Package 'hydroTSM'

May 8, 2026

Type Package

Title Time Series Management and Analysis for Hydrological Modelling

Version 0.8-6

Maintainer Mauricio Zambrano-Bigiarini <mzb.devel@gmail.com>

Description
S3 functions for management, analysis, interpolation and plotting of time series used in hydrology and related environmental sciences. In particular, this package is highly oriented to hydrological modelling tasks. The focus of this package has been put in providing a collection of tools useful for the daily work of hydrologists (although an effort was made to optimise each function as much as possible, functionality has had priority over speed). Bugs / comments / questions / collaboration of any kind are very welcomed, and in particular, datasets that can be included in this package for academic purposes.

License GPL (>= 2)

Depends R (>= 3.5.0), zoo (>= 1.7-2)

Imports xts (>= 0.9-7), e1071, stats, utils, methods, graphics, grDevices, lattice, classInt, timechange

Suggests knitr, rmarkdown, gridExtra

VignetteBuilder knitr

URL https://hzambran.github.io/hydroTSM/, https://github.com/hzambran/hydroTSM, https://CRAN.R-project.org/package=hydroTSM

MailingList https://stat.ethz.ch/mailman/listinfo/r-sig-ecology

BugReports https://github.com/hzambran/hydroTSM/issues

LazyLoad yes

NeedsCompilation no

Repository CRAN

Author Mauricio Zambrano-Bigiarini [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-9536-643X>), Joschka Thurner [ctb]

Date/Publication 2026-04-28 07:20:03 UTC

Contents

`hydroTSM-package` *Management, analysis, and plot of hydrological time series, with focus on hydrological modelling*

Description

S3 functions for management, analysis, interpolation and plotting of time series used in hydrology and related environmental sciences. In particular, this package is highly oriented to hydrological modelling tasks. The focus of this package has been put in providing a collection of tools useful for the daily work of hydrologists (although an effort was made to optimise each function as much as possible, functionality has had priority over speed). Bugs / comments / questions / collaboration of any kind are very welcomed, and in particular, datasets that can be included in this package for academic purposes.

Details

Package:	hydroTSM
Type:	Package
Version:	0.8-0
Date:	2026-04-27
License:	GPL >= 2
LazyLoad:	yes
Packaged:	Mon Apr 27 08:57:54 -04 2026 ; MZB
BuiltUnder:	R version 4.6.0 (2026-04-24) – "Because it was There" ; aarch64-apple-darwin23

—

Datasets:

`Cauquenes7336001` Hydrometeorological time series for "Cauquenes en El Arrayan" catchment.
`EbroPPtsMonthly` Ebro Monthly Precipitation Time Series.
`KarameaAtGorgeQts` Karamea at Gorge, time series of hourly streamflows.
`MaquehueTemuco` San Martino, ts of daily precipitation.
`OcaEnOnaQts` Oca in "Ona" (Q0931), time series of daily streamflows
`SanMartinoPPts` San Martino, ts of daily precipitation.

Temporal aggregation:

annualfunction single representative annual value of a zoo object.
weeklyfunction single representative weekly value of a zoo object.
monthlyfunction single representative monthly value of a zoo object.
seasonalfunction representative values of each weather season of a zoo object.
daily2annual Aggregation from daily to annual
subdaily2annual Aggregation from subdaily to annual.
monthly2annual Aggregation from monthly to annual.
daily2monthly Aggregation from daily to monthly.
subdaily2monthly Aggregation from subdaily to monthly.
daily2weekly Aggregation from daily to weekly.
dm2seasonal Aggregation from daily or monthly to seasonal.
subdaily2seasonal Aggregation from subdaily to seasonal.
subdaily2daily Aggregation from subdaily to daily.
subdaily2weekly Aggregation from subdaily to weekly.
subhourly2hourly Aggregation from subhourly to hourly.
subhourly2nminutesAggregation from subhourly to n-minutes.

Temporal manipulation:

dip Days in period.
diy Days in year.
hip Hours in period.
mip Months in period.
yip Years in period.
izoo2rzoo Irregular zoo object to regular zoo objectl.
time2season Time to weather season.
vector2zoo Numeric and date/times to zoo object.

Hydrological functions:

baseflow Baseflow computation.
climograph Climograph
dwdays Dry and wet days.
fdc Flow duration curve.
fdcu Flow duration curve with uncertainty bounds.
hydroplot Exploratory figure for hydrological time series.
sname2plot Hydrological time series plotting and extraction.
plot_pq Plot precipitation and streamflow time series in the same figure.
si Seasonality Index for precipitation.

sname2ts Station Name -> Time Series.
zoo2RHtest Zoo object -> RHTest.

Miscelaneous functions:

calendarHeatmap Calendar heat map.
cmv Counting missing values.
drawxaxis Draw a temporal horizontal axis.
dwi Days with information.
extract Extract a subset of a zoo object.
hydropairs Visual correlation matrix.
infillxy Infills NA values.
istdx Inverse standarization.
ma Moving average.
matrixplot 2D color matrix.
rm1stchar Remove first character.
sfreq Sampling frequency.
smry Improved summary function.
stdx Standarization.

Author(s)

Mauricio Zambrano-Bigiarini

Maintainer: Mauricio Zambrano-Bigiarini <mzb.devel@gmail>

See Also

https://github.com/hzambran/hydroGOF.
https://github.com/hzambran/hydroPSO.
https://gitlab.com/rmarinao/hydroMOPSO.

Examples

```
## Loading the monthly time series (10 years) of precipitation for the Ebro River basin.
data(EbroPPtsMonthly)

#######
## Ex1) Graphical correlation among the ts of monthly precipitation of the first
##      3 stations in 'EbroPPtsMonthly' (its first column stores the dates).
hydropairs(EbroPPtsMonthly[,2:4])

#######
## Ex2) Annual values of precipitation at the station "P9001"
sname2ts(EbroPPtsMonthly, sname="P9001", dates=1, var.type="Precipitation",
```

```
        tstep.out="annual")

#######
## Ex3) Monthly and annual plots
sname2plot(EbroPPtsMonthly, sname="P9001", var.type="Precipitation", pfreq="ma")


#######
## Ex5)  Mean monthly values of streamflows
## Loading daily streamflows (3 years) at the station
## Oca en Ona (Ebro River basin, Spain)
data(OcaEnOnaQts)
monthlyfunction(OcaEnOnaQts, FUN=mean, na.rm=TRUE)
```

(sub)daily2annual *(sub)Daily/Monthly -> Annual*

Description

Generic function for transforming a (sub)DAILY/MONTHLY (weekly and quarterly) regular time series into an ANNUAL one.

Usage

```
daily2annual(x, ...)
subdaily2annual(x, ...)
monthly2annual(x, ...)

## Default S3 method:
daily2annual(x, FUN, na.rm=TRUE, na.rm.max=0,
        out.fmt=c("%Y", "%Y-%m-%d"), start.month=1, ...)

## S3 method for class 'zoo'
daily2annual(x, FUN, na.rm=TRUE, na.rm.max=0,
        out.fmt=c("%Y", "%Y-%m-%d"), start.month=1, ...)

## S3 method for class 'data.frame'
daily2annual(x, FUN, na.rm=TRUE, na.rm.max=0,
        out.fmt=c("%Y", "%Y-%m-%d"), start.month=1, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", verbose = TRUE, ...)

## S3 method for class 'matrix'
daily2annual(x, FUN, na.rm = TRUE, na.rm.max=0,
        out.fmt=c("%Y", "%Y-%m-%d"), start.month=1,
        dates=1, date.fmt = "%Y-%m-%d", out.type = "data.frame",
        verbose = TRUE,  ...)
```

Arguments

x	zoo, data.frame or matrix object, with (sub)daily/monthly time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represents the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that have to be applied for aggregating from (sub)daily/monthly into annual time step (e.g., for precipitation FUN=sum and for temperature and streamflows ts FUN=mean). FUN MUST accept the na.rm argument, because na.rm is passed to FUN. When FUN=max or FUN=min the date(time) where the maximum/minimum value actually occurs is returned in the output object, otherwise, a generic 1st of january for each year is returned.
na.rm	Logical. Should missing values be removed? -) TRUE : the annual values are computed only for years with a percentage of missing values less than na.rm.max -) FALSE: if there is AT LEAST one NA within a year, the corresponing annual values in the output object will be NA.
na.rm.max	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each year to keep the yearly aggregated value in the output object of this function. In other words, if the percentage of missing values in a given year is larger than na.rm.max the corresponding annual value will be NA.
out.fmt	Character indicating the date format for the output zoo object. See format in as.Date. Possible values are: -) %Y : only the year will be used for the time. Default option. (e.g., "1961" "1962"...) -) %Y-%m-%d: a complete date format will be used for the time. (e.g., "1961-01-01" "1962-01-01"...). See Details.
start.month	numeric in [1,..,12], representing the starting month (1:Jan, ..., 12:Dec) to be used in the computation of annual values. By default start.month=1.
dates	numeric, factor or Date object indicating how to obtain the dates for corresponding to each gauging station If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days on it be equal to the number of element in x
date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	Character that defines the desired type of output. Valid values are: -) data.frame: a data.frame, with as many columns as stations are included in

	x, and row names indicating the Year -) db : a data.frame, with 3 columns will be produced. The first column (StationID) will store the ID of the station The second column (Year) will store the year, The third column (Value) will contain the annual value corresponding to the two previous columns.
verbose	logical; if TRUE, progress messages are printed
...	arguments additional to na.rm passed to FUN.

Value

When FUN!=max and FUN!=min the output is a zoo object with annual time frequency, where the time attribute has the format defined in out.fmt.
When FUN!=max and FUN!=min and out.fmt="%Y-%m-%d" the time attribute of the output zoo object will use the 1st of January of each year to create a full Date object from the corresponding year of each element of the output object (e.g., fi the year is 2022, the time attribute will be 2022-01-01). The only exception occurrs when FUN=max or FUN=min, where the time attribute of each element will correspond to the actual date where the annual maximum/minimum occurs (which is very useful for identifying the date of the annual maximum or the annual minimum of a time series).

When FUN=max or FUN=min and x is a single time series, the output is a zoo object with annual time frequency, where the time attribute has the same class than time(x), and the date(time) value corresponds to the date(time) where the maximum/minimum value actually occurs.
When FUN=max or FUN=min and x has two or more time series, the output is a list object where each element has an annual time frequency. The time attribute of each list element has the same class than time(x), and the date(time) value of each list element corresponds to the date(time) where the maximum/minimum value actually occurs.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

subhourly2hourly, daily2monthly, monthly2annual, hydroplot, annualfunction, vector2zoo, as.Date

Examples

```
######################
## Ex1: Computation of annual values, removing any missing value in 'x'

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three months (Jan/1921 - Mar/1921)
x <- window(x, end="1921-03-31")
```

```
## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Agreggating from Daily to Annual, removing any missing value in 'x'
( a <- daily2annual(x, FUN=sum, na.rm=TRUE) )

######################
## Ex2: Compuation of annual values only when the percentage of NAs in each
#       year is lower than a user-defined percentage (10% in this example).

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three months (Jan/1921 - Mar/1921)
x <- window(x, end="1921-12-31")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Daily to annual, only for months with less than 10% of missing values
( a2 <- daily2annual(x, FUN=sum, na.rm=TRUE, na.rm.max=0.1) )

# Verifying that the second and third month of 'x' had 10% or more of missing values
cmv(x, tscale="annual")


######################
## Ex3: Getting the annual maximum value, including the date where this annual
##      maximum actually occurs
daily2annual(x, FUN=max)


######################
## Ex4: Monthly to Annual, allowing a maximum of 20% of missing values in each month
m <- daily2monthly(x, FUN=sum, na.rm=TRUE, na.rm.max=0.2)
monthly2annual(m, FUN=sum, na.rm=TRUE)


######################
## Ex5: Loading the time series of HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts
```

```
# Sub-daily to monthly ts
subdaily2annual(x, FUN=mean, na.rm=TRUE)

############
## Ex6: Loading the monthly time series of precipitation within the Ebro River basin
data(EbroPPtsMonthly)

# computing the annual values for the first 10 gauging stations in 'EbroPPtsMonthly'
a <- monthly2annual(EbroPPtsMonthly[,1:11], FUN=sum, dates=1)

# same as before, but with a nicer format of years
a <- monthly2annual(EbroPPtsMonthly[,1:11], FUN=sum, dates=1, out.fmt="%Y")
```

(sub)daily2monthly *(sub)Daily -> Monthly*

Description

Generic function for transforming a DAILY (sub-daily or weekly) regular time series into a MONTHLY one

Usage

```
daily2monthly(x, ...)
subdaily2monthly(x, ...)

## Default S3 method:
daily2monthly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'zoo'
daily2monthly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'data.frame'
daily2monthly(x, FUN, na.rm=TRUE, na.rm.max=0, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", out.fmt="numeric",
        verbose=TRUE, ...)

## S3 method for class 'matrix'
daily2monthly(x, FUN, na.rm=TRUE, na.rm.max=0, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", out.fmt="numeric",
        verbose=TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with (sub)daily time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represents the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).

FUN	Function that have to be applied for transforming from daily to monthly time step (e.g., for precipitation FUN=sum and for temperature and streamflow ts FUN=mean). FUN MUST accept the na.rm argument, because na.rm is passed to FUN.
na.rm	Logical. Should missing values be removed? -) TRUE : the monthly values are computed only for months with a percentage of missing values less than na.rm.max -) FALSE: if there is AT LEAST one NA within a month, the corresponing monthly values in the output object will be NA.
na.rm.max	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each month to keep the monthly aggregated value in the output object of this function. In other words, if the percentage of missing values in a given month is larger than na.rm.max the corresponding monthly value will be NA.
dates	numeric, factor or Date object indicating how to obtain the dates for each gauging station If dates is a number (default), it indicates the index of the column in *x* that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days on it be equal to the number of elements in x
date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	Character that defines the desired type of output. Valid values are: -) data.frame: a data.frame, with as many columns as stations are included in x, and row names indicating the month and year for each value. -) db : a data.frame, with 4 columns will be produced. The first column (StationID) stores the ID of the station, The second column (Year) stores the year The third column (Month) stores the Month The fourth column (Value) stores the numerical values corresponding to the values specified in the three previous columns.
out.fmt	OPTIONAL. Only used when x is a matrix or data.frame object /cr character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo.
verbose	logical; if TRUE, progress messages are printed
...	arguments additional to na.rm passed to FUN.

Value

a zoo object with monthly time frequency

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

cmv, subhourly2hourly, daily2annual, subdaily2daily, monthlyfunction, hydroplot, vector2zoo, izoo2rzoo, as.Date

Examples

```
######################
## Ex1: Computation of monthly values, removing any missing value in 'x'

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three months (Jan/1921 - Mar/1921)
x <- window(x, end="1921-03-31")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Agreggating from Daily to Monthly, removing any missing value in 'x'
m <- daily2monthly(x, FUN=sum, na.rm=TRUE)

## Agreggating from Daily to Monthly, removing any missing value in 'x', but now
## allowing a maximum of 20% of missing values in each month
m <- daily2monthly(x, FUN=sum, na.rm=TRUE, na.rm.max=0.2)

######################
## Ex2: Computation of monthly values only when the percentage of NAs in each
#       month is lower than a user-defined percentage (10% in this example).

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three months (Jan/1921 - Mar/1921)
x <- window(x, end="1921-03-31")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Daily to monthly, only for months with less than 10% of missing values
m2 <- daily2monthly(x, FUN=sum, na.rm=TRUE, na.rm.max=0.1)

# Verifying that the second and third month of 'x' had 10% or more of missing values
```

```
cmv(x, tscale="month")

######################
## Ex3: Loading the HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

# Sub-daily to monthly ts
subdaily2monthly(x, FUN=mean, na.rm=TRUE)
```

---

annualfunction *Annual Function*

Description

Generic function for obtaining a SINGLE annual value of a zoo object, by applying any R function to ALL the values in x belonging to the same year, and then applying the same function to ALL the previously computed annual values (e.g., for computing the average annual precipitation or the mean annual streamflow of a long-term time series).

Usage

```
annualfunction(x, FUN, na.rm = TRUE, ...)

## Default S3 method:
annualfunction(x, FUN, na.rm = TRUE, ...)

## S3 method for class 'zoo'
annualfunction(x, FUN, na.rm = TRUE, ...)

## S3 method for class 'data.frame'
annualfunction(x, FUN, na.rm = TRUE, dates=1, date.fmt = "%Y-%m-%d",
        verbose = TRUE, ...)

## S3 method for class 'matrix'
annualfunction(x, FUN, na.rm = TRUE, dates=1, date.fmt = "%Y-%m-%d",
        verbose = TRUE, ...)
```

Arguments

x	zoo, xts, data.frame or matrix object, with daily/monthly/annual time series. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that will be used to compute the final annual value (e.g., FUN may be some of mean, sum, max, min, sd) .

na.rm	Logical. Should missing values be removed?. -) TRUE : the annual values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA within a year, the resulting annual value will be NA
dates	numeric, factor or Date object indicating how to obtain the dates corresponding to each gauging station. If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it have to be converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of elements in x
date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
verbose	Logical; if TRUE, progress messages are printed.
...	further arguments passed to or from other methods.

Value

When x is a time series, a single annual value is returned.
For a data frame, a named vector with the appropriate method being applied column by column.

Note

FUN is first applied to all the values of x belonging to the same year and then it is applied to all the previously computed annual values to get the final result. Its result will depend on the sampling frequency of x and the type of function provided by FUN (**special attention have to be put when** FUN=sum)

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

monthlyfunction, daily2annual, monthly2annual, yip

Examples

```
## Loading the SanMartino daily precipitation data (1921-1990)
data(SanMartinoPPts)
x <- SanMartinoPPts

# Amount of years in 'x' (needed for computing the average)
nyears <- length( seq(from=time(x[1]), to=time(x[length(x)]), by="years") )

## Average annual precipitation for the 70 years period.
```

```
# It is necessary to divide by the amount of years to obtain the average annual value,
# otherwise it will give the total precipitation for all the 70 years.
annualfunction(x, FUN=sum, na.rm=TRUE) / nyears


#####################
### verification ####
# Daily to annual
a <- daily2annual(x, FUN=sum, na.rm=TRUE)

# Mean annual value
mean(a)

#############################
#############################
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)
x <- EbroPPtsMonthly

## Dates of 'x'
dates <- as.Date(x[,1])


## Computation of the average annual precipitation
## Not run:

## Transforming 'x' into a zoo object
z <- zoo( x[, 2:ncol(x)], dates)

# Amount of years in 'x' (needed for computing the average)
nyears <- yip(from=start(z), to=end(z), out.type="nmbr" )

## Average annual precipitation, for the first 5 stations in 'x'
annualfunction(z[ ,1:5], FUN=sum)/nyears


## End(Not run)
```

`baseflow` *Baseflow*

Description

Given a complete (without missing values) series of streamflow, this function computes the baseflow using the filter proposed by Arnold and Allen (1999).

Usage

