Class`**:** Object of class `"character"`

Extends

Class `"oldClass"`, directly.

See Also

`phenoAmp`, `phenoPhase`

Examples

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

Index