```
baseflow(x, ...)
```

```
## S3 method for class 'zoo'
baseflow(x, beta=0.925, from=start(x), to=end(x),  n.pass=3L,
         pass.start=c("forward", "backward"), date.fmt, tz,
         na.fill=c("none", "linear", "spline"), out.type=c("last", "all"),
         plot=TRUE, xcol="black", bfcol=c("blue", "darkcyan", "darkorange3"),
         pch=15, cex=0.3, ...)
```

Arguments

x	zoo or numeric object with streamflow records. The suggested time frequency should be hourly or daily, but the algorithm will work with any time frequency.
beta	numeric representing the filter parameter. Default value is 0.925 as recommended by Arnold and Allen (1999)
from	Character indicating the starting date for subsetting x. It has to be in the format indicated by date.fmt. The default value corresponds to the date of the first element of x.
to	Character indicating the ending date for subsetting x. It has to be in the format indicated by date.fmt. The default value corresponds to the date of the last element of x.
n.pass	Integer indicating the number times the filter will be passed over the streamflow data stored in x. Default is 3.
pass.start	Character indicating the direction of the first pass of the filter. Subsequent passes alternate direction (e.g., forward-backward). Valid values are: -) forward => the first pass of the filter is applied forward -) backward => the first pass of the filter is applied backward
date.fmt	character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See 'Details' section in strptime. By default, date.fmt is missing, and it is automatically set to %Y-%m-%d when time(x) is Date object, and set to %Y-%m-%d %H:%M:%S when x is a sub-daily zoo object.
tz	character, with the specification of the time zone used for from, to. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(x). This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for from and to. It should be used with caution, being well aware of the time zone of the data. See examples.
na.fill	Character indicating how to fill any NA present in x. Valid values are: -) remove => NAs are not plotted -) linear => NAs are removed by linear interpolation, using na.approx -) spline => NAs are removed by spline interpolation, using na.spline

out.type	Character indicating the type of result that is given by this function. Valid values are: -) last => only the baseflow computed after the third pass of the filter is returned. -) all => the 3 baseflows computed after each pass of the filter are returned in a matrix or zoo object.
plot	logical. Indicates if the baseflow should be plotted or not. If plotted, the original x values are plotted as well.
xcol	character, representing the color to be used for ploting the streamflow time series. Only used when plot=TRUE.
bfcol	character of lenght 3, representing the color(s) to be used for ploting the baseflow time series. The first, second and third element are used to represent the baseflow after the third, second and first pass of the filter, respectively. Only used when plot=TRUE.
pch	numeric, representing the symbols used for ploting the streamflow time series (both, the original series and the baseflow). Only used when plot=TRUE.
cex	a numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex"). See plot.default. Only used when plot=TRUE.
...	further arguments passed to or from other methods. Not used yet.

Details

Although most procedures to separate baseflow from total streamflow are based on physical reasoning, some elements of all separation techniques are subjective.

The digital filter technique (Nathan and McMahon, 1990) implemented in this function was originally proposed by Lyne and Hollick (1979) for signal analysis and processing. Although this technique has no true physical meaning, it is objective and reproducible.

The equation of the filter is:

q(t) = Beta*q(t-1) + [(1+Beta)/2]*[Q(t) - Q(t-1)]

where q(t) is the filtered surface runoff (quick response) at the time step t (one day), Q is the original streamflow, and Beta is the filter parameter (Beta=0.925). The value Beta=0.925 was obtained by Nathan and McMahon (1990) and Arnold et al. (1995) to give realistic results when compared to manual separation techniques.

Baseflow b(t) is then computed as:

b(t) = Q(t) - q(t)

The filter can be passed over the streamflow data three times (forward, backward, and forward), depending on the user' selected estimates of baseflow from pilot studies. In general, each pass will result in less baseflow as a percentage of total streamflow.

Value

If out.type="last"
(default value), only the baseflow computed after the third pass of the filter is returned.

If `out.type="all"`
: the 3 baseflows computed after each pass of the filter are returned in a matrix or zoo object.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail> ; Hector Garces-Figueroa

References

Arnold, J. G., Allen, P. M., Muttiah, R., Bernhardt, G. (1995). Automated base flow separation and recession analysis techniques. Groundwater, 33(6), 1010–1018. doi:10.1111/j.1745-6584.1995.tb00046.x.

Arnold, J. G., Allen, P. M. (1999). Automated methods for estimating baseflow and ground water recharge from streamflow records. JAWRA Journal of the American Water Resources Association, 35(2), 411–424. doi:10.1111/j.1752-1688.1999.tb03599.x.

Lyne, V., Hollick, M. (1979). Stochastic time-variable rainfall-runoff modelling. Proceedings of the Hydrology and Water Resources Symposium, Perth, 10–12 September. Institution of Engineers National Conference Publication, No. 79/10, 89–92.

Nathan, R. J., & McMahon, T. A. (1990). Evaluation of automated techniques for base flow and recession analyses. Water resources research, 26(7), 1465–1473. doi:10.1029/WR026i007p01465.

Ladson, A. R., Brown, R., Neal, B., & Nathan, R. (2013). A standard approach to baseflow separation using the Lyne and Hollick filter. Australasian Journal of Water Resources, 17(1), 25-34. doi:10.7158/13241583.2013.11465417.

See Also

`plot_pq`, `hydroplot`, `fdc`, `fdcu`

Examples

```
######################
## Ex1: Computing and plotting the baseflows for the full time period
##      of a given time series of streamflows.

## First, we load the daily Q time series for the Cauquenes en
## El Arrayan catchment, where Q, [m3/s] are stored in the sixth column.
data(Cauquenes7336001)
q <- Cauquenes7336001[, 6]

## Computing the daily baseflow for the full time period
#baseflow(q) # it can not run due to NA values in 'x'

# filling the NA values using spline interpolation
baseflow(q, na.fill="spline")

## Computing and plotting the daily baseflow for the full time period
baseflow(q, na.fill="spline", plot=TRUE)
```

```
######################
## Ex2: Computing and plotting the daily baseflow only for a
##      specific time period, from April to December 2000.
baseflow(q, na.fill="spline", from="2000-04-01", to="2000-12-31")


######################
## Ex3: Computing and plotting the three daily baseflows (one for each pass
##      of the filter) only for a specific time period, from April to December
##      2000.
baseflow(q, na.fill="spline", from="2000-04-01", to="2000-12-31",
         out.type="all", plot=TRUE)
```

`calendarHeatmap` *Calendar heatmap*

Description

Function to plot a heatmap of a daily zoo object with a calendar shape

Usage

```
calendarHeatmap(x, ...)

## S3 method for class 'zoo'
calendarHeatmap(x, from, to, date.fmt="%Y-%m-%d",
                main="Calendar Heat Map",
                col=colorRampPalette(c("red", "orange", "yellow", "white",
                    "lightblue2", "deepskyblue", "blue3"),  space = "Lab")(8),
                cuts, cuts.dec=0, cuts.labels,
                cuts.style=c("fisher", "equal", "pretty", "fixed", "sd",
                             "quantile", "kmeans", "bclust", "mzb"),
                legend.title="", legend.fontsize=15,
                do.png=FALSE, png.fname="mypng.png", png.width=1500,
                png.height=900, png.pointsize=12, png.res=90,
                do.pdf=FALSE, pdf.fname="mypdf.pdf", pdf.width=11,
                pdf.height=8.5, pdf.pointsize=12, ...)
```

Arguments

`x`	daily zoo object to be plotted. Its maximum amount of daily data should be less than 6 years or otherwise it will not be plotted.
`from`	Character indicating the starting date for subsetting `x`. It has to be in the format indicated by `date.fmt`.
`to`	Character indicating the ending date for subsetting `x`. It has to be in the format indicated by `date.fmt`.
`date.fmt`	character indicating the format in which the dates are stored in `from` and `to`, e.g. `%Y-%m-%d`. See 'Details' section in `strptime`.

main	character, Main chart title.
col	A color palette, i.e. a vector of n contiguous colors generated by functions like rainbow, heat.colors, topo.colors, bpy.colors or one of your own making, perhaps using colorRampPalette. If none is provided, a color ramp with 8 colours is created using colorRampPalette.
cuts	Numeric, indicating the values used to divide the range of x in the legend of colours. If not provided, it is automatically selected as a function of lenght(col).
cuts.dec	Number of decimal places used to present the numbers in the legend of colours.
cuts.labels	Character indicating the label to be used in the colour legend for each one of the values defined by cuts. If not provided, as.character(cuts) is used.
cuts.style	discarded because takes too much time or not alway provide the required number of classes: "dpih", "headtails", "hclust", "jenks".
legend.title	text to be displayed above the legend of colours.
legend.fontsize	size of text (in points) used in the legend of colours.
do.png	Do you want to write the figure as a .png file? logical, set TRUE to save the image.
png.fname	character, indicating the name of the file (possibly with a meaninful file extension) that will be used to write the output file.
png.width	numeric, the width of the device.
png.height	numeric, the height of the device.
png.pointsize	integer, the default pointsize of plotted text, interpreted as big points (1/72 inch) at res ppi.
png.res	integer, the nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default, and to convert points to pixels.
do.pdf	Do you want to write the figure as a .pdf file? logical, set TRUE to save the image.
pdf.fname	character, indicating the name of the file (possibly with a significant extension) that will be used to write the output file.
pdf.width	numeric, the width of the device.
pdf.height	numeric, the height of the device.
pdf.pointsize	integer, the default point size to be used. Strictly speaking, in bp, that is 1/72 of an inch, but approximately in points. Defaults to 12.
...	further arguments passed to functions or from other methods. Not used yet.

Details

The original function calendarHeat was developed by Paul Bleicher, as an R version of a graphic from http://stat-computing.org/dataexpo/2009/posters/wicklin-allison.pdf (not available any longer). The original function was made available online in 2009, but then it was removed. Now it is available at https://github.com/tavisrudd/r_users_group_1/blob/master/calendarHeat.R. The

original function has "Copyright 2009 Humedica", but it was distributed under the GPL-2 licence, as well as this new version of the function.

This slighly modified verison of the function is also distributed under the GPL-2 licence, and teh main changes with respect to the original function are:

1) uses a zoo object instead of a numeric and character vector, for values and dates, respectively.
2) it allows a customisation of the mian title of the output figure.
3) it allows a customisation of the color palette.
4) it uses a categorical legend instead of a continuos one.
5) it is named `calendarHeatmap` instead of `calendarHeat`.

Value

The output of this function is a lattice figure with the calendar heatmap

Note

The maximum amount of years to be plotted is six (6).

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

`colorRampPalette`, `levelplot`

Examples

```
############
# EXAMPLE 1: basic plotting of a calendar heatmap
############

# Loading daily streamflow data for Karamet at Gorges.
data(KarameaAtGorgeQts)

x <- KarameaAtGorgeQts
x <- subdaily2daily(x, FUN=mean)

# Temporal subsetting for a amaximum of 6 years
x <- window(x, start="1980-01-01", end="1985-12-31")

# Calendar Heatmap, 8 colours and cuts defined using Fisher method
calendarHeatmap(x)
```

Cauquenes7336001 *Hydrometeorological time series for "Cauquenes en El Arrayan" catchment*

Description

Daily time series of precipitation, maximum air temperature, minimum air temperature, potential evapotranspiration, and observed streamflows for the catchment draining into the 'Cauquenes en El Arrayan' streamflow station (Cod.BNA: 7336001, Lat:-36.02, Lon:-72.38). This catchment is located in El Maule Region in Chile, with a pluvial regime, a total drainage area of 622.1 km2, and elevations ranging from 134 to 736 m a.s.l. Data were downloaded from the CAMELS-CL dataset (https://camels.cr2.cl) from 01/Jan/1979 to 31/Dec/2019 (including some missing values).

Usage

```
data(Cauquenes7336001)
```

Format

zoo matrix with 5 time series:
-) *P_mm*: Spatially-averaged mean daily values of precipitation computed based on the CR2met dataset, [mm/day].
-) *Tmx_degC*: Spatially-averaged maximum daily values of air temperature computed based on the CR2met dataset, [degree Celsius].
-) *Tmin_degC*: Spatially-averaged minimum daily values of air temperature computed based on the CR2met dataset, [degree Celsius].
-) *PET_mm*: Spatially-averaged mean daily values of precipitation computed based on the Hargreaves-Samani equation and daily maximum and minimum air temperatures obtained from the CR2met dataset, [mm/day].
-) *Qobs_mm*: Daily sreamflows, [mm], measured at the "Cauquenes en El Arrayan" (7336001) station.
-) *Qobs_m3s*: Daily sreamflows, [m3/s], measured at the "Cauquenes en El Arrayan" (7336001) station.

Source

Provided by Center for Climate and Resilience Research, Universidad de Chile, Santiago, Chile (https://camels.cr2.cl/, last accessed [Nov 2023]).
These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

References

Alvarez-Garreton, C., Mendoza, P. A., Boisier, J. P., Addor, N., Galleguillos, M., Zambrano-Bigiarini, M., Lara, A., Puelma, C., Cortes, G., Garreaud, R., McPhee, J., and Ayala, A (2018). The CAMELS-CL dataset: catchment attributes and meteorology for large sample studies-Chile dataset. Hydrology and Earth System Sciences, 22(11), 5817-5846. doi:10.5194/hess-22-5817-2018.

climograph *Climograph*

Description

Function to draw a climograph based on precipitation and air temperature data, with several options for customisation.

Usage

```
climograph(pcp, tmean, tmx, tmn, na.rm=TRUE,
           from, to, date.fmt="%Y-%m-%d",
           main="Climograph", pcp.label="Precipitation, [mm]",
         tmean.label="Air temperature, [\U00B0 C]", start.month=1, pcp.solid.thr,
           pcp.ylim, temp.ylim,pcp.col="lightblue", pcp.solid.col="skyblue2",
           tmean.col="darkred", tmn.col="blue", tmx.col="red",
           pcp.labels=TRUE,
           tmean.labels=TRUE, tmx.labels=TRUE, tmn.labels=TRUE,
           pcp.labels.cex=0.8, temp.labels.cex=0.8,
           pcp.labels.dx=c(rep(ifelse(plot.pcp.probs, -0.25,  0.0),6),
                           rep(ifelse(plot.pcp.probs, -0.25,  0.0),6)),
           pcp.labels.dy=rep(2, 12),
         temp.labels.dx=c(rep(-0.2,6), rep(0.2,6)), temp.labels.dy=rep(-0.4, 12),
           plot.pcp.probs=TRUE, pcp.probs=c(0.25, 0.75),
           plot.temp.probs=TRUE, temp.probs=c(0.25, 0.75),
           temp.probs.col=c("#3399FF", "#FF9966", "#FFCC66"),
           temp.probs.alpha=0.3,
           lat, lon
           )
```

Arguments

pcp	variable of type zoo with monthly, daily or subdaily precipitation data.
tmean	variable of type 'zoo' with monthly, daily or subdaily mean temperature data.
tmx	variable of type 'zoo' with monthly, daily or subdaily maximum temperature data. ONLY used (together with tmn) when tmean is missing.
tmn	variable of type 'zoo' with monthly, daily or subdaily minimum temperature data. ONLY used (together with tmx) when tmean is missing.
na.rm	Logical. Should missing values be removed? -) TRUE : the monthly values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA within a month, the resulting average monthly value is NA .

from	OPTIONAL, used for extracting a subset of values. Character indicating the starting date for the values to be extracted. It must be provided in the format specified by date.fmt.
to	OPTIONAL, used for extracting a subset of values. Character indicating the ending date for the values to be extracted. It must be provided in the format specified by date.fmt.
date.fmt	Character indicating the format in which the dates are stored in *dates*, *from* and *to*. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
main	Character representing the main title of the plot.
pcp.label	Character used in the legend to represent the monthly average precipitation (mostly thought for languages different from English).
tmean.label	Character used in the legend to represent the monthly average temperature (mostly thought for languages different from English).
start.month	[OPTIONAL]. Only used when the (hydrological) year of interest is different from the calendar year. numeric in [1:12] indicating the starting month of the (hydrological) year. Numeric values in [1, 12] represents months in [January, December]. By default start.month=1.
pcp.solid.thr	[OPTIONAL]. Only used when using (sub)daily precipitation and temperature are gives as input data. numeric, indicating the temperature, in degrees Celsius, used to discriminate between solid and liquid precipitation. When daily tmean <= pcp.solid.thr the precipitation for that day is considered as solid precipitation.
pcp.ylim	[OPTIONAL] numeric of length 2 with the the range used for the precipitation axis. The second value should be larger than the first one.
temp.ylim	[OPTIONAL] numeric of length 2 with the the range used for the secondary temperature axis. The second value should be larger than the first one.
pcp.col	Color used in the legend to represent the monthly average precipitation.
pcp.solid.col	Color used in the legend to represent the monthly average solid precipitation.
tmean.col	Color used in the legend to represent the monthly average temperature.
tmn.col	Color used in the legend to represent the monthly minimum temperature.
tmx.col	Color used in the legend to represent the monthly maximum temperature.
pcp.labels	logical. Should monthly precipitation values to be shown above the bars?. By default pcp.labels=TRUE.
tmean.labels	logical. Should monthly mean temperature values to be shown above the lines?. By default tmean.labels=TRUE.
tmx.labels	logical. Should monthly maximum temperature values to be shown above the lines?. By default tmx.labels=TRUE.
tmn.labels	logical. Should monthly minimum temperature values to be shown above the lines?. By default tmn.labels=TRUE.

pcp.labels.cex numeric giving the amount by which plotting characters used to show the numeric values of monthly precipitation values are scaled relative to the default.
It is only used when pcp.labels=TRUE.

temp.labels.cex
numeric giving the amount by which plotting characters used to show the numeric values of monthly air temperature values (mean, maximum, minimum) are scaled relative to the default.
It is only used when tmean.labels=TRUE or tmx.labels=TRUE or tmn.labels=TRUE.

pcp.labels.dx numeric of length 12 giving the amount of horizontal coordinate positions that have to be used to shift the labels of monthly precipitation values.
It is only used when pcp.labels=TRUE.
Lengths smaller than 12 are recycled and larger lengths are not used.

pcp.labels.dy numeric of length 12 giving the amount of vertical coordinate positions that have to be used to shift the labels of monthly precipitation values.
It is only used when pcp.labels=TRUE.
Lengths smaller than 12 are recycled and larger lengths are not used.

temp.labels.dx numeric of length 12 giving the amount of horizontal coordinate positions that have to be used to shift the labels of monthly air temperature values (mean, maximum, minimum).
It is only used when tmean.labels=TRUE or tmx.labels=TRUE or tmn.labels=TRUE.
Lengths smaller than 12 are recycled and larger lengths are not used.

temp.labels.dy numeric of length 12 giving the amount of vertical coordinate positions that have to be used to shift the labels of monthly air temperature values (mean, maximum, minimum).
It is only used when tmean.labels=TRUE or tmx.labels=TRUE or tmn.labels=TRUE.
Lengths smaller than 12 are recycled and larger lengths are not used.

plot.pcp.probs logical used to decide whether to show uncertainty values around the monthly mean precipitation values. By default plot.pcp.probs=TRUE.
When plot.pcp.probs=TRUE the pcp.probs argument is used to define the values of the lower an upper uncertainty bounds.

pcp.probs numeric of length 2. It defines the quantile values used to compute the lower an upper uncertainty bounds for each one of the 12 monthly precipitation values.
By default pcp.probs=c(0.25, 0.75), which indicates that the quantiles 0.25 and 0.75 are used to compute the lower an upper uncertainty bounds for each one of the 12 monthly precipitation values. If pcp is a (sub)daily zoo object, it is first aggregated into monthly values using mean, and then the pcp.probs quantiles are computed over all the monthly values belonging to a calendar month.

plot.temp.probs
logical used to decide whether to show uncertainty values around the monthly mean temperature values. By default plot.temp.probs=TRUE.
When plot.temp.probs=TRUE the temp.probs argument is used to define the values of the lower an upper uncertainty bounds.

temp.probs numeric of length 2. It is used to define quantile values used to compute the lower an upper uncertainty bounds for each one of the 12 monthly mean temperature values.
If tmx and tmn are provided, then temp.probs are used to compute the lower an

upper uncertainty bounds for each one of the 12 monthly maximum/minimum temperature values.
By default `temp.probs=c(0.25, 0.75)`, which indicates that the quantiles 0.25 and 0.75 are used to compute the lower an upper uncertainty bounds for each one of the 12 monthly mean(maximum/minimum) values. If `tmx/tmn` is provided and is a (sub)daily zoo object, it is first aggregated into monthly values using `mean`, and then the `temp.probs` quantiles are computed over all the monthly values belonging to a calendar month.

`temp.probs.col` character of length 3, with the colors used to for plotting the uncertainty bands around the average monthly values of the minimum, mean and maximum air temperature, respectively.
If `tmx` and `tmn` are not provided by the user, the second element of `temp.probs.col` will still be used to define the color of the uncertainty band around the mean monthly values of air temperature.

`temp.probs.alpha`
numeric of length 1, with the factor used to modify the opacity of `temp.probs.col`. Typically in [0,1], with 0 indicating a completely transparent colour and 1 indicating no transparency.

`lat` [OPTIONAL] numeric or character used to show the latitude for which the climograph was plotted for.

`lon` [OPTIONAL] numeric or character used to show the longitude for which the climograph was plotted for.

Note

If the output climograph present some mixed or not legible characters, you might try resizing the graphical window and run climograph again with the new size, until you get the climograph in the way you want to.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

monthlyfunction

Examples

```
######################
## Ex1: Loading the DAILY precipitation, maximum and minimum air temperature at
##      station Maquehue Temuco Ad (Chile)
data(MaquehueTemuco)

# Subsetting to only the 1981-1990 temporal period
MaquehueTemuco <- window(MaquehueTemuco, start="1981-01-01", end="1990-12-31")

pcp <- MaquehueTemuco[, 1]
tmx <- MaquehueTemuco[, 2]
```

```
tmn <- MaquehueTemuco[, 3]

## Plotting a full climograph
m <- climograph(pcp=pcp, tmx=tmx, tmn=tmn, na.rm=TRUE,
                main="Maquehue Temuco Ad (Chile)", lat=-38.770, lon=-72.637)

## Not run:
## Plotting a climograph with uncertainty bands around mean values,
## but with no labels for tmx and tmn
m <- climograph(pcp=pcp, tmx=tmx, tmn=tmn, na.rm=TRUE, tmx.labels=FALSE, tmn.labels=FALSE,
                main="Maquehue Temuco Ad (Chile)", lat=-38.770, lon=-72.637)

## Plotting a climograph with uncertainty bands around mean values, but with no labels for
##  tmx, tmn and pcp
m <- climograph(pcp=pcp, tmx=tmx, tmn=tmn, na.rm=TRUE,
               pcp.labels=FALSE, tmean.labels=FALSE, tmx.labels=FALSE, tmn.labels=FALSE,
                main="Maquehue Temuco Ad (Chile)", lat=-38.770, lon=-72.637)

## Plotting a climograph with no uncertainty bands around mean values
m <- climograph(pcp=pcp, tmx=tmx, tmn=tmn, na.rm=TRUE, plot.pcp.probs=FALSE, plot.temp.probs=FALSE,
                main="Maquehue Temuco Ad (Chile)", lat=-38.770, lon=-72.637)

## Plotting the most basic climograph: only mean values of precipiation and air temperature
m <- climograph(pcp=pcp, tmean=0.5*(tmn+tmx), na.rm=TRUE, plot.pcp.probs=FALSE,
                plot.temp.probs=FALSE, main="Maquehue Temuco Ad (Chile)",
                lat=-38.770, lon=-72.637)


## Plotting a full climograph, starting in April (start.month=4) instead of January (start.month=1),
## to better represent the hydrological year in Chile (South America)
m <- climograph(pcp=pcp, tmx=tmx, tmn=tmn, na.rm=TRUE,
                start.month=4, temp.labels.dx=c(rep(-0.2,4), rep(0.2,6),rep(-0.2,2)),
                main="Maquehue Temuco Ad (Chile)", lat=-38.770, lon=-72.637)


## Plotting a full climograph with monthly data
pcp.m <- daily2monthly(pcp, FUN=sum)
tmx.m <- daily2monthly(tmx, FUN=mean)
tmn.m <- daily2monthly(tmn, FUN=mean)
m <- climograph(pcp=pcp.m, tmx=tmx.m, tmn=tmn.m, na.rm=TRUE,
                main="Maquehue Temuco Ad (Chile)", lat=-38.770, lon=-72.637)

## End(Not run)
```

---

cmv *Counting Missing Values*

Description

Generic function for counting the percentage/amount of missing values in a zoo object, using a user-defined temporal scale.

Usage

```
cmv(x, ...)

## Default S3 method:
cmv(x, tscale=c("hourly", "daily", "weekly", "monthly",
            "quarterly", "seasonal", "annual"),
            out.type=c("percentage", "amount"), dec=3,
            start="00:00:00", start.fmt= "%H:%M:%S", tz,
            start.month=1, ...)

## S3 method for class 'zoo'
cmv(x, tscale=c("hourly", "daily", "weekly", "monthly",
            "quarterly", "seasonal", "annual"),
            out.type=c("percentage", "amount"), dec=3,
            start="00:00:00", start.fmt= "%H:%M:%S", tz,
            start.month=1, ...)

## S3 method for class 'data.frame'
cmv(x, tscale=c("hourly", "daily", "weekly", "monthly",
            "quarterly", "seasonal", "annual"),
            out.type=c("percentage", "amount"), dec=3,
            start="00:00:00", start.fmt= "%H:%M:%S", tz,
            start.month=1, dates=1, date.fmt="%Y-%m-%d", ...)

## S3 method for class 'matrix'
cmv(x, tscale=c("hourly", "daily", "weekly", "monthly",
            "quarterly", "seasonal", "annual"),
            out.type=c("percentage", "amount"), dec=3,
            start="00:00:00", start.fmt= "%H:%M:%S", tz,
            start.month=1, dates=1, date.fmt="%Y-%m-%d", ...)
```

Arguments

x	zoo, data.frame or matrix object, with the time series to be analised. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represents the time series measured in a gauging statio, and the column names of x have to correspond to the ID of each station (starting by a letter).
tscale	character with the temporal scale to be used for analysing the mssing data. Valid values are: -) hourly: the percentage/amount of missing values will be given for each hour and ,therefore, the expected time frequency of x must be sub-hourly. -) daily: the percentage/amount of missing values will be given for each day

and, therefore, the expected time frequency of x must be sub-daily (i.e., hourly or sub-hourly).
-) weekly: the percentage/amount of missing values will be given for each week (starting on Monday) and, therefore, the expected time frequency of x must be sub-weekly (i.e., daily, (sub)hourly).
-) monthly: the percentage/amount of missing values will be given for each month and, therefore, the expected time frequency of x must be sub-monthly (i.e., daily, hourly or sub-hourly).
-) quarterly: the percentage/amount of missing values will be given for each quarter and, therefore, the expected time frequency of x must be sub-quarterly (i.e., monthly, daily, hourly or sub-hourly).
-) seasonal: the percentage/amount of missing values will be given for each weather season (see ?time2season) and, therefore, the expected time frequency of x must be sub-seasonal (i.e., monthly, daily, hourly or sub-hourly).
-) annual: the percentage/amount of missing values will be given for each year and, therefore, the expected time frequency of x must be sub-annual (i.e., seasonal, monthly, daily, hourly or sub-hourly).

dec — integer indicating the amount of decimal places included in the output.
It is only used when out.type=='percentage'.

start — character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by start.fmt.
This value is used to define the time when a new day begins (e.g., for some rain gauge stations).
-) All the values of x with a time attribute before start are considered as belonging to the day before the one indicated in the time attribute of those values.
-) All the values of x with a time attribute equal to start are considered to be equal to "00:00:00" in the output zoo object.
-) All the values of x with a time attribute after start are considered as belonging to the same day as the one indicated in the time attribute of those values.

It is useful when the daily values start at a time different from "00:00:00". Use with caution. See examples.

start.fmt — character indicating the format in which the time is provided in start, By default date.fmt=%H:%M:%S. See format in as.POSIXct.

start.month — numeric in [1,..,12], representing the starting month (1:Jan, ..., 12:Dec) to be used in the computation of annual values. By default start.month=1.

tz — character, with the specification of the time zone used in both x and start. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct.
If tz is missing (the default), it is automatically set to the time zone used in time(x).
This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.

dates — numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of x (e.g., gauging station).
If dates is a number, it indicates the index of the column in x that stores the

	date and times. If dates is a factor, it is converted into POSIXct class, using the date format specified by date.fmt If dates is already of POSIXct or POSIXt class, this function verifies that the number of elements on it be equal to the number of elements in x.
date.fmt	character indicating the format in which the dates are stored in dates, By default date.fmt=%Y-%m-%d %H:%M:%S. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	character indicating how should be returned the missing values for each temporal scale. Valid values are: -) percentage: the missing values are returned as an real value, representing the percentage of missing values in each temporal scale. -) amount: the missing values are returned as an integer value, representing the absolute amount of missing values in each temporal scale.
...	further arguments passed to or from other methods.

Details

The amount of missing values in each temporal scale is computed just by counting the amount of NAs in each hour / day / week / month / quarter / season / year, while the percentage of missing values in each temporal scale is computed by dividing the previous number by the total number of data elements in each hour / day / week / month / quarter / season / year.

This function was developed to allow the selective removal of values when agregting from a high temporal resolution into a lower temporal resolution (e.g., from hourly to daily or from daily to monthly), using any of the temporal aggregation functions available int his package (e.g., hourly2daily, daily2monthly)

Value

a zoo object with the percentage/amount of missing values for each temporal scale selected by the user.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

dwi, subhourly2hourly, subdaily2daily, daily2monthly, daily2annual, monthlyfunction, izoo2rzoo

Examples

```
######################
## Ex1: Loading the DAILY precipitation data at SanMartino (25567 daily values)
data(SanMartinoPPts)
x <- SanMartinoPPts
```

```
# Subsetting to only the 1981-1990 temporal period
x <- window(x, start="1981-01-01", end="1990-12-31")

## Transforming into NA the 10% of values in 'x'
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

# Getting the amount of NAs in 'x' for each week (starting on Monday)
cmv(x, tscale="weekly")

# Getting the amount of NAs in 'x' for each month
cmv(x, tscale="monthly")

# Getting the amount of NAs in 'x' for each quarter
cmv(x, tscale="quarterly")

# Getting the amount of NAs in 'x' for each weather season
cmv(x, tscale="seasonal")

# Getting the amount of NAs in 'x' for each year
cmv(x, tscale="annual")
######################
## Ex2: Loading the time series of HOURLY streamflows for the station
## Karamea at Gorge (52579 hourly values)
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

# Subsetting to only the first two years (1981-1982)
x <- window(x, end=as.POSIXct("1982-12-31 23:59:00", tz="UTC") )

## Transforming into NA the 30% of values in 'x'
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

# Getting the amount of NAs in 'x' for each day
cmv(x, tscale="daily")

# Getting the amount of NAs in 'x' for each weather season
cmv(x, tscale="seasonal")
```

---

daily2weekly *Daily -> Weekly*

Description

Generic function for transforming a DAILY (or sub-daily) regular time series into a WEEKLY one

Usage

```
daily2weekly(x, ...)

## Default S3 method:
daily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'zoo'
daily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'data.frame'
daily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", out.fmt="numeric",
        verbose=TRUE, ...)

## S3 method for class 'matrix'
daily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", out.fmt="numeric",
        verbose=TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with (sub)daily time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represents the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that have to be applied for transforming from daily to weekly time step (e.g., for precipitation FUN=sum and for temperature and streamflow ts FUN=mean). FUN MUST accept the na.rm argument, because na.rm is passed to FUN.
na.rm	Logical. Should missing values be removed? -) TRUE : the weekly values are computed only for weeks with a percentage of missing values less than na.rm.max -) FALSE: if there is AT LEAST one NA within a month, the corresponing weekly values in the output object will be NA.
na.rm.max	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each week to keep the weekly aggregated value in the output object of this function. In other words, if the percentage of missing values in a given week is larger than na.rm.max the corresponding weekly value will be NA.
dates	numeric, factor or Date object indicating how to obtain the dates for each gauging station If dates is a number (default), it indicates the index of the column in *x* that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days on it be equal to the number of elements in x

date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	Character that defines the desired type of output. Valid values are: -) data.frame: a data.frame, with as many columns as stations are included in x, and row names indicating the month and year for each value. -) db : a data.frame, with 4 columns will be produced. The first column (StationID) stores the ID of the station, The second column (Year) stores the year The third column (Month) stores the Month The fourth column (Value) stores the numerical values corresponding to the values specified in the three previous columns.
out.fmt	OPTIONAL. Only used when x is a matrix or data.frame object /cr character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo.
verbose	logical; if TRUE, progress messages are printed
...	arguments additional to na.rm passed to FUN.

Value

a zoo object with weekly time frequency

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

cmv, subhourly2hourly, daily2monthly, daily2annual, subdaily2daily, weeklyfunction, hydroplot, vector2zoo, izoo2rzoo, as.Date

Examples

```
######################
## Ex1: Computation of weekly values, removing any missing value in 'x'

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three weeks (Jan/1921 - Mar/1921)
x <- window(x, end="1921-03-31")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
```

```
x[na.index] <- NA

## Agreggating from Daily to Weekly, removing any missing value in 'x'
w <- daily2weekly(x, FUN=sum, na.rm=TRUE)

######################
## Ex2: Computation of Weekly values only when the percentage of NAs in each
#       week is lower than a user-defined percentage (10% in this example).

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three weeks (Jan/1921 - Mar/1921)
x <- window(x, end="1921-03-31")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Daily to Weekly, only for weeks with less than 10% of missing values
w2 <- daily2weekly(x, FUN=sum, na.rm=TRUE, na.rm.max=0.1)

# Verifying that the weeks 01, 02, 06, 08, 10, 11, 12 of 'x' had 10% or more of missing values
cmv(x, tscale="weekly")

######################
## Ex3: Computation of Weekly values in a two-column zoo object,
##      only when the percentage of NAs in each week is lower than a user-defined
##      percentage (10% in this example).

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three weeks (Jan/1921 - Mar/1921)
x <- window(x, end="1921-03-31")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Creating a two-column zoo object
X <- cbind(x, y=x)

## Daily to Weekly, only for weeks with less than 10% of missing values
w2 <- daily2weekly(X, FUN=sum, na.rm=TRUE, na.rm.max=0.1)
```

```
# Verifying that the weeks 01, 02, 06, 08, 10, 11, 12 of 'x' had 10% or more of missing values
cmv(X, tscale="weekly")
```

---

dip *Days in Period*

Description

Given any starting and ending dates, it generates:
1) a vector of class Date with all the days between `from` and `to` (both of them included), OR
2) the amount of days between the two dates

Usage

```
dip(from, to, date.fmt = "%Y-%m-%d", out.type = "seq")
```

Arguments

`from`	Character indicating the starting date for creating the sequence. It has to be in the format indicated by `date.fmt`.
`to`	Character indicating the ending date for creating the sequence. It has to be in the format indicated by `date.fmt`.
`date.fmt`	character indicating the format in which the dates are stored in `from` and `to`, e.g. `%Y-%m-%d`. See `format` in `as.Date`. ONLY required when `class(dates)=="factor"` or `class(dates)=="numeric"`.
`out.type`	Character indicating the type of result that is given by this function. Valid values are: 1) `seq` : a vector of class Date with all the days between the two dates, OR 2) `nmbr`: a single numeric value with the amount of days between the two dates.

Value

Depending on the value of `out.type`, it returns:
1) a vector of class Date with all the days between `from` and `to` (both of them included), OR
2) the amount of days between the two dates

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

`mip`, `yip`, `hip`, `diy`

Examples

```
## Sequence of daily dates between "1961-01-01" and "1961-12-31" ##
dip("1961-01-01", "1961-12-31")

## Number of days between "1961-01-01" and "1965-06-30",
## but using "%d-%m-%Y" as date format.
dip("01-01-1961", "30-06-1965", date.fmt= "%d-%m-%Y", out.type = "nmbr")
```

`diy` *Days in Year*

Description

Given a single numeric value representing a year, it generates:
1) a vector of dates with all the days within the year, OR
2) the amount of days in the specified year

Usage

```
diy(year, out.type = "seq")
```

Arguments

`year`	numeric, the year for which the sequence of days will be generated
`out.type`	Character indicating the type of result that is given by this function. Valid values are: -) `seq` : a vectorial sequence with all the days within the given year -) `nmbr`: the number of days in the vectorial sequence with all the days within the given year

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

`hip`, `dip`, `mip`, `yip`

Examples

```
## Sequence of daily dates for the year 1961
diy(1961)

## Computing the number of days between in 1961
diy(1961, out.type = "nmbr")
```

dm2seasonal *(sub)Daily/Monthly -> Seasonal Values*

Description

Generic function for computing a seasonal value for every year of a sub-daily/daily/weekly/monthly time series

Usage

```
dm2seasonal(x, ...)
subdaily2seasonal(x, ...)

## Default S3 method:
dm2seasonal(x, season, FUN, na.rm = TRUE, out.fmt="%Y", ...)

## S3 method for class 'zoo'
dm2seasonal(x, season, FUN, na.rm = TRUE, out.fmt="%Y", ...)

## S3 method for class 'data.frame'
dm2seasonal(x, season, FUN, na.rm = TRUE, dates=1, date.fmt = "%Y-%m-%d",
            out.type = "data.frame", out.fmt="%Y", ...)

## S3 method for class 'matrix'
dm2seasonal(x, season, FUN, na.rm = TRUE, dates=1, date.fmt = "%Y-%m-%d",
            out.type = "data.frame", out.fmt="%Y", ...)
```

Arguments

x	zoo, xts, data.frame or matrix object, with sub-daily, daily, weekly or monthly time series. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
season	character, indicating the weather season to be used for selecting the data. Valid values are: -) DJF : December, January, February -) MAM : March, April, May -) JJA : June, July, August -) SON : September, October, November -) DJFM: December, January, February, March -) AM : April, May -) JJAS: June, July, August, September -) ON : October, November

FUN — Function that will be applied to ALL the values of x belonging to the given weather season (e.g., FUN can be some of mean, max, min, sd).
The FUN value for the winter season (DJF or DJFM) is computed considering the consecutive months of December, January and February/March. See 'Note' section.

na.rm — Logical. Should missing values be removed?
-) TRUE : the seasonal values are computed considering only those values different from NA (**very important when** FUN=sum)
-) FALSE: if there is AT LEAST one NA within a weather season, the corresponding seasonal values are NA

out.fmt — Character indicating the date format for the output time series. See format in as.Date. Possible values are:
-) %Y : only the year will be used for the time. Default option. (e.g., "1961" "1962"...)
-) %Y-%m-%d: a complete date format will be used for the time. (e.g., "1961-01-01" "1962-01-01"...)

dates — numeric, factor or Date object indicating how to obtain the dates.
If dates is a number (default), it indicates the index of the column in x that stores the dates
If dates is a factor, it is converted into Date class, by using the date format specified by date.fmt
If dates is already of Date class, the code verifies that the number of days on it be equal to the number of elements in x

date.fmt — Character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date.
ONLY required when class(dates)=="factor" or class(dates)=="numeric".

out.type — Character that defines the desired type of output. Valid values are:
-) data.frame: a data.frame, with as many columns as stations are included in x, the year corresponding to each seasonal value are used as row names.
-) db : a data.frame, with 4 columns will be produced.
The first column (StationID) stores the ID of the station The second column (Year) stores the year,
The third column (Season) stores the season,
The fourth column (Value) contains the seasonal value corresponding to the values specified in the previous three columns

... — further arguments passed to or from other methods.

Value

A numeric vector with the seasonal values for all the years in which x is defined.

Warning

For any year, the FUN value for the winter season (DJF), is computed considering only January and February, and the value of December is used for computing the winter value of the next year.

Note

FUN is applied to all the values of x belonging to the selected season, so the results of this function depends on the frequency sampling of x and the type of function given by FUN

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

, hydroplot, seasonalfunction, time2season, extract, daily2monthly, daily2annual, monthly2annual

Examples

```
############
## Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

## Winter (DJF) values of precipitation for each year of 'x'
dm2seasonal(x, FUN=sum, season="DJF")

############
## Loading the HOURLY discharge data for the Karamea at Gorge streamgauge station
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

## Mean winter (DJF) values of streamflow for each year of 'x'
dm2seasonal(x, FUN=mean, season="DJF")
subdaily2seasonal(x, FUN=mean, season="DJF") # same as above
```

---

drawTimeAxis *Customized Time Axis*

Description

For a nice time series plot, this function draws a customized time axis, with annual, monthly, daily and sub-daily time marks and labels.

Usage

```
drawxaxis(x, tick.tstep = "auto", lab.tstep = "auto",
          lab.fmt=NULL, cex.axis=1, mgp=c(3, 2, 0), ...)
```

Arguments

x	time series that will be plotted using the X axis that will be draw class(x) must be ts or zoo
tick.tstep	Character indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: auto, years, quarters, months,weeks, days, hours, minutes, seconds.
lab.tstep	Character indicating the time step that have to be used for putting the labels on the time axis. Valid values are: auto, years, quarters, months,weeks, days, hours, minutes, seconds.
lab.fmt	Character indicating the format to be used for the label of the axis. See format in as.Date. If not specified (lab.fmt=NULL), it will try to use: -) "%Y-%m-%d" when lab.tstep=="days", -) "%b-%Y" when lab.tstep=="year" or lab.tstep=="month".
cex.axis	magnification of axis annotation relative to cex (See par).
mgp	The margin line (in mex units) for the axis title, axis labels and axis line (See par). Default value is mgp = c(3, 2, 0).
...	further arguments passed to the axis function or from other methods.

Note

From version 0.3-0 it changed its name from drawxaxis to drawTimeAxis, in order to have a more intuitive name. The old drawxaxis function is deprecated, but still be kept for compatibility reasons.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

Examples

```
## Loading the SanMartino precipitation data
data(SanMartinoPPts)
x <- window(SanMartinoPPts, end=as.Date("1930-12-31"))

## Plotting the daily ts only, and then automatic 'x' axis
plot(x, xaxt = "n", xlab="Time")
drawTimeAxis(x)

## Plotting the daily ts only, and then monthly ticks in the 'x' axis,
## with annual labels.
plot(x, xaxt = "n", xlab="Time")
drawTimeAxis(x, tick.tstep="months", lab.tstep="years")
```

dwdays *Amount of dry/wet days in a time series*

Description

Given a daily time series (usually precipitation), this function computes the average amount of wet/dry days in each month.

Usage

```
dwdays(x, ...)

## Default S3 method:
dwdays(x, thr=0, type="wet", na.rm=TRUE, ... )

## S3 method for class 'data.frame'
dwdays(x, thr=0, type="wet", na.rm=TRUE,
        dates=1, date.fmt="%Y-%m-%d", verbose=TRUE,...)

## S3 method for class 'matrix'
dwdays(x, thr=0, type="wet", na.rm=TRUE,
        dates=1, date.fmt="%Y-%m-%d", verbose=TRUE,...)
```

Arguments

x	zoo, data.frame or matrix object, usually with daily time series of precipitation. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
thr	numeric. Value of daily precipitation used as threshold for classifying a day as dry/wet or not. Days with a precipitation value larger to thr are classified as *wet days*, whereas precipitation values lower to thr are classified as *dry days*.
type	character, indicating if the daily values have to be classified as dry or wet days. It works linked to the values specified in thr. Valid values are: wet, dry.
na.rm	Logical. Should missing values be removed before counting?
dates	numeric, factor or Date object indicating how to obtain the dates If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of element in x

date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
verbose	logical; if TRUE, progress messages are printed
...	further arguments passed to or from other methods.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

Examples

```
## Loading the SanMartino precipitation data
data(SanMartinoPPts)
x <- SanMartinoPPts

## Average amount of wet days in each month (for this example, this means days
## with precipitation larger than 0.1mm)
dwdays(x, thr=0.1)
```

---

dwi *Days with Information*

Description

This function generates a table indicating the number of days with information (<>NA) within a zoo object, aggregated by year, month or month per year.

Usage

```
dwi(x, ...)

## Default S3 method:
dwi(x, out.unit = "years", from = start(x), to = end(x),
     date.fmt = "%Y-%m-%d", tstep="days", ...)

## S3 method for class 'zoo'
dwi(x, out.unit = "years", from = start(x), to = end(x),
     date.fmt = "%Y-%m-%d", tstep="days", ...)

## S3 method for class 'data.frame'
dwi(x, out.unit = "years", from, to, date.fmt = "%Y-%m-%d", tstep="days",
     dates = 1, verbose = TRUE, ...)

## S3 method for class 'matrix'
dwi(x, out.unit = "years", from, to, date.fmt = "%Y-%m-%d", tstep="days",
     dates = 1, verbose = TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with daily/monthly/annual time series. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
out.unit	aggregation time for the computation of the amount of days with information. Valid values are: -) months: monthly; -) years : annual; -) mpy : month per year (not available for data.frames)
from	Character indicating the starting date for the computations. It has to be in the format indicated by date.fmt. When x is a data.frame and this value is not provided, the date corresponding to the first row of x is used
to	Character indicating the ending date for the computations. It has to be in the format indicated by date.fmt. When x is a data.frame and this value is not provided, the date corresponding to the last row of x is used
date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
tstep	Time step used for storing the values in x. Valid values are: days, months, years. Since the version 0.3-0 of hydroTSM, this argument is not required any more, because it is not used any longer.
dates	numeric, factor or Date object indicating how to obtain the dates for each column of x If dates is a number, it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of element in x
verbose	logical; if TRUE, progress messages are printed
...	further arguments passed to or from other methods.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

matrixplot

Examples

```
## Loading the SanMartino precipitation data
data(SanMartinoPPts)
x <- SanMartinoPPts

## Not run:
## Days with information per year
dwi(x)

## Days with information per month per year.
dwi(x, out.unit="mpy")

## End(Not run)

############
## Not run:
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

## Months with information per year in the 9 first stations of 'EbroPPtsMonthly'
a <- dwi(EbroPPtsMonthly[,1:10], out.unit="years", dates=1)

## Before plotting the results in 'a', and just for obtaining a more interesting
## plot, 70 random numbers (between 1 and 11) are introduced in 'a'
a[sample(length(a), size = 70)] <- rep(1:11, length=70)

## Plotting the amount of months with information per year in each station
matrixplot(a, var.type="Days", main="Number of months with info per year")

## End(Not run)
```

EbroPPtsMonthly *Ebro Monthly Precipitation Time Series*

Description

Time series of monthly precipitation on 331 stations of the Ebro River basin (NW Spain), for the period January/1961 to December/1963.

Usage

```
data(EbroPPtsMonthly)
```

Format

A data.frame with 331 monthly time series, plus the first field storing the dates corresponding to each row.

Details

Monthly time series of precipitation on 331 stations of the Ebro River basin (Spain), where some data were in-filled by using the MOSS method and from daily time series used in the technical report "Estudio de Recursos de la Cuenca del Ebro".

Source

Downloaded from the web site of the Confederacion Hidrografica del Ebro (CHE) http://www.chebro.es/ (original link http://oph.chebro.es/DOCUMENTACION/PrecipitacionMensualRelleno/PrecipitacionMensualRe last accessed [March 2010]).
These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

---

extract *Extract from Zoo*

Description

Extracts from a zoo object all the values belonging to a given month, year or weather season.

Usage

```
extract(x, ...)

## Default S3 method:
extract(x, trgt, ...)

## S3 method for class 'zoo'
extract(x, trgt, ...)
```

Arguments

x	zoo object
trgt	numeric or character indicating the elements to extract from x. Valid values are: 1) integer(s) from 1 to 12: trgt is considered as month(s) (1=JAN, 2=FEB,...., 12=DEC), and all the values in x belonging to the month(s) specified by trgt will be extracted. 2) integer(s) > 12: trgt is considered as year(s), and all the values in x belonging to the year(s) specified by trgt will be extracted 3) character: trgt is considered as a weather season, and all the values in x belonging to the season specified by trgt will be extracted. Valid values are: -) DJF : December, January, February -) MAM : March, April, May -) JJA : June, July, August -) SON : September, October, November -) DJFM: December, January, February, March

-) AM : April, May
-) JJAS: June, July, August, September
-) ON : October, November

... further arguments passed to or from other methods

Value

a zoo object with the extracted values.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

time2season, seasonalfunction, daily2annual, daily2monthly

Examples

```
### Loading temperature data ##
data(SanMartinoPPts)
x <- SanMartinoPPts

## Extracting all the values belonging to February (FEB=2)
extract(x, trgt=2)

## Extracting all the values belonging to February (FEB=2) and April (APR=4)
extract(x, trgt=c(2,4))

## Extracting all the values belonging to the year 1970
extract(x, trgt=1970)

## Extracting all the values belonging to the years 1970 and 1972
extract(x, trgt=c(1970,1972))

## Extracting all the values belonging to the autumn
extract(x, trgt="SON")
```

---

fdc *Flow Duration Curve*

Description

Computes and plots the Flow Duration Curve (FDC) corresponding to a given time series of streamflow discharges.

Usage

```
fdc(x, ...)

## Default S3 method:
fdc(x,lQ.thr=0.7,hQ.thr=0.2, plot=TRUE, log="y",
    main="Flow Duration Curve", xlab="% Time flow equalled or exceeded",
   ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05), col="black",
   pch=1, lwd=1, lty=1, cex=0.4, cex.axis=1.2, cex.lab=1.2, leg.txt=NULL, leg.cex=1,
    leg.pos="topright", thr.pos="bottomleft",
    verbose= TRUE, thr.shw=TRUE, new=TRUE, ...)

## S3 method for class 'matrix'
fdc(x, lQ.thr=0.7, hQ.thr=0.2, plot=TRUE, log="y",
    main= "Flow Duration Curve",  xlab="% Time flow equalled or exceeded",
    ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05),
    col=palette("default")[1:ncol(x)], pch=1:ncol(x), lwd=rep(1, ncol(x)),
    lty=1:ncol(x), cex=0.4, cex.axis=1.2, cex.lab=1.2, leg.txt=NULL,
    leg.cex=1, leg.pos="topright", thr.pos="bottomleft",
    verbose=TRUE, thr.shw=TRUE, new=TRUE, ...)

## S3 method for class 'data.frame'
fdc(x, lQ.thr=0.7, hQ.thr=0.2, plot=TRUE, log="y",
     main= "Flow Duration Curve", xlab="% Time flow equalled or exceeded",
     ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05),
     col=palette("default")[1:ncol(x)], pch=1:ncol(x), lwd=rep(1, ncol(x)),
     lty=1:ncol(x), cex=0.4, cex.axis=1.2, cex.lab=1.2, leg.txt=NULL,
     leg.cex=1, leg.pos="topright", thr.pos="bottomleft",
     verbose=TRUE, thr.shw=TRUE, new=TRUE, ...)

## S3 method for class 'zoo'
fdc(x, lQ.thr=0.7, hQ.thr=0.2, plot=TRUE, log="y",
     main= "Flow Duration Curve", xlab="% Time flow equalled or exceeded",
     ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05),
     col=palette("default")[1:NCOL(x)], pch=1:NCOL(x), lwd=rep(1, NCOL(x)),
     lty=1:NCOL(x), cex=0.4, cex.axis=1.2, cex.lab=1.2, leg.txt=NULL,
     leg.cex=1, leg.pos="topright", thr.pos="bottomleft",
     verbose=TRUE, thr.shw=TRUE, new=TRUE, ...)
```

Arguments

x	numeric, zoo, data.frame or matrix object with the observed streamflows for which the flow duration curve have to be computed. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter). When x is a matrix or data.frame, the flow duration curve is computed for each column.

lQ.thr	numeric, low-flow separation threshold. If this value is different from NA, a vertical line is drawn in this value, and all the values to the right of it should be deemed as low flows. Default value is 0.7.
hQ.thr	numeric, high-flow separation threshold. If this value is different from NA, a vertical line is drawn in this value, and all the values to the left of it should br deemed as high flows. Default value is 0.2.
plot	logical. Indicates if the flow duration curve should be plotted or not. Default value is TRUE.
log	character, indicates which axis has to be plotted with a logarithmic scale. Default value is y
main	See plot. An overall title for the plot: see title.
xlab	A title for the x axis. See plot.
ylab	A title for the y axis. See plot.
ylim	The y limits of the plot. See plot.default.
yat	Only used when log="y". numeric, with points at which tick-marks will try to be drawn in the Y axis, in addition to the defaults computed by R. See the at argument in Axis.
xat	Only used when log="x". numeric, with points at which tick-marks will try to be drawn in the x axis, in addition to the defaults computed by R. See the at argument in Axis.
col	The colors to be used for lines and points. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion. Lines will all be plotted in the first colour specified. See plot.default.
pch	A vector of plotting characters or symbols: see points. See plot.default.
lwd	The line width, see par. See plot.default.
lty	The line type, see par. See plot.default.
cex	See plot.default. A numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex"). 'NULL' and 'NA' are equivalent to '1.0'. Note that this does not affect annotation
cex.axis	magnification of axis annotation relative to 'cex'.
cex.lab	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
leg.txt	vector with the names that have to be used for each column of x.
leg.cex	numeric, indicating the character expansion factor for the legend, *relative* to current par("cex"). Default value = 1
leg.pos	keyword to be used to position the legend. One of the list '"bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"'. This places the legend on the inside of the plot frame at the given location. See legend.

thr.pos	keyword to be used to position the streamflow values corresponding to the user-defined thresholds lQ.thr and hQ.thr. It is only used when thr.shw=TRUE. One of the list '"bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"'. This places the the streamflow values on the inside of the plot frame at the given location. See legend.
verbose	logical; if TRUE, progress messages are printed (when x is a matrix or data.frame).
thr.shw	logical, indicating if the streamflow values corresponding to the user-defined thresholds lQ.thr and hQ.thr have to be shown in the plot.
new	logical, if TRUE (default), a new plotting window is created.
...	further arguments passed to or from other methods (to the plotting functions)

Value

numeric, matrix or data.frame whose columns contains the % of time each one of the streamflow magnitudes given as input was equalled or exceeded. The resulting values have to be multiplied by 100 to get a percentage.

When plot is TRUE (default), the resulting flow duration curve is plotted in a new window.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

References

Vogel, R., and N. M. Fennessey (1994), Flow duration curves I: A new interpretation and confidence intervals, ASCE, Journal of Water Resources Planning and Management, 120(4).

Vogel, R., and N. Fennessey (1995), Flow duration curves II: A review of applications in water resources planning, Water Resources Bulletin, 31(6), 1029-1039, doi:10.1111/j.1752-1688.1995.tb03419.x.

Yilmaz, K. K., H. V. Gupta, and T. Wagener (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resour. Res., 44, W09417, doi:10.1029/2007WR006716.

See Also

fdcu

Examples

```
## Loading daily streamflows at the station Oca en Ona (Ebro River basin, Spain) ##
data(OcaEnOnaQts)

## Daily Flow Duration Curve
fdc(OcaEnOnaQts)

####################
```

```
# Getting the streamflow values corresponding to 5 and 95% of time equalled or
# exceeded (and also the first streamflow value in 'x' just for verification)
x  <- OcaEnOnaQts

# First streamflow value (x1=42.1 m3/s)
x1 <- x[1]

# Daily FDC for 'x'
y <- fdc(x)

# value of the FDC for x1 (y1=0.002739726)
y1 <- y[1]

# Performing cubic (or Hermite) spline interpolation of 'x' and 'y'
f <- splinefun(y,x)

# Getting the (known) streamflow value for 'y1'
f(y1) # 42.1 m3/s, equal to the known 'x1'

# Streamflow values corresponding to 5 and 95% of time equalled or exceeded
f(c(.05, .95))

####################
## Getting
data(OcaEnOnaQts)

## Daily Flow Duration Curve
fdc(OcaEnOnaQts)
```

---

fdcu *Flow Duration Curve with uncertainty bounds.*

Description

Computes and plots the Flow Duration Curve (FDC) for the streamflows given by x and for two uncertainty bounds, with the possibility of plotting an additional FDC representing simulated streamflows for x, in order to compare them.

Usage

```
fdcu(x, lband, uband, ...)

## Default S3 method:
fdcu(x, lband, uband, sim=NULL, lQ.thr=0.7, hQ.thr=0.2, plot=TRUE, log="y",
     main="Flow Duration Curve", xlab="% Time flow equalled or exceeded",
     ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05),
     col=c("black", "red"), pch=c(1, 15), lwd=c(1, 0.8), lty=c(1, 3), cex=0.2,
     cex.axis=1.2, cex.lab=1.2, leg.txt= c("Qobs", "Qsim", "95PPU"),
     leg.cex=1, leg.pos="auto", verbose= TRUE, thr.shw=TRUE, border=NA,
```

```
     bands.col="lightcyan", bands.density=NULL, bands.angle=45, new=TRUE, ...)

## S3 method for class 'matrix'
fdcu(x, lband, uband, sim=NULL, lQ.thr=0.7, hQ.thr=0.2, plot=TRUE, log="y",
     main="Flow Duration Curve", xlab="% Time flow equalled or exceeded",
     ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05),
     col=matrix(c(rep("black", ncol(x)),
     palette("default")[2:(ncol(x)+1)]), byrow=FALSE, ncol=2),
     pch=matrix(rep(c(1, 15), ncol(x)), byrow=TRUE, ncol=2),
     lwd=matrix(rep(c(1, 0.8), ncol(x)), byrow=TRUE, ncol=2),
     lty=matrix(rep(c(1, 3), ncol(x)), byrow=TRUE, ncol=2),
     cex=rep(0.1, ncol(x)), cex.axis=1.2, cex.lab=1.2,
     leg.txt=c("OBS", colnames(x), "95PPU"),  leg.cex=1, leg.pos="auto",
     verbose= TRUE,  thr.shw=TRUE, border=rep(NA, ncol(x)),
     bands.col=rep("lightcyan", ncol(x)), bands.density=rep(NULL, ncol(x)),
     bands.angle=rep(45, ncol(x)), new=TRUE, ...)

## S3 method for class 'data.frame'
fdcu(x, lband, uband, sim=NULL, lQ.thr=0.7, hQ.thr=0.2, plot=TRUE, log="y",
     main="Flow Duration Curve", xlab="% Time flow equalled or exceeded",
     ylab="Q, [m3/s]", ylim, yat=c(0.01, 0.1, 1), xat=c(0.01, 0.025, 0.05),
     col=matrix(c(rep("black", ncol(x)),
     palette("default")[2:(ncol(x)+1)]), byrow=FALSE, ncol=2),
     pch=matrix(rep(c(1, 15), ncol(x)), byrow=TRUE, ncol=2),
     lwd=matrix(rep(c(1, 0.8), ncol(x)), byrow=TRUE, ncol=2),
     lty=matrix(rep(c(1, 3), ncol(x)), byrow=TRUE, ncol=2),
     cex=rep(0.1, ncol(x)), cex.axis=1.2, cex.lab=1.2,
     leg.txt=c("OBS", colnames(x), "95PPU"), leg.cex=1, leg.pos="auto",
     verbose= TRUE, thr.shw=TRUE, border=rep(NA, ncol(x)),
     bands.col=rep("lightcyan", ncol(x)), bands.density=rep(NULL, ncol(x)),
     bands.angle=rep(45, ncol(x)), new=TRUE, ...)
```

Arguments

x	numeric, zoo, data.frame or matrix object with the observed streamflows for which the flow duration curve have to be computed. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter). When x is a matrix or data.frame, the flow duration curve is computed for each column.
lband	numeric, zoo, data.frame or matrix object with the streamflows representing the the lower uncertainty bound of x, for which the flow duration curve have to be computed. Measurements at several gauging stations can be stored in a data.frame of matrix object. When lband is a matrix or data.frame, the flow duration curve is

computed for each column.

uband — numeric, zoo, data.frame or matrix object with the streamflows representing the the upper uncertainty bound of x, for which the flow duration curve have to be computed.
Measurements at several gauging stations can be stored in a data.frame of matrix object. When uband is a matrix or data.frame, the flow duration curve is computed for each column.

sim — OPTIONAL.
numeric, zoo, data.frame or matrix object with the streamflows simulated for x, for which the flow duration curve have to be computed.
Measurements at several gauging stations can be stored in a data.frame of matrix object. When sim is a matrix or data.frame, the flow duration curve is computed for each column.

lQ.thr — numeric, low flows separation threshold. If this value is different from 'NA', a vertical line is drawn in this value, and all the values to the left of it are deemed low flows.

hQ.thr — numeric, high flows separation threshold. If this value is different from 'NA', a vertical line is drawn in this value, and all the values to the right of it are deemed high flows

plot — logical. Indicates if the flow duration curve should be plotted or not.

log — character, indicates which axis has to be plotted with a logarithmic scale. Default value is y.

main — See plot. An overall title for the plot: see title.

xlab — See plot. A title for the x axis: see title.

ylab — See plot. A title for the y axis: see title.

ylim — See plot.default. The y limits of the plot.

yat — Only used when log="y".
numeric, with points at which tick-marks will try to be drawn in the Y axis, in addition to the defaults computed by R. See the at argument in Axis.

xat — Only used when log="x".
numeric, with points at which tick-marks will try to be drawn in the x axis, in addition to the defaults computed by R. See the at argument in Axis.

col — See plot.default. The colors for lines and points. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.

pch — See plot.default. A vector of plotting characters or symbols: see points.

lwd — See plot.default. The line width, see par.

lty — See plot.default. The line type, see par.

cex — See plot.default. A numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex"). 'NULL' and 'NA' are equivalent to '1.0'. Note that this does not affect annotation.

cex.axis	magnification of axis annotation relative to 'cex'.
cex.lab	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
leg.txt	vector with the names that have to be used for each column of x.
leg.cex	numeric, indicating the character expansion factor for the legend, *relative* to current par("cex"). Default value = 1
leg.pos	keyword to be used to position the legend. One of the list '"bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"'. This places the legend on the inside of the plot frame at the given location. See legend. When leg.pos="auto", the legend provided by leg.txt is located on the 'bottomleft' when log="y" and on the 'topright' otherwise
verbose	logical; if TRUE, progress messages are printed
thr.shw	logical, indicating if the streamflow values corresponding to the user-defined thresholds lQ.thr and hQ.thr have to be shown in the plot. When leg.pos="auto", the legend with the threshold values is located on the 'topright' when log="y" and on the 'bottomleft' otherwise
border	See polygon. The color to draw the border of the polygon with the uncertainty bounds. The default, 'NA', means to omit borders.
bands.col	See polygon. The color for filling the polygon. The default, 'NA', is to leave polygons unfilled, unless bands.density is specified. If bands.density is specified with a positive value this gives the color of the shading lines.
bands.density	See polygon. The density of shading lines for the polygon with the uncertainty bounds, in lines per inch. The default value of 'NULL' means that no shading lines are drawn. A zero value of bands.density means no shading nor filling whereas negative values (and 'NA') suppress shading (and so allow color filling).
bands.angle	See polygon. The slope of shading lines for the polygon with the uncertainty bounds, given as an angle in degrees (counter-clockwise).
new	logical, if TRUE, a new plotting window is created.
...	further arguments passed to or from other methods (to the plotting functions)

Note

If you do not want to use logarithmic scale for the streamflow axis, you can do it by passing the log=" " to the ... argument.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

References

Vogel, R., and N. M. Fennessey (1994), Flow duration curves I: A new interpretation and confidence intervals, ASCE, Journal of Water Resources Planning and Management, 120(4).

Vogel, R., and N. Fennessey (1995), Flow duration curves II: A review of applications in water resources planning, Water Resources Bulletin, 31(6), 1029-1039, doi:10.1111/j.1752-1688.1995.tb03419.x.

Yilmaz, K. K., H. V. Gupta, and T. Wagener (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resour. Res., 44, W09417, doi:10.1029/2007WR006716.

See Also

fdc

Examples

```
## Loading daily streamflows at the station Oca en Ona (Ebro River basin, Spain) ##
data(OcaEnOnaQts)
q <- OcaEnOnaQts

# Creating a fictitious lower uncertainty band
lband <- q - min(q, na.rm=TRUE)

# Giving a fictitious upper uncertainty band
uband <- q + mean(q, na.rm=TRUE)

# Plotting the flow duration curve corresponding to 'q', with two uncertainty bounds
fdcu(q, lband, uband)
```

---

hip *Hours in Period*

Description

Given any starting and ending date/time objects, it generates:
1) a vector of class c("POSIXct" "POSIXt") with all the hours between the two date/time objects (both of them included), OR
2) the amount of hours between the two date/time objects

Usage

```
hip(from, to, date.fmt="%Y-%m-%d %H", out.type = "seq", tz="UTC")
```

Arguments

from	Character or POSIXct object indicating the starting date/time for creating the sequence. It has to be in the format indicated by date.fmt.
to	Character indicating the ending date/time for creating the sequence. It has to be in the format indicated by date.fmt.

`date.fmt` character indicating the format in which the date/time objects are stored in `from` and `to`, e.g. `%Y-%m-%d %H:%M`. See `format` in `as.Date`.
ONLY required when `class(dates)=="factor"` or `class(dates)=="numeric"`.

`out.type` Character indicating the type of result that is given by this function. Valid values are:
1) `seq` : a vector of class Date with all the days between the two dates, OR
2) `nmbr`: a single numeric value with the amount of days between the two dates.

`tz` specification of the desired time zone yo be used. System-specific (see time zones), but `""` is the current time zone, and `"GMT"` (the default value) is UTC (Universal Time, Coordinated). See `Sys.timezone` and `as.POSIXct`.
This argument can be used when working with subdaily zoo objects to force using the local time zone instead of GMT as time zone.

Value

Depending on the value of `out.type`, it returns:
1) a vector of class `c("POSIXct" "POSIXt")` with all the hours between `from` and `to` (both of them included), OR
2) the amount of hours between the two date/time objects

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

`dip`, `mip`, `yip`, `diy`, `timeBasedSeq`

Examples

```
## Sequence of hours between "1961-01-01 00:00" and "1961-01-10 00:00", giving the
## starting and ending date/time objects with hours and skipping the minutes (default)
hip("1961-01-01 00", "1961-12-31 00")

## Sequence of hours between "1961-01-01 00:00" and "1961-01-10 00:00", giving the
## starting and ending date/time objects only with hours and minutes(skipping the minutes)
hip("1961-01-01 00:00", "1961-12-31 00:00", date.fmt="%Y-%m-%d %H:%M")

## Number of hours between the 10:00 AM of "1961-Jan-02" and the 11:00 AM of "1961-Jan-01",
## using "%d/%m/%Y" as date/time format.
hip("01/01/1961 10", "02/01/1961 11", date.fmt= "%d/%m/%Y %H", out.type = "nmbr")
```

hydropairs *Visual Correlation Matrix*

Description

Visualization of a correlation matrix. On top the (absolute) value of the correlation plus the result of the cor.test as stars. On bottom, the bivariate scatterplots, with a fitted line. On the diagonal, an histogram of each variable.

Usage

```
hydropairs(x, dec = 3, use = "pairwise.complete.obs", method = "pearson",...)
```

Arguments

x	data.frame or matrix object with measurements at several locations. Each column of x represent values measured at different locations.
dec	decimal places to be used for showing the correlation values
use	See cor. An optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
method	See cor. A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated
...	further arguments passed to or from other methods, in particular it is used in the pairs function.

Value

On top	the (absolute) value of the correlation plus the result of the cor.test as points
On bottom	the bivariate scatterplots, with a fitted line
On diagonal	histograms (from pairs)

Note

Original idea taken from the R Graph Gallery (nowadays not available on its original link: http://addictedtor.free.fr/graphiques

Histogram panel was taken from the R help of the original pairs function

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

cor, pairs

Examples

```
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

## Visualizing the correlation among the monthly precipitation values
## of the first 3 gauging stations in 'EbroPPtsMonthly'.
## The first column of 'EbroPPtsMonthly' has the dates.
hydropairs(EbroPPtsMonthly[,2:4])
```

---

hydroplot *Hydrological time series plotting and extraction.*

Description

hydroplot: When x is a zoo object it plots (a maximum of) 9 graphs (lines plot, boxplots and/or histograms) of the daily, monthly, annual and/or seasonal time series.

sname2plot: When x is a data frame whose columns contain the time series of several gauging stations, it takes the name of one gauging station and plots the graphs described above.

Usage

```
hydroplot(x, ...)
sname2plot(x, ...)

## Default S3 method:
hydroplot(x, FUN, na.rm=TRUE, ptype="ts+boxplot+hist", pfreq="dma",
          var.type, var.unit="units", main=NULL, xlab="Time", ylab,
          win.len1=0, win.len2=0, tick.tstep="auto", lab.tstep="auto",
          lab.fmt=NULL, cex=0.3, cex.main=1.3, cex.lab=1.3, cex.axis=1.3,
          col=c("blue", "lightblue", "lightblue"),
          from=start(x), to=end(x), dates=1, date.fmt= "%Y-%m-%d", tz=NULL,
        stype="default", season.names=c("Winter", "Spring", "Summer", "Autumn"),
          h=NULL, ...)

## S3 method for class 'zoo'
hydroplot(x, FUN, na.rm=TRUE, ptype="ts+boxplot+hist", pfreq="dma",
          var.type, var.unit="units", main=NULL, xlab="Time", ylab,
          win.len1=0, win.len2=0, tick.tstep="auto", lab.tstep="auto",
          lab.fmt=NULL, cex=0.3, cex.main=1.3, cex.lab=1.3, cex.axis=1.3,
          col=c("blue", "lightblue", "lightblue"),
          from=start(x), to=end(x), dates=1, date.fmt= "%Y-%m-%d", tz=NULL,
        stype="default", season.names=c("Winter", "Spring", "Summer", "Autumn"),
```

```
          h=NULL, ...)

## S3 method for class 'data.frame'
hydroplot(x, FUN, na.rm=TRUE, ptype="ts+boxplot+hist", pfreq="dma",
          var.type, var.unit="units", main=NULL, xlab="Time", ylab,
          win.len1=0, win.len2=0, tick.tstep="auto", lab.tstep="auto",
          lab.fmt=NULL, cex=0.3, cex.main=1.3, cex.lab=1.3, cex.axis=1.3,
          col=c("blue", "lightblue", "lightblue"),
          from=start(x), to=end(x), dates=1, date.fmt= "%Y-%m-%d", tz=NULL,
        stype="default", season.names=c("Winter", "Spring", "Summer", "Autumn"),
          h=NULL, ...)

## Default S3 method:
sname2plot(x, sname, FUN, na.rm=TRUE, ptype="ts+boxplot+hist",
           pfreq="dma", var.type, var.unit="units", main=NULL,
           xlab="Time", ylab=NULL, win.len1=0, win.len2=0,
           tick.tstep="auto", lab.tstep="auto", lab.fmt=NULL,
           cex=0.3, cex.main=1.3, cex.lab=1.3, cex.axis=1.3,
           col=c("blue", "lightblue", "lightblue"),
           dates=1, date.fmt = "%Y-%m-%d", from=NULL, to=NULL, stype="default",
           season.names=c("Winter", "Spring", "Summer", "Autumn"),
           h=NULL, ...)

## S3 method for class 'zoo'
sname2plot(x, sname, FUN, na.rm=TRUE, ptype="ts+boxplot+hist",
           pfreq="dma", var.type, var.unit="units", main=NULL,
           xlab="Time", ylab=NULL, win.len1=0, win.len2=0,
           tick.tstep="auto", lab.tstep="auto", lab.fmt=NULL,
           cex=0.3, cex.main=1.3, cex.lab=1.3, cex.axis=1.3,
           col=c("blue", "lightblue", "lightblue"),
           dates=1, date.fmt = "%Y-%m-%d", from=NULL, to=NULL, stype="default",
           season.names=c("Winter", "Spring", "Summer", "Autumn"),
           h=NULL, ...)

## S3 method for class 'data.frame'
sname2plot(x, sname, FUN, na.rm=TRUE, ptype="ts+boxplot+hist",
           pfreq="dma", var.type, var.unit="units", main=NULL,
           xlab="Time", ylab=NULL, win.len1=0, win.len2=0,
           tick.tstep="auto", lab.tstep="auto", lab.fmt=NULL,
           cex=0.3, cex.main=1.3, cex.lab=1.3, cex.axis=1.3,
           col=c("blue", "lightblue", "lightblue"),
           dates=1, date.fmt = "%Y-%m-%d", from=NULL, to=NULL, stype="default",
           season.names=c("Winter", "Spring", "Summer", "Autumn"),
           h=NULL, ...)
```

Arguments

x	zoo, xts or data.frame object, with columns storing the time series of one or more gauging stations.

sname	ONLY required when x is a data frame. Character representing the name of a station, which have to correspond to one column name in x
FUN	ONLY required when var.type is missing AND pfreq != "o". Function that have to be applied for transforming from daily to monthly or annual time step (e.g., For precipitation FUN=sum and for temperature and flow ts, FUN=mean)
na.rm	Logical. Should missing values be removed before the computations?
ptype	Character indicating the type of plot that will be plotted. Valid values are: -) ts => only time series -) ts+boxplot => only time series + boxplot -) ts+hist => only time series + histogram -) ts+boxplot+hist => time series + boxplot + histogram
pfreq	Character indicating how many plots are desired by the user. Valid values are: -) dma : Daily, Monthly and Annual values are plotted -) dm : Daily and Monthly values are plotted -) ma : Monthly and Annual values are plotted -) o : Only the original zoo object is plotted, and ptype is changed to ts -) seasonal: Line and bloxplots of seasonal time series (see stype and season.names). When pfreq is seasonal, ptype is set to ts+boxplot
var.type	ONLY required when FUN is missing. character representing the type of variable being plotted. Used for determining the function used for computing the monthly and annual values when FUN is missing. Valid values are: -) Precipitation => FUN=sum -) Temperature => FUN=mean -) Flow => FUN=mean
var.unit	Character representing the measurement unit of the variable being plotted. ONLY used for labelling the axes (e.g., "mm" for precipitation, "C" for temperature, and "m3/s" for flow.)
main	Character representing the main title of the plot. If the user do not provide a title, this is created automatically as: main= paste(var.type, "at", sname, sep=" "),
xlab	A title for the x axis. See plot.
ylab	A title for the y axis. See plot.
win.len1	number of days for being used in the computation of the first moving average. A value equal to zero indicates that this moving average is not going to be computed.
win.len2	number of days for being used in the computation of the second moving average. A value equal to zero indicates that this moving average is not going to be computed.
tick.tstep	Character indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: -) days,

-) months,
-) years

lab.tstep — Character indicating the time step that have to be used for putting the labels on the time axis. Valid values are:
-) days,
-) months,
-) years

lab.fmt — Character indicating with the format to be used for the label of the axis. See format in as.Date. If not specified, it will try "%Y-%m-%d" when lab.tstep=="days", "%b" when lab.tstep=="month", and "%Y" when lab.tstep=="year".

cex — A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. (See par).

cex.main — The magnification to be used for main titles relative to the current setting of cex (See par).

cex.lab — The magnification to be used for x and y labels relative to the current setting of cex (See par).

cex.axis — The magnification to be used for axis annotation relative to the current setting of cex (See par).

col — A character vector with 3 elements, representing the colors to be used for plotting the lines of the ts, the boxplots, and the histograms, respectively.
When pfreq="o", only one character element is needed. See plot.default).

dates — ONLY required when x is a data frame. It is a numeric, factor or Date object indicating how to obtain the dates corresponding to the sname station.
If dates is a number (default), it indicates the index of the column in x that stores the dates
If dates is a factor, it is converted into Date class, using the date format specified by date.fmt
If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of element in x

date.fmt — Character indicating the format in which the dates are stored in *dates*, *from* and *to*. See format in as.Date.
ONLY required when class(dates)=="factor" or class(dates)=="numeric".

tz — character, with the specification of the time zone used for x, from, and to. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct.

If tz is NULL (the default), it is automatically set to the time zone used in time(x).

If tz is provided, it forces time(x) to be in the tome zone specified by tz, without modifying the the values (hours, minutes, seconds, etc).

A list of valid time zones can be obtained by calling the base function OlsonNames().

	This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for `from` and `to`. It should be used with caution, being well aware of the time zone of the data. See examples.
`from`	OPTIONAL, used for extracting a subset of values. Character indicating the starting date for the values to be extracted. It must be provided in the format specified by `date.fmt`.
`to`	OPTIONAL, used for extracting a subset of values. Character indicating the ending date for the values to be extracted. It must be provided in the format specified by `date.fmt`.
`stype`	OPTIONAL, only used when `pfreq=seasonal`. character, indicating which weather seasons will be used for computing the output. Possible values are: -) `default` => "winter"= DJF = Dec, Jan, Feb; "spring"= MAM = Mar, Apr, May; "summer"= JJA = Jun, Jul, Aug; "autumn"= SON = Sep, Oct, Nov -) `FrenchPolynesia` => "winter"= DJFM = Dec, Jan, Feb, Mar; "spring"= AM = Apr, May; "summer"= JJAS = Jun, Jul, Aug, Sep; "autumn"= ON = Oct, Nov
`season.names`	OPTIONAL, only used when `pfreq=seasonal`. character of length 4 indicating the names of each one of the weather seasons defined by `stype`.These names are only used for plotting purposes
`h`	OPTIONAL, only used when `pfreq=seasonal`, for plotting horizontal lines in each seasonal plot. numeric, with 1 or 4 elements, with the value used for plotting an horizontal line in each seasonal plot, in the following order: winter (DJF), spring (MAM), summer (JJA), autumn (SON).
`...`	further arguments passed to the `plot.zoo` and `axis` functions or from other methods.

Details

Plots of the daily/monthly/annual/seasonal values of the time series given as input.
Depending on the value of `pfreq`, daily, monthly, annual and/or seasonal time series plots, boxplots and histograms are produced.
Depending on the value of `ptype`, time series plots, boxplots and/or histograms are produced.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

`sname2ts`

Examples

```
#############
## Loading daily streamflows at the station Oca en Ona (Ebro River basin, Spain) ##
data(OcaEnOnaQts)
```

```
## 3 ts, 3 boxplots and 3 histograms
hydroplot(OcaEnOnaQts, FUN=mean, ylab= "Q", var.unit = "m3/s")

## only the original time series
hydroplot(OcaEnOnaQts, pfreq="o")

## only the year 1962 of the original time series
hydroplot(OcaEnOnaQts, pfreq="o", from="1962-01-01", to="1962-12-31")

## Not run:
## seasonal plots
hydroplot(OcaEnOnaQts, pfreq="seasonal", FUN=mean, stype="default")

## custom season names (let's assume to be in the Southern Hemisphere)
hydroplot(OcaEnOnaQts, pfreq="seasonal", FUN=mean,
          stype="default", season.names=c("Summer","Autumn", "Winter","Spring"))


#############
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

op <- par(mar=c(5.1, 4.1, 4.1, 2.1))
on.exit(par(op))

## Plotting seasonal precipitation at station "P9001"
sname2plot(EbroPPtsMonthly, sname="P9001", FUN=sum, dates=1, pfreq="seasonal",
           stype="default")

## Plotting the monthly and annual values of precipitation at station "P9001",
## stored in 'EbroPPtsMonthly'.
sname2plot(EbroPPtsMonthly, sname="P9001", var.type="Precipitation", dates=1,
           pfreq="ma")

## End(Not run)
```

`infillxy` *Infills NA values*

Description

Infill all the missing values (NA) in `x` with the corresponding values in `sim`.

Usage

```
infillxy(x, ...)
## Default S3 method:
infillxy(x, sim, ...)
## S3 method for class 'matrix'
```

```
infillxy(x, sim, ...)
## S3 method for class 'data.frame'
infillxy(x, sim, ...)
```

Arguments

x	numeric, data.frame or matrix in which some values are missing (NA).
sim	numeric, data.frame or matrix, with the same dimension of x, which contains the values that will be used for infilling the missing (NA) values in x
...	further arguments passed to or from other methods.

Details

It gives as a result an object of the same dimension of x, in which all the NA values were infilled with the corresponding values of sim.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

Examples

```
obs <- c(1, NA, 3, 4, NA, 5)
sim <- rep(2, 6)

## Filling in the missing values in 'x' with the corresponding values in 'sim'
infillxy(x=obs, sim)
```

---

isComplete *Is Complete?*

Description

Generic function for identifying whether a zoo object has a regular time frequency without missing values from the first one to the last one.

Usage

```
isComplete(x, ...)

## S3 method for class 'zoo'
isComplete(x, tz, out.type=c("single", "all"), verbose=TRUE, ...)
```

Arguments

x	zoo object with a time attribute that should have a regular time frequency (e.g., hourly, daily, monthly, annual).
tz	character, indicating the time zone in which the date of x is located. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(x). If tz is provided, it forces time(x) to be in the tome zone specified by tz, without modifying the the values (hours, minutes, seconds, etc). A list of valid time zones can be obtained by calling the base function OlsonNames(). This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for from and to. It should be used with caution, being well aware of the time zone of the data. See examples.
out.type	Character indicating the type of result that is given by this function. Valid values are: -) single => only a logical output is returned. See details in the Value section. -) all => a list with 3 elements is returned. See details in the Value section.
verbose	logical; if TRUE, informative messages are shown in the screen. When verbose=TRUE and x has some missing temporal layers, the DateTime(s) with missing layer(s) are shown in the screen.
...	further arguments passed to or from other methods.

Value

If out.type="single"
: (default value), it returns a logical value identifying whether x has a regular time frequency with complete time layers from its first layer to the last one.

If out.type="all"
: it returns a list with the following three elements:

-) isComplete: logical value identifying whether x has a regular time frequency with complete time layers from its first layer to the last one.

-) NumberMissingDT: integer indicating the amount of missing layers from the first layer of x to the last one.

-) missingDateTimes: Numeric, Date, or POSIXct vector showing the DateTime of the missing layers in x.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

window

Examples

```
############
# EXAMPLE 1: Hourly zoo object, with a POSIXct time attribute
############

# Loading the HOURLY streamflow data for Karamea at Gorge
data(KarameaAtGorgeQts)
x.h <- KarameaAtGorgeQts

# Checking whether 'x' is complete or not
isComplete(x.h)

############
## EXAMPLE 2: Daily object, with a Date time attribute
############

# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x.d <- SanMartinoPPts

# Checking whether 'x' is complete or not
isComplete(x.d)

# Removing two values and testing it again
x.d <-x.d[-c(3,5)]
isComplete(x.d)


############
# EXAMPLE 3: Weekly zoo object, with a Date time attribute
############

# Creating a dummy weekly object with a Date time attribute
x.w           <- x.d[1:12]
time(x.w) <-  seq(from=as.Date("2001-01-01"), to=as.Date("2001-03-19"),
                  by="1 week")

# Checking whether 'x.w' is complete or not
isComplete(x.w)

# Removing two values and testing it again
x.w2 <- x.w[-c(3,5)]
isComplete(x.w2)


############
# EXAMPLE 4: Monthly zoo object, with a yearmon time attribute
############
```

```
# Transforming the daily object into a monthly one
x.m1 <- daily2monthly(x.d, FUN=sum)

# Checking whether 'x.m1' is complete or not
isComplete(x.m1)

# Removing two values and testing it again
x.m12 <- x.m1[-c(3,5)]
isComplete(x.m12)


############
# EXAMPLE 5: Monthly zoo object, with a Date time attribute
############

# Creating a monthly object with a Date time attribute
x.m2       <- x.m1
time(x.m2) <-  as.Date( paste0( format(zoo::as.yearmon( time(x.m1) ), "%Y-%m"), "-01") )

# Checking whether 'x.m2' is complete or not
isComplete(x.m2)

# Removing two values and testing it again
x.m22 <- x.m2[-c(3,5)]
isComplete(x.m22)


############
# EXAMPLE 6: Annual zoo object, with a Date time attribute
############

# Creating a dummy annual object with Date time attribute
x.a1       <- x.m1[1:12]
time(x.a1) <- hydroTSM::yip("2001-01-01", "2012-01-01")

# Checking whether 'x.a1' is complete or not
isComplete(x.a1)

# Removing two values and testing it again
x.a12 <- x.a1[-c(3,5)]
isComplete(x.a12)


############
# EXAMPLE 7: 3-hourly zoo object, with a POSIXct time attribute
############

# Creating a dummy 3-hourly object with a POSIXct time attribute
x.3h       <- x.m1[1:12]
time(x.3h) <- seq(from=as.POSIXct("2001-01-01 00:00:00", tz="UTC"),
                  to=as.POSIXct("2001-01-02 09:00:00", tz="UTC"),
                  by="3 hours")
```

```
# Checking whether 'x.3h' is complete or not
isComplete(x.3h)

# Removing two values and testing it again
x.3h2 <- x.3h[-c(3,5)]
isComplete(x.3h2)


############
# EXAMPLE 8: 8-day zoo object, with a Date time attribute
############

# Creating a dummy 8-day object with a Date time attribute
x.8d       <- x.m1[1:12]
time(x.8d) <- seq(from=as.Date("2001-01-01"), to=as.Date("2001-03-31"), by="8 days")

# Checking whether 'x.8d' is complete or not
isComplete(x.8d)

# Removing two values and testing it again
x.8d2 <- x.8d[-c(3,5)]
isComplete(x.8d2)


############
# EXAMPLE 9: 3-monthly zoo object, with a Date time attribute
############

# Creating a dummy 3-monthly object with a Date time attribute
x.3m       <- x.m1[1:12]
time(x.3m) <- seq(from=as.Date("2001-01-01"), to=as.Date("2003-12-01"), by="3 months")

# Checking whether 'x.3m' is complete or not
isComplete(x.3m)

# Removing two values and testing it again
x.3m2 <- x.3m[-c(3,5)]
isComplete(x.3m2)
```

`istdx` *Inverse Standarization*

Description

This function back transforms a standarized vector/matrix `z` into their original values, i.e., re-scales all the values in the [0,1] interval to the original range of values `z = re-scale(x) = x*[ xmax - xmin ] + xmin`.

Usage

```
istdx(x, ...)
## Default S3 method:
istdx(x, xmin, xrange, ...)
```

Arguments

x	standarized vector or matrix to be re-scaled, all the values have to be in the range [0,1]
xmin	numeric with the minimum value(s) in the original x -) if x is a vector, xmin has to be a real -) if x is a matrix/data.frame, xmin has to be a vector, with the minimum values for each column of the original x. In this case, the vector of minimums can be obtained as: `xmin <- apply(x, 2, min, na.rm=TRUE)`

xrange	numeric with the range of value(s) in the original x
	-) if x is a vector, xrange has to be a real
	-) if x is a matrix/data.frame, xrange has to be a vector, with the range of values for each column of the original x. In this case, the vector of ranges can be obtained as:
	`xrange <- apply(x, 2,range, na.rm=TRUE)`
	`xrange <- apply(xrange, 2, diff, na.rm=TRUE)`

...	further arguments passed to or from other methods

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

stdx, scale

Examples

```
## Loading daily streamflows at the station Oca en Ona (Ebro River basin, Spain) ##
data(OcaEnOnaQts)
x <- OcaEnOnaQts

## Computing xmin and the range of 'x'
xmin <- min(x, na.rm=TRUE)
r <- diff(range(x, na.rm=TRUE))

## Standarized variable
s <- stdx(x)

## Inverse of the standarized variable
si <- istdx(s, xmin, xrange=r)

## 'si' and 'x' should be the same
summary(x-si)
```

```
###########
### Standarizing a subset of the stations 9 to 12 in 'EbroPPtsMonthly'

## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

pp <- EbroPPtsMonthly[1:70,10:13]
xmin   <- apply(pp, 2, min, na.rm=TRUE)
xrange <- apply(pp, 2, range, na.rm=TRUE)
xrange <- apply(xrange, 2, diff, na.rm=TRUE)

## Standarized variable
s <- stdx(as.matrix(pp))

## Inverse of the standarized variable
si <- istdx(s, xmin, xrange)

## 'si' and 'pp' should be the same
summary(pp - si)
```

---

izoo2rzoo *Irregular Zoo -> Regular Zoo*

Description

It takes an irregular zoo object (with non-existing values for some dates/times) and converts it into a regularly spaced zoo object within the time period defined by `from` and `to`, by filling the missing dates with 'NA'

Usage

```
izoo2rzoo(x, ...)

## Default S3 method:
izoo2rzoo(x, from= start(x), to= end(x),
                  date.fmt, tstep, tz,
                  na.action=c("keep", "linear", "spline"), ... )

## S3 method for class 'zoo'
izoo2rzoo(x, from= start(x), to= end(x),
                  date.fmt, tstep, tz,
                  na.action=c("keep", "linear", "spline"), ... )
```

Arguments

x	irregular zoo object (vector or matrix) representing a time series (very likely read with some user-defined procedure, and with some missing values for particular days/months/years)
from	Character indicating the starting date for creating the regularly spaced zoo object. The default value corresponds to the date of the first element of x It has to be in the format indicated by date.fmt.
to	Character indicating the ending date for creating the regularly spaced zoo object. The default value corresponds to the date of the last element of x It has to be in the format indicated by date.fmt.
date.fmt	character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See 'Details' section in strptime. By default, date.fmt is missing, and it is automatically set to %Y-%m-%d when time(x) is Date object, and set to %Y-%m-%d %H:%M:%S when x is a sub-daily zoo object.
tstep	character, indicating the time step used for creating the time sequence going from from to to that will be used as time(x) Valid values are (but not limited to) hours, days, months, years. By default, tstep is missing, and it is automatically set to "minutes" when sfreq(x) is min, to "hours" when sfreq(x) is hourly, to "days" when sfreq(x) is daily, to "weeks" when sfreq(x) is weekly, to "months" when sfreq(x) is monthly, to "quarters" when sfreq(x) is quarterly, and to "years" when sfreq(x) is annual.
tz	character, with the specification of the time zone used for x, from, and to. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(x). If tz is provided, it forces time(x) to be in the tome zone specified by tz, without modifying the the values (hours, minutes, seconds, etc). A list of valid time zones can be obtained by calling the base function OlsonNames(). This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for from and to. It should be used with caution, being well aware of the time zone of the data. See examples.
na.action	character indicating whether to keep 'NA' values in x (default) or interpolate them with linear or spline interpolation using the na.approx or the na.spline function of the zoo package.
...	further arguments passed to or from other methods

Details

If the full time period of x is a subset of the time period defined by from and to, the time period of the resulting zoo is the one defined by from and to, assigning 'NA' to all the dates in which x does not have a value.

Value

a regularly spaced zoo object, with values given by `x` and time stamps going from `from` to `to` at intervals defined by `tsteps`.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

zoo, vector2zoo, as.POSIXct, Sys.timezone

Examples

```
##
## Example 1: Adding NA for February 29th to an existing zoo object

# dummy values and dates (February 29th is not present !)
x <- 1:9
dates <- c("1964-02-25", "1964-02-26", "1964-02-27", "1964-02-28", "1964-03-01",
           "1964-03-02", "1964-03-03", "1964-03-04", "1964-03-05")

# From 'character' to 'Date' class
dates <- as.Date(dates)

## From 'numeric' to 'zoo' class
( x <- zoo(x, dates) ) # Feb 29th is still not present in 'x'
## checking the length of 'x'
length(x) # 9 elements (there is no data for Feb 29th)

## Adding a missing value (NA in this case) for Feb 29th
( y <- izoo2rzoo(x) )

## checking the new length
length(y) # 1 element more than the original 'x' (thre is an NA value in Feb 29th)


##
## Example 2: Extending the original 'x' object from February 1st to the end of March,
#             assigning 'NA' to the days in which 'x' do not have a value.
( y <- izoo2rzoo(x, from="1964-02-01", to="1964-03-31") )


##
## Example 3: Working with a zoo matrix with two identical 'x' time series,
##            from 1964-02-25 to 1964-03-05
( Y <- cbind(x,x) )

# Adding a missing value (NA in this case) for Feb 29th in all the columns of Y
( rY <- izoo2rzoo(Y) )
```

```
##
## Example 4: Working with hourly data, from 01:00 to 10:00 UTC on 12th December 2000
dates  <- ISOdatetime(year=2000, month=12, day=12, hour=1:10, min=0, sec=0, tz="UTC")
values <- 1:10
x      <- zoo(values, dates)

# removing four values in 'x', from 02:00 to 05:00, i.e., they will not be present
# anymore in 'x' at all, not even NA !)
x <- x[-c(2:5)]
time(x)
length(x)

# Adding missing values (NA in this case) from 02:00 to 05:00
y <-  izoo2rzoo(x)
time(y)
length(y)


##
## Example 5: Extending hourly data to a DateTime before 'start(x)',
##            specifying only the date.
##            Time of 'x' is in local time zone (tz="") instead of UTC
dt <- hip("2021-01-01 00:00:00", "2021-01-01 20:00:00", tz="")
x  <- zoo(0:20, dt)
(y  <- izoo2rzoo(x, from="2020-12-31") )# 00:00:00 is ommited
(time(y))


##
## Example 6: Extending hourly data to a DateTime before 'start(x)',
##            specifying date and time.
##            Time of 'x' is in local time zone (tz="") instead of UTC
dt <- hip("2021-01-01 00:00:00", "2021-01-01 20:00:00", tz="")
x  <- zoo(0:20, dt)
( y  <- izoo2rzoo(x, from="2020-12-31 20:00:00") )


##
## Example 7: Extending hourly data to a DateTime before 'start(x)',
##            specifying date and time, and forcing UTC to be the time zone.
##            Time of 'x' is in local time zone (tz="") instead of UTC, but
##            it will be treated as UTC by using the 'tz' argument
dt <- hip("2021-01-01 00:00:00", "2021-01-01 20:00:00", tz="")
x  <- zoo(0:20, dt)
(time(x))
(y  <- izoo2rzoo(x, from="2020-12-31 20:00:00", tz="UTC") )# 00:00:00 is ommited
(time(y))

##
## Example 8: Extending hourly data to a DateTime after 'end(x)',
##            specifying date and time.
##            Time of 'x' is in local time zone (tz="") instead of UTC
dt <- hip("2021-01-01 00:00:00", "2021-01-01 20:00:00", tz="")
```

```
x  <- zoo(0:20, dt)
( y  <- izoo2rzoo(x, to="2021-01-02 12:00:00") )


##
## Example 9: Extending hourly data to a DateTime before 'start(x)'.
##            Note that the 'tz' argument can be ommited in the 'hip' function,
##            because by default it assumes UTC as time zone
dt <- hip("2021-01-01 00:00:00", "2021-01-01 20:00:00", tz="UTC")
x  <- zoo(0:20, dt)
( y  <- izoo2rzoo(x, from="2020-12-31 20:00:00", tz="UTC") )


##
## Example 10: Extending hourly data to a date before 'start(x)'. However, hourly 'x'
##            values are given at HH:15:00 hours instead of HH:00:00 hours.

## Loading the time series of hourly streamflows for the station Karamea at Gorge
## Time Zone for 'KarameaAtGorgeQts' data is 'UTC' (see ?KarameaAtGorgeQts), but it will
## be tr4ated as 'Pacific/Auckland' (Zealand Standard Time) for this example
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

# Subsetting 'x' to its first day only
# (01/Jan/1980 08:15:00 - 01/Jan/1980 23:15:00)
x <- window(x, end="1980-01-01 23:59:00")

# Adding NA hourly data since 1979-12-31 21:15:00
izoo2rzoo(x, from="1979-12-31 21:15:00", tz="Pacific/Auckland")
```

KarameaAtGorgeQts *Karamea at Gorge, time series of hourly streamflows*

Description

Time series with hourly streamflows for the Karamea River(New Zealand) measured at the gauging station "Gorge", for the period 01/Jan/1980 to 31/Dec/1985.

Station Number: 95102, Easting Coordinate (NZMG): 2444629.0, Northing Coordinate (NZMG): 5994427.0, Catchment Area (km2): 1160.0.

In November 25th, 2023 (for hydroTSM v0.7-0), the time zone of this data sete was changed from "none" (i.e., your local time zone was used every time you loaded this dataset) to GMT+12, in order to avoid missing datetimes at times where daylight saving time ocurred. However, due to CRAN requirements, in January 2024 (for hydroTSM v0.7-1), the time zone of this data sete was changed from "GMT+12" to UTM, in order to avoid missing datetimes at times where daylight saving time ocurred. After the previous change, the initial DateTime of this dataset changed from "1980-01-01 08:15:00 -03" (in my local computer) to "1979-12-31 20:15:00 UTC" (everywhere).

Only two NA elements were removed from the original dataset: the first one and the last one.

Usage

```
data(KarameaAtGorgeQts)
```

Format

zoo object.

Source

Provided by the National Institute of Water and Atmospheric Research https://niwa.co.nz/, thanks to the gentle collaboration of Shailesh Singh

These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

---

ma *Moving Average*

Description

Generic function for computing a moving (sliding) average of ts.

Usage

```
ma(x, ...)

## Default S3 method:
ma(x, win.len, FUN = mean, ...)

## S3 method for class 'zoo'
ma(x, win.len, FUN = mean, ...)
```

Arguments

x	ts or zoo object.
win.len	number of terms that will be considered in the mean. It have to be odd
FUN	Function that have to be applied for computing the moving average. Usually, FUN MUST be mean
...	further arguments passed to or from other methods.

Value

a vector with the moving average termns. The length of the resulting vector is the same of x, but the first and last (win.len-1)/2 elements will be NA.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

Examples

```
## Loading daily streamflows at the station Oca en Ona (Ebro River basin, Spain) ##
data(OcaEnOnaQts)
x <- OcaEnOnaQts

## Daily to Monthly ts
m <- daily2monthly(x, FUN=mean, na.rm=FALSE)

# Plotting the monthly values
plot(m, xlab="Time")

## Plotting the annual moving average in station 'x'
lines(ma(m, win.len=12), col="blue")
```

MaquehueTemuco *San Martino, ts of daily precipitation.*

Description

Daily time series of precipitation, maximum and minimun air temperature at station Maquehue Temuco Ad. (DMC_ID:380013), Araucania Region, Chile (Lat:-38.770, Lon:-72.637), with data from 01/Jan/1950 to 31/Dec/2015 (including some gaps).

Usage

```
data(MaquehueTemuco)
```

Format

zoo matrix with 3 columns:
-) *pcp*: daily precipitation, [mm/day].
-) *tmx*: daily maximum air temperature, [degree Celsius].
-) *tmn*: daily minimum air temperature, [degree Celsius].

Source

Provided by Center for Climate and Resilience Research, Universidad de Chile, Santiago, Chile (CR2, https://dataclima.cr2.cl/bases_de_datos?database_db_type_id=5. Original data available at the Chilean Meteorological Organization (DMC, https://climatologia.meteochile.gob.cl/application/diariob/visorDeDatosEma/380013). Last accessed [May 2025]).
These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

`matrixplot` *Matrixplot*

Description

Plots a color matrix, representing the values stored in `x`.
Originally, it was thought to represent the amount of days with information per year in a set of gauging stations, but it can be used for plotting the information stored in any two dimensional matrix.

Usage

```
matrixplot(x, ColorRamp="Days", ncolors = 10, main = "",
           cuts,  cuts.dec=2, cuts.labels,
           cuts.style=c("equal", "pretty", "fixed", "sd", "quantile",
                        "kmeans", "bclust", "fisher"),
           legend.cex=0.8,  legend.title="",  legend.title.cex=1.5,
           legend.fontsize=15, ...)
```

Arguments

`x`	matrix to be plotted. Originally: -) Each column of `x` represent a different gauging station, and it stores the values measured on it -) Each row of `x` represent the years, and they stores the amount of days with information in each station
`ColorRamp`	Character or function defining a personalized color ramp for plotting the maps. Valid character values are in `c("Days", "Precipitation", "Temperature", "PCPAnomaly", "PCPAnomaly2" "TEMPAnomaly", "TEMPAnomaly2", "TEMPAnomaly3")`.
`ncolors`	numeric, indicating the number of color intervals that will be used for representing the information content of `x`.
`main`	Main title for the plot
`cuts`	Numeric, indicating the values used to divide the range of 'x' in the legend of colours. If not provided, it is automatically selected as a function of 'lenght(ColorRamp)'.
`cuts.dec`	Number of decimal places used to present the numbers that divide the range of 'x' in the legend of colours.
`cuts.labels`	Character indicating the label to be used in the ccolour legend for each one of the values defined by 'cuts'. If not provided, as.character(cuts)' is used.
`cuts.style`	character, indicating how to finding class intervals for continuous numerical variables for choosing colours to be used in the figure. See `classIntervals`
`legend.cex`	character expansion factor *relative* to current `par("cex")` used for the legend text.
`legend.title`	text to be displayed above the legned of colours (e.g., showing the measurement units of the raster being displayed).

`legend.title.cex`
: expansion factor(s) for the legend title. Currently it is not used. See `legend.fontsize` instead.

`legend.fontsize`
: size of text (in points) used in the legend title (e.g., showing the measurement units of the raster being displayed).

`...`
: further arguments passed to `levelplot` function (**lattice** package) or from other methods

Note

Adapted from a not available web page (http://www2.warwick.ac.uk/fac/sci/moac/currentstudents/peter_cock/r/matrix_conto

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

`dwi`, `classIntervals`

Examples

```
## Loading the SanMartino precipitation data
data(SanMartinoPPts)

# Selecting only the values up to Dec/1960
x <- window(SanMartinoPPts, end=as.Date("1960-12-31"))

## Daily zoo to monthly zoo
m <- daily2monthly(x, FUN=sum, na.rm=TRUE)

# Creating a data.frame with monthly values per year in each column
M <- matrix(m, ncol=12, byrow=TRUE)
colnames(M) <- month.abb
rownames(M) <- unique(format(time(m), "%Y"))

# Plotting the monthly precipitation values from 1921 to 1960.
# Useful for identifying dry/wet months
matrixplot(M, ColorRamp="Precipitation",
           main="Monthly precipitation at San Martino st., [mm/month]")
```

---

mip *Months in Period*

Description

Given any starting and ending dates, it generates:
1) a vector of class 'Date' with all the months between the two dates (both of them included), OR
2) the amount of months between the two dates

Usage

```
mip(from, to, date.fmt = "%Y-%m-%d", out.type = "seq")
```

Arguments

from	Character indicating the starting date for creating the sequence. It has to be in the format indicated by date.fmt.
to	Character indicating the ending date for creating the sequence. It has to be in the format indicated by date.fmt.
date.fmt	Character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See format in as.Date.
out.type	character indicating the type of result that is given by this function. Valid values are: -) seq : a vectorial sequence with all the months within the given year -) nmbr: the number of days in the vectorial sequence with all the months within the given year

Value

Depending on the value of out.type, it returns:
1) a vector of class Date with all the months between from and to (both of them included), OR
2) a single numeric value with the amount of months between the two dates.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

dip, diy, hip, yip

Examples

```
# Sequence of monthly dates between "1961-01-01" and "1961-12-31" ##
mip("1961-01-01", "1961-12-31")

## Computing the number of months between "1961-01-01" and "1965-06-30",
## with the date format  "%d-%m-%Y" ##
mip("01-01-1961", "30-06-1965", date.fmt= "%d-%m-%Y", out.type = "nmbr")
```

monthlyfunction *Monthly Function*

Description

Generic function for obtaining 12 monthly values of a zoo object, by applying any R function to ALL the values in the object belonging to each one of the 12 calendar months (Jan...Dec).

Usage

```
monthlyfunction(x, ...)

## Default S3 method:
monthlyfunction(x, FUN, na.rm = TRUE, ...)

## S3 method for class 'zoo'
monthlyfunction(x, FUN, na.rm=TRUE,...)

## S3 method for class 'data.frame'
monthlyfunction(x, FUN, na.rm = TRUE, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", verbose = TRUE, ...)

## S3 method for class 'matrix'
monthlyfunction(x, FUN, na.rm = TRUE, dates=1,
        date.fmt = "%Y-%m-%d", out.type = "data.frame", verbose = TRUE, ...)
```

Arguments

x	zoo, xts, data.frame or matrix object, with daily or monthly time series. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that will be applied to ALL the values in x belonging to each one of the 12 months of the year (e.g., FUN can be some of mean, sum, max, min, sd).
na.rm	Logical. Should missing values be removed? -) TRUE : the monthly values and FUN are computed considering only those values in x different from NA -) FALSE: if there is AT LEAST one NA within a month, the corresponding monthly value will be NA
dates	It is only used when x is not a zoo object. numeric, factor, Date indicating how to obtain the dates. If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into 'Date' class, using the date format specified by date.fmt

	If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of elements in x
date.fmt	It is only used when x is not a zoo object. character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	It is only used when x is a matrix or data.frame. Character defining the desired type of output. Valid values are: -) data.frame: a data.frame, with 12 columns representing the months, and as many rows as gauging stations are included in x -) db : a data.frame, with 4 columns will be produced. Useful for a posterior boxplot The first column ('StationID') will store the ID of the station, The second column ('Year') will store the year, The third column ('Month') will store month, The fourth column ('Value') will contain the monthly value corresponding to the three previous columns.
verbose	Logical; if TRUE, progress messages are printed
...	further arguments passed to or from other methods

Value

When x is a zoo object, a numeric vector with 12 elements representing the computed monthly value for each month.
When x is a data.frame which columns represent measurements at different gauging stations, the resulting object is a data.frame with 12 columns and as many rows as gauging stations are in x, each row storing the computed 12 monthly value for each gauging station.

Note

Due to the fact that FUN is applied over all the elements in x belonging to a given calendar month, its result will depend on the sampling frequency of x and the type of function provided by FUN (**special attention have to be put when** FUN=sum)

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

annualfunction, seasonalfunction, dm2seasonal, daily2monthly, daily2annual

Examples

```
## Loading daily streamflows (3 years) at the station
## Oca en Ona (Ebro River basin, Spain)
data(OcaEnOnaQts)
x <- OcaEnOnaQts
```

```
## Mean monthly streamflows at station 'x'
monthlyfunction(x, FUN=mean, na.rm=TRUE)


############################
## Boxplot of monthly values

## Daily to Monthly
m <- daily2monthly(x, FUN=mean, na.rm=TRUE)

## Median of the monthly values at the station
monthlyfunction(m, FUN=median, na.rm=TRUE)

## Vector with the three-letter abbreviations of the month names
cmonth <- format(time(m), "%b")

## Creating ordered monthly factors
months <- factor(cmonth, levels=unique(cmonth), ordered=TRUE)

## Boxplot of the monthly values
boxplot( coredata(m) ~ months, col="lightblue", main="Monthly streamflows, [m3/s]")


############################
############################
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)
x <- EbroPPtsMonthly

## Dates of 'x'
dates <- as.Date(x[,1])

## Monthly precipitation of all the stations in 'x'
## Not run:

## Sum of the monthly values in each station of 'x'
z <- zoo( x[, 2:ncol(x)], dates)

# Amount of years in 'x' (needed for computing the average)
nyears <- yip(from=start(z), to=end(z), out.type="nmbr" )

m <- monthlyfunction(z, FUN=sum)


## Another way of computing the sum of the monthly values in each station of 'x'
## This way is usefult for posteriori boxplots
m2 <- monthlyfunction(x, FUN=sum, dates=1, out.type="db")

## Average monthly precipitation in each station of 'x'
m2$Value <- m2$Value / nyears

## Creating monthly factors
m2$Month <- factor(m2$Month, levels=month.abb)
```

```
## boxplot of the monthly values in all stations
boxplot(Value ~ Month, m2, col="lightyellow", main="Monthly Precipitation, [mm/month]")

## End(Not run)
```

`OcaEnOnaQts` *Oca in "Ona" (Q0931), time series of daily streamflows.*

Description

Time series with daily streamflows of the Oca River (subcatchment of the Ebro River basin, Spain) measured at the gauging station "Ona" (Q093), for the period 01/Jan/1961 to 31/Dic/1963

Usage

```
data(OcaEnOnaQts)
```

Format

zoo object.

Source

Downloaded from the web site of the Confederacion Hidrografica del Ebro (CHE) http://www.chebro.es/ (original link http://oph.chebro.es/documentacion/CaudalEA/CaudalEA.htm, last accessed [March 2010]).

These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

`plot_pq` *Plot precipitation and streamflow time series in the same figure.*

Description

Given a time series of precipitation and streamflow, this function plots the two time series in the same figure, streamflows as a normal time series and precipitation as bars coming from the upper part of the plotting window.

Usage

```
plot_pq(p, ...)

## S3 method for class 'zoo'
plot_pq(p, q, ptype=c("original", "monthly"),
                na.fill=c("remove", "linear", "spline"),
                from=start(p), to=end(p), date.fmt=NULL, tz=NULL,
                main=ifelse(ptype=="original", "Precipitation and Streamflows",
                            "Monthly Precipitation and Streamflows"),
                xlab=ifelse(ptype=="original", "Time", "Month"),
                ylab=c("P, [mm]", "Q, [m3/s]"),
                p.col=ifelse(ptype=="original", "blue", "lightblue"),
                q.col=ifelse(ptype=="original", "black", "blue"),
                leg.title="", leg.text=c("P", "Q"),
                q.pch=16, q.cex=0.3,

                start.month=1,
                plot.p.probs=TRUE, p.probs=c(0.25, 0.75),
                p.alpha=0.8,
                plot.q.probs=TRUE, q.probs=c(0.25, 0.75),
                q.probs.col="lightskyblue1", q.probs.alpha=0.8,
                labels=TRUE, labels.cex=0.8,
                labels.p.dy=NULL,
                labels.q.dx=c(rep(-0.2,6), rep(0.2,6)),
                labels.q.dy=rep(median(q, na.rm=TRUE)*1.3, 12),

                ...)
```

Arguments

p	zoo object with precipitation time series, with any time frequency.
q	zoo object with streamflow time series, with any time frequency.
ptype	Character indicating the type of plot to be produced. Valid values are: -) original => a time series plot with precipitation in the upper panel (as bars from the time axis) and streamflows in the lower panel (as dotted lines). -) monthly => a plot with mean monthly values of precipitation in the upper panel (as bars from top to bottom) and streamflows in the lower panel (as bars from bottom to up). Quantiles of precipitation and streamflows are also plotted depending on the values defined in p.probs and q.probs, respectively.
na.fill	Character indicating how to fill any NA present in p or q. -) remove => NAs are not plotted -) linear => NAs are removed by linear interpolation, using na.approx -) spline => NAs are removed by spline interpolation, using na.spline
from	Character indicating the starting date for subsetting p and q. The default value corresponds to the date of the first element of p It has to be in the format indicated by date.fmt.

to	Character indicating the ending date for subsetting p and q. The default value corresponds to the date of the last element of p It has to be in the format indicated by date.fmt.
date.fmt	character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See 'Details' section in strptime. By default, date.fmt is missing, and it is automatically set to %Y-%m-%d when time(p) is Date object, and set to %Y-%m-%d %H:%M:%S when x is a sub-daily zoo object.
tz	character, with the specification of the time zone used for from, to. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(p). This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for from and to. It should be used with caution, being well aware of the time zone of the data. See examples.
main	The main title (on top) of the figure.
xlab	a title for the x axis. See title.
ylab	a two-element title for the y axis: see title. The first element is used for the right y axis (i.e., for precipitation). The second element is used for the left y axis (i.e., for streamflows).
p.col	character, representing the colors to be used for plotting the precipitation time series.
q.col	character, representing the colors to be used for plotting the streamflow time series.
leg.title	a character string or length-one expression giving a title to be placed at the top of the legend. legend.
leg.text	a two-element character to appear in the legend placed at the bottom of the figure. The first element is used for precipitation and the second element is used for streamflows.
q.pch	numeric, representing the symbols used for plotting the streamflow time series.
q.cex	a numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex"). See plot.default
start.month	numeric in [1:12] indicating the starting month for the monthlycurve. Numeric values in [1, 12] represents months in [January, December]. By default start.month=1.
plot.p.probs	logical used to decide whether to show lower and upper uncertainty bounds for each one of the 12 monthly precipitation values. By default plot.p.probs=TRUE. When plot.p.probs=TRUE the p.probs argument is used to define the values of the lower and upper uncertainty bounds.

p.probs — numeric of length 2. It defines the quantile values used to compute the lower and upper uncertainty bounds for each one of the 12 monthly precipitation values. This uncertainty bounds are drawn as vertical lines over the bars used to plot the 12 monthly precipitation values.
By default p.probs=c(0.25, 0.75), which indicates that the quantiles 0.25 and 0.75 are used to compute the lower and upper uncertainty bounds for each one of the 12 monthly precipitation values. If p is a (sub)daily zoo object, it is first aggregated into monthly values using mean, and then the p.probs quantiles are computed over all the monthly values belonging to a calendar month.

p.alpha — numeric of length 1, with the factor used to modify the opacity of p.col. Typically in [0,1], with 0 indicating a completely transparent colour and 1 indicating no transparency.

plot.q.probs — logical used to decide whether to show uncertainty bands around each one of the 12 monthly average or median streamflow values. By default plot.q.probs=TRUE. When plot.q.probs=TRUE the q.probs argument is used to define the values of the lower and upper uncertainty bands.

q.probs — numeric of length 2. It is used to define quantile values used to compute the lower and upper uncertainty bands around each one of the 12 monthly average or median streamflow.
If q is a (sub)daily zoo object, it is first aggregated into monthly values using FUN, and then the q.probs quantiles are computed over all the monthly values belonging to a calendar month..
By default q.probs=c(0.25, 0.75), which indicates that the quantiles 0.25 and 0.75 are used to compute the lower and upper uncertainty bounds for each one of the 12 monthly average or median values. If q is provided and is a (sub)daily zoo object, it is first aggregated into monthly values using FUN, and then the q.probs quantiles are computed over all the monthly values belonging to a calendar month.

q.probs.col — character with the color used to plot the uncertainty bands around the average or median streamflow values.

q.probs.alpha — numeric of length 1, with the factor used to modify the opacity of q.probs.col. Typically in [0,1], with 0 indicating a completely transparent colour and 1 indicating no transparency.

labels — logical. Should monthly streamflow values to be shown above the lines?. By default labels=TRUE.

labels.cex — numeric giving the amount by which plotting characters used to show the numeric values of monthly streamflow values are scaled relative to the default.

labels.p.dy — numeric of length 12 giving the amount of vertical coordinate positions that have to be used to vertically shift the labels of monthly precipitation values.
It is only used when labels=TRUE.
Lengths smaller than 12 are recycled and larger lengths are not used.

labels.q.dx — numeric of length 12 giving the amount of horizontal coordinate positions that have to be used to horizontally shift the labels of monthly streamflow values.
It is only used when labels=TRUE.
Lengths smaller than 12 are recycled and larger lengths are not used.

labels.q.dy	numeric of length 12 giving the amount of vertical coordinate positions that have to be used to vertically shift the labels of monthly streamflow values. It is only used when labels=TRUE. Lengths smaller than 12 are recycled and larger lengths are not used.
...	further arguments passed to or from other methods. Not used yet.

Details

Given a time series of precipitation and streamflow, this function plots the two time series in the same figure, streamflows as a normal time series and precipitation as bars coming from the upper part of the plotting window.

Value

A figure with the two time series in the same graphical area, streamflows as a normal time series and precipitation as bars coming from the upper part of the plotting window.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

hydroplot, climograph, fdc, fdcu, monthlyfunction

Examples

```
######################
## Ex1: Plotting precipitation and streamflows for the full time period of both
##      time series.
##      First, we load the daily P and Q time series for the Cauquenes en
##      El Arrayan catchment. P, [mm] is the first column and Q, [mm] is the
##      fifth column.

data(Cauquenes7336001)

# Subsetting to only the 1981-1990 temporal period
Cauquenes7336001 <- window(Cauquenes7336001, start="1981-01-01", end="1990-12-31")

p <- Cauquenes7336001[, 1]
q <- Cauquenes7336001[, 5]

## Plotting P and Q for the full time period of both time series
plot_pq(p=p, q=q)

## Not run:
  ######################
  ## Ex2: Plotting precipitation and streamflows only for a specific time period,
  ##      from April to December 2000.
  plot_pq(p, q, from="1985-04-01", to="1985-12-31")
```

```
  ######################
  ## Ex3: Plotting monthly values of precipitation and streamflows for the
  ##      full time period of both time series.
  plot_pq(p, q, ptype="monthly")

  ######################
  ## Ex4: Plotting monthly values of precipitation and streamflows for the
  ##      full time period of both time series, but using a hydrologic year
  ##      starting on April
  plot_pq(p, q, ptype="monthly", start.month=4)

## End(Not run)
```

`rm1stchar` *Remove First Character(s)*

Description

Deletes the first `n` character(s) of a character object.

Usage

```
rm1stchar(x, n = 1)
```

Arguments

`x`	Character, e.g, each element may represent the name of a single gauging station.
`n`	numeric, indicating the number of characters that have to be removed from the beginning of `x`

Value

character object of the same length as `x`.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

`substr`

Examples

```
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

# Getting the name of each gauging station.
names <- colnames(EbroPPtsMonthly)

# Removing the initial letter 'P' of the name of each gauging station.
rm1stchar(names)
```

SanMartinoPPts *San Martino, ts of daily precipitation.*

Description

Daily time series of precipitation, at station San Martino di Castrozza, Trento Province, Italy, with data from 01/Jan/1921 to 31/Dec/1990.

Usage

```
data(SanMartinoPPts)
```

Format

zoo object.

Source

Provided by MeteoTrentino, Trento, Italy (via prof. Riccardo Rigon).
These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

---

seasonalfunction *Seasonal Function*

Description

Generic function for applying any R function to a zoo object, in order to obtain 4 representative seasonal values.

Usage

```
seasonalfunction(x, ...)

## Default S3 method:
seasonalfunction(x, FUN, na.rm = TRUE, type="default", ...)

## S3 method for class 'zoo'
seasonalfunction(x, FUN, na.rm = TRUE, type="default", ...)

## S3 method for class 'data.frame'
seasonalfunction(x, FUN, na.rm = TRUE, type="default",
                         dates=1, date.fmt = "%Y-%m-%d",
                         out.type = "data.frame", verbose = TRUE, ...)

## S3 method for class 'matrix'
seasonalfunction(x, FUN, na.rm = TRUE, type="default",
                         dates=1, date.fmt = "%Y-%m-%d",
                         out.type = "data.frame", verbose = TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with daily or monthly time series. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that will be applied to ALL the values in x belonging to each one of the 4 weather seasons (e.g., FUN can be some of mean, max, min, sd).
na.rm	Logical. Should missing values be removed before the computations? -) TRUE : the monthly values are computed considering only those values in x different from NA (**very important when** FUN=sum) -) FALSE: if there is AT LEAST one NA within a month, the FUN and monthly values are NA
type	character, indicating which weather seasons will be used for computing the output. Possible values are: -) default => "winter"= Dec, Jan, Feb; "spring"= Mar, Apr, May; "summer"=Jun, Jul, Aug; "autumn"= Sep, Oct, Nov -) FrenchPolynesia => "winter"= Dec, Jan, Feb, Mar; "spring"= Apr, May; "summer"=Jun, Jul, Aug, Sep; "autumn"= Oct, Nov
dates	numeric, factor, Date indicating how to obtain the dates. If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into Date class, by using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of element in x

date.fmt	Character indicating the format in which the dates are stored in dates, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	Character defining the desired type of output. Valid values are: -) data.frame: a data.frame, with 4 columns representing the weather seasons, and as many rows as stations are included in x -) db : a data.frame, with 4 colums will be produced. Useful for a posterior boxplot The first column (StationID) will store the ID of the station, The second column (Year) will store the year, The third column (Season) will store the season, The fourth column (Value) will contain the seasonal value corresponding to that year and that station.
verbose	Logical; if TRUE, progress messages are printed
...	further arguments passed to or from other methods

Warning

The *FUN* value for the winter season (DJF) is computed considering the consecutive months of December, January and February. Therefore, if x starts in January and ends in December of any year, the winter value of the first year is computed considering only the January and February value of that year, whereas the December value of the first year is used to compute the winter value of the next year.

Note

FUN is applied to all the values of x belonging to each one of the four weather seasons, so the results of this function depends on the frequency sampling of x and the type of function given by FUN

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

dm2seasonal, time2season, monthlyfunction, annualfunction, extract

Examples

```
## Loading the SanMartino precipitation data
data(SanMartinoPPts)
x <- SanMartinoPPts

# Amount of years
nyears <- yip(from=start(x), to=end(x), out.type="nmbr")

## Mean annual precipitation.
# It is necessary to divide by the amount of years to obtain the mean annual value,
```

```
# otherwise it will give the total precipitation for all the 70 years
seasonalfunction(x, FUN=sum, na.rm=TRUE) / nyears

####################
### verification ####
# Mean winter (DJF) value
sum( extractzoo(x, trgt="DJF") ) / nyears

# Mean spring (MAM) value
sum( extractzoo(x, trgt="MAM") ) / nyears

# Mean summer (JJA) value
sum( extractzoo(x, trgt="JJA") ) / nyears

# Mean autumn (SON) value
sum( extractzoo(x, trgt="SON") ) / nyears
```

`sfreq` *Sampling Frequency*

Description

This function identifies the sampling frequency of a zoo object. It is wrapper to the `periodicity` function of the **xts** package.

Usage

```
sfreq(x, min.year = 1800)
```

Arguments

`x`	variable of type zoo, xts or ts, with AT LEAST 2 elements, AND with a (sub)hourly, hourly, daily, weekly, monthly, quarterly, or annual (yearly) sampling frequency.
`min.year`	integer used for a correct identification of the sampling frequency when `x` is an annual (yearly) time series.

Details

See further details in the `periodicity` function of the **xts** package.

Value

Character. Possible values are:
-) `minute` : indicating that the sampling frequency in `x` is sub-hourly
-) `hourly` : indicating that the sampling frequency in `x` is hourly
-) `daily` : indicating that the sampling frequency in `x` is daily

-) weekly : indicating that the sampling frequency in x is weekly
-) monthly : indicating that the sampling frequency in x is monthly
-) quarterly : indicating that the sampling frequency in x is quarterly
-) annual : indicating that the sampling frequency in x is annual

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

frequency, periodicity

Examples

```
## Ex1: sub-hourly data
## Creating a dummy 15-min zoo object, with 1 as the only value in each time period
dt <- seq( from=as.POSIXct("2021-06-30 00:15"), to=as.POSIXct("2021-06-30 23:45"), by="15 min" )
ndt <- length(dt)
shr <- zoo( rep(1, ndt), dt)
sfreq(shr)


## Ex2: hourly data
## Loading the time series of HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
hr <- KarameaAtGorgeQts
sfreq(hr)


## Ex3: Daily data
## Loading daily streamflows at the station Oca en Ona (Ebro River basin, Spain)
data(OcaEnOnaQts)
d <- OcaEnOnaQts
sfreq(d)


## Ex4: Monthly data
m <- daily2monthly(d, FUN=mean, na.rm=TRUE)
sfreq(m)


## Ex5: Annual data
a <- daily2annual(d, FUN=mean, na.rm=TRUE, out.fmt="%Y-%m-%d")
sfreq(a)
```

shiftyears *Shiftyears*

Description

It shifts years backwards for correct computation of annual values when the hydrological year does not start in january. Originally created only for internal usage in this pacakge.

Usage

```
shiftyears(ltime, lstart.month)
```

Arguments

ltime	vector with the date/times of each element of a zoo object.
lstart.month	numeric in [1,..,12], representing the starting month to be used in the computation of annual values. By default lstart.month=1 (Jan).

Value

Integer vector with the shifted years corresponding to each elemen of x.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

climograph, cmv, daily2annual

Examples

```
# Loading the DAILY precipitation data at SanMartino
data(SanMartinoPPts)
x <- SanMartinoPPts

# Subsetting 'x' to its first three months (Jan/1921 - Mar/1921)
x <- window(x, end="1921-12-31")

# Starting in January, it means no shift
shiftyears(ltime=time(x), lstart.month=1)

# Starting in April:
shiftyears(ltime=time(x), lstart.month=4)
```

si *Seasonality Index*

Description

Function to compute the seasonality index defined by Walsh and Lawler (1981) to classify the precipitation regime.

Usage

```
si(x, na.rm=TRUE, from=start(x), to=end(x), date.fmt="%Y-%m-%d", start.month=1)
```

Arguments

x	zoo object with daily or subdaily precipitation data.
na.rm	Logical. Should missing values be removed? -) TRUE : the monthly values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA within a month, the resulting average monthly value is NA .
from	OPTIONAL, used for extracting a subset of values. Character indicating the starting date for the values to be extracted. It must be provided in the format specified by date.fmt.
to	OPTIONAL, used for extracting a subset of values. Character indicating the ending date for the values to be extracted. It must be provided in the format specified by date.fmt.
date.fmt	Character indicating the format in which the dates are stored in *dates*, *from* and *to*. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
start.month	[OPTIONAL]. Only used when the (hydrological) year of interest is different from the calendar year. numeric in [1:12] indicating the starting month of the (hydrological) year. Numeric values in [1, 12] represents months in [January, December]. By default start.month=1.

Details

The seasonality index is computed as following:

si = (1/R) *sum(i=1, i=12, abs(xi - R/12))
where:
-) xi: mean monthly precipitation for month i
-) R: mean annual precipitation

This index can theoretically vary from 0 (when all months have the same rainfall) to 1.83 (when all the rainfall ocurrs in a single month). A qualitative classification of degrees of seasonality is the following:

si values	Rainfall regime
<= 0.19	Very equable
0.20 - 0.39	Equable but with a definite wetter season
0.40 - 0.59	Rather seasonal with a short drier season
0.60 - 0.79	Seasonal
0.80 - 0.99	Markedly seasonal with a long drier season
1.00 - 1.19	Most rain in 3 months or less
>= 1.20	Extreme, almost all rain in 1-2 months

Value

numeric with the seasonality index

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

References

Walsh, R. and Lawler, D. (1981). Rainfall seasonality: Description, spatial patterns and change through time (British Isles, Africa). Weather, 36(7), 201-208. doi:10.1002/j.1477-8696.1981.tb05400.x.

See Also

subdaily2daily

Examples

```
############################
## Ex 1: Seasonality index for a rain gauge with equable precipitation ,
##       but with a definite wetter season

## Loading daily precipitation data at the station San Martino di Castrozza,
## Trento Province, Italy, from 01/Jan/1921 to 31/Dec/1990.
data(SanMartinoPPts)
x <- SanMartinoPPts

## Amount of years in 'x' (needed for computations)
( nyears <- yip(from=start(x), to=end(x), out.type="nmbr" ) )


## Boxplot of monthly values, to look at the seasonal cycle

## Daily to Monthly
m <- daily2monthly(x, FUN=sum, na.rm=TRUE)
```

```
## Mean monthly values at the station
monthlyfunction(m, FUN=sum, na.rm=TRUE) / nyears

## Vector with the three-letter abbreviations of the month names
cmonth <- format(time(m), "%b")

## Creating ordered monthly factors
months <- factor(cmonth, levels=unique(cmonth), ordered=TRUE)

## Boxplot of the monthly values of precipitation
boxplot( coredata(m) ~ months, col="lightblue",
         main="Monthly precipitation, [mm]", ylab="P, [mm]")

# computing seasonality index
( si(x) )

############################
## Ex 2: Seasonality index for a rain gauge with markedly seasonal regime
##       with a long dry season

## Loading daily precipitation data at the station Cauquenes en El Arrayan,
## Maule Region, Chile, from 01/Jan/1979 to 31/Dec/2020.
data(Cauquenes7336001)
x <- Cauquenes7336001[, 1] # P is the first column

## Boxplot of monthly values, to look at the seasonal cycle

## Daily to Monthly
m <- daily2monthly(x, FUN=sum, na.rm=TRUE)

## Mean monthly values at the station
monthlyfunction(m, FUN=sum, na.rm=TRUE) / nyears

## Vector with the three-letter abbreviations of the month names
cmonth <- format(time(m), "%b")

## Creating ordered monthly factors
months <- factor(cmonth, levels=unique(cmonth), ordered=TRUE)

## Boxplot of the monthly values of precipitation
boxplot( coredata(m) ~ months, col="lightblue",
         main="Monthly precipitation, [mm]", ylab="P, [mm]")

# computing seasonality index
( si(x) )
```

`smry` *Summary*

Description

Extended summary function for numeric objects, with 13 summary statistics.

Usage

```
smry(x, ...)

## Default S3 method:
smry(x, na.rm=TRUE, digits = max(3, getOption("digits")-3), ...)

## S3 method for class 'zoo'
smry(x, na.rm=TRUE, digits = max(3, getOption("digits")-3), ...)

## S3 method for class 'Date'
smry(x, na.rm=TRUE, digits = max(3, getOption("digits")-3), ...)

## S3 method for class 'matrix'
smry(x, na.rm=TRUE, digits = max(3, getOption("digits")-3), ...)

## S3 method for class 'data.frame'
smry(x, na.rm=TRUE, digits = max(3, getOption("digits")-3), ...)
```

Arguments

x	a numeric object, vector, matrix or data.frame, for which a summary is desired.
na.rm	a logical value indicating whether 'NA' values should be stripped before the computation proceeds.
digits	numeric, with the amount of decimal places to be included in the result
...	further arguments passed to or from other methods.

Value

Computed summary statistics are:

Min	Minimum
1stQ	First quartile (lower-hinge)
Mean	Mean value
Median	Median
3rdQ	Third quartile (upper-hinge
Max	Maximum of the input values.
IQR	Interquartile Range. `IQR(x) = quantile(x,3/4) - quantile(x,1/4)`
sd	Standard deviation. It uses 'n-1' as denominator.
cv	Coefficient of variation (`cv= sd / |mean|`)
skewness	Skewness (using **e1071** package)
kurtosis	Kurtosis (using **e1071** package)
n	Total number of elements
NA's	Amount of missing values

Note

Skewness and Kurtosis are computed with the e1071 package

Author(s)

Mauricio Zambrano-Bigiarini <mzb.devel@gmail>

See Also

summary, fivenum, IQR, sd, skewness, kurtosis

Examples

```
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

## Summary of monthly precipitation values for the first 7 stations in 'EbroPPtsMonthly'
smry(EbroPPtsMonthly[,2:8])
```

---

sname2ts *Station Name -> Time Series*

Description

This function takes a data.frame whose columns contains the time series of several gauging stations, along with a character representing the name of one gauging station, and extracts the time series corresponding to that station.

Usage

```
sname2ts(x, sname, dates=1, date.fmt = "%Y-%m-%d", var.type,
         tstep.out = "daily", FUN, na.rm = TRUE, from, to)
```

Arguments

x	data.frame containing the complete times series of all the stations. It may also contain 1 column with the dates of the measurements, or they can be provided in a different way (see dates below).
sname	Character representing the name of a station, which have to correspond to one column name in x
dates	numeric, factor, Date object indicating how to obtain the dates corresponding to the sname station. -) If dates is a number (default), it indicates the index of the column in x that stores the dates -) If dates is a factor, it is converted into Date class, using the date format specified by date.fmt -) If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of element in x

date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
var.type	character representing the type of variable being plotted. Used for determining the function used for computing the monthly or/and annual values when FUN is missing. Valid values are: -) Precipitation => FUN=sum -) Temperature => FUN= mean -) Flow => FUN= mean
tstep.out	character that defines the time step of the desired output time series. Valid values are: -) daily : daily time series -) monthly: monthly time series -) annual : annual time series
FUN	ONLY required when var.type is missing and tstep is one of monthly or annual. Function that have to be applied for transforming from daily to monthly or annual time step (e.g., for precipitation FUN=sum and for temperature and flow ts, FUN=mean)
na.rm	a logical value indicating whether 'NA' values should be stripped before the computation proceeds.
from	OPTIONAL, used for extracting a subset of values. Character indicating the starting date for the values to be extracted. It must be provided in the format specified by date.fmt.
to	OPTIONAL, used for extracting a subset of values. Character indicating the ending date for the values to be extracted. It must be provided in the format specified by date.fmt.

Value

zoo object

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

sname2plot

Examples

```
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

## Annual values of temperature at the station "T9105", stored in 'EbroPPtsMonthly'.
sname2ts(EbroPPtsMonthly, sname="P9001", dates=1, FUN=sum, tstep.out="annual")
```

stdx *Standarization*

Description

Standarizes a vector or matrix, i.e., scales all the values in a way that the transformed values will be within the range [0, 1].

Usage

```
stdx(x, ...)
```

Arguments

x	vector, matrix or data.frame to be scaled
...	further arguments passed to or from other methods

Details

$$z = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

scale, istdx

Examples

```
############
## Loading the monthly time series of precipitation within the Ebro River basin.
data(EbroPPtsMonthly)

# Standarizing only some values of 'EbroPPtsMonthly'
stdx(as.matrix(EbroPPtsMonthly[1:70,10:13]))
```

subdaily2daily *Sub-daily -> Daily*

Description

Generic function for transforming a Sub-DAILY time series into a DAILY one

Usage

```
subdaily2daily(x, ...)

## Default S3 method:
subdaily2daily(x, FUN, na.rm = TRUE, na.rm.max=0,
                       start="00:00:00", start.fmt= "%H:%M:%S", tz, ...)

## S3 method for class 'zoo'
subdaily2daily(x, FUN, na.rm = TRUE, na.rm.max=0,
                       start="00:00:00", start.fmt= "%H:%M:%S", tz, ...)

## S3 method for class 'data.frame'
subdaily2daily(x, FUN, na.rm = TRUE, na.rm.max=0,
           start="00:00:00", start.fmt= "%H:%M:%S", tz,
           dates=1, date.fmt="%Y-%m-%d %H:%M:%S", out.fmt="zoo",
           verbose= TRUE, ...)

## S3 method for class 'matrix'
subdaily2daily(x, FUN, na.rm = TRUE, na.rm.max=0,
           start="00:00:00", start.fmt= "%H:%M:%S", tz,
           dates=1, date.fmt="%Y-%m-%d %H:%M:%S", out.fmt="zoo",
           verbose= TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with sub-daily time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represents the time series measured in each gauging station, and the column names of x should correspond to the ID of each station (starting by a letter).
FUN	Function that have to be applied for aggregating from sub-daily into daily time step (e.g., for precipitation FUN=sum and for temperature and streamflows ts FUN=mean). FUN MUST accept the na.rm argument, because na.rm is passed to FUN.
na.rm	Logical. Should missing values be removed? -) TRUE : the daily values are computed only for days with a percentage of missing values less than na.rm.max

-) FALSE: if there is AT LEAST one NA within a day, the corresponing daily value in the output object will be NA.

na.rm.max — Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each day to keep the daily aggregated value in the output object of this function. In other words, if the percentage of missing values in a given day is larger than na.rm.max the corresponding daily value will be NA.

start — character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by start.fmt. This value is used to define the time when a new day begins (e.g., for some rain gauge stations).
-) All the values of x with a time attribute before start are considered as belonging to the day before the one indicated in the time attribute of those values.
-) All the values of x with a time attribute equal to start are considered to be equal to "00:00:00" in the output zoo object.
-) All the values of x with a time attribute after start are considered as belonging to the same day as the one indicated in the time attribute of those values.

It is useful when the daily values start at a time different from "00:00:00". Use with caution. See examples.

start.fmt — character indicating the format in which the time is provided in start, By default date.fmt=%H:%M:%S. See format in as.POSIXct.

tz — character, with the specification of the time zone used in both x and start. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct.
If tz is missing (the default), it is automatically set to the time zone used in time(x).
This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.

dates — numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of x (e.g., gauging station)
If dates is a number, it indicates the index of the column in x that stores the date and times
If dates is a factor, it is converted into POSIXct class, using the date format specified by date.fmt
If dates is already of POSIXct or POSIXt class, the code verifies that the number of elements on it be equal to the number of elements in x

date.fmt — character indicating the format in which the dates are stored in dates, By default date.fmt=%Y-%m-%d %H:%M:%S. See format in as.Date.
ONLY required when class(dates)=="factor" or class(dates)=="numeric".

out.fmt — OPTIONAL. Only used when x is a matrix or data.frame object /cr character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo (default)

verbose — logical; if TRUE, progress messages are printed

... — arguments additional to na.rm passed to FUN.

Details

This function assumes that x has a strictly regular time frequency (see is.regular) and raise a warning otherwise.

Value

a zoo object with daily time series

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

subhourly2hourly, subdaily2monthly, subdaily2annual, subdaily2seasonal, as.POSIXct, dm2seasonal, monthlyfunction, seasonalfunction, hydroplot, vector2zoo, izoo2rzoo

Examples

```
## Ex1: Computation of daily values, removing any missing value in 'x'

## Loading the time series of hourly streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

# Plotting the hourly streamflow values
plot(x)

# Subsetting 'x' to its first three days (01/Jan/1980 - 03/Jan/1980)
x <- window(x, end="1980-01-03 23:59:00")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Agreggating from Sub-Daily to Daily, removing any missing value in 'x'
( d1 <- subdaily2daily(x, FUN=mean, na.rm=TRUE) )


## Ex2: Computation of daily values, removing any missing value in 'x' and
##      considering that the new day starts at 08:00:00 local time
( d2 <- subdaily2daily(x, FUN=mean, na.rm=TRUE, start="08:00:00") )

## Ex3: Computation of daily values, removing any missing value in 'x' and
##      considering that the new day starts at 08:00:00, and forcing
#       UTC both for 'x' and 'start'
( d3 <- subdaily2daily(x, FUN=mean, na.rm=TRUE, start="08:00:00", tz="UTC") )

######################
```

```
## Ex4: Compuation of daily values only when the percentage of NAs in each
#       day is lower than a user-defined percentage (10% in this example).
( d4 <- subdaily2daily(x, FUN=mean, na.rm=TRUE, na.rm.max=0.1) )
```

---

subdaily2weekly *Subdaily -> Weekly*

Description

Generic function for transforming a DAILY (or sub-daily) regular time series into a WEEKLY one

Usage

```
subdaily2weekly(x, ...)

## Default S3 method:
subdaily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0,
           start="00:00:00", start.fmt= "%H:%M:%S", tz, ...)

## S3 method for class 'zoo'
subdaily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0,
           start="00:00:00", start.fmt= "%H:%M:%S", tz, ...)

## S3 method for class 'data.frame'
subdaily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0,
           start="00:00:00", start.fmt= "%H:%M:%S", tz,
           dates=1, date.fmt = "%Y-%m-%d %H:%M:%S",
           out.fmt="zoo", verbose=TRUE, ...)

## S3 method for class 'matrix'
subdaily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0,
           start="00:00:00", start.fmt= "%H:%M:%S", tz,
           dates=1, date.fmt = "%Y-%m-%d %H:%M:%S",
           out.fmt="zoo", verbose=TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with (sub)daily time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represents the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that have to be applied for transforming from daily to weekly time step (e.g., for precipitation FUN=sum and for temperature and streamflow ts FUN=mean). FUN MUST accept the na.rm argument, because na.rm is passed to FUN.

na.rm	Logical. Should missing values be removed? -) TRUE : the weekly values are computed only for weeks with a percentage of missing values less than na.rm.max -) FALSE: if there is AT LEAST one NA within a month, the corresponing weekly values in the output object will be NA.
na.rm.max	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each week to keep the weekly aggregated value in the output object of this function. In other words, if the percentage of missing values in a given week is larger than na.rm.max the corresponding weekly value will be NA.
start	character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by start.fmt. This value is used to define the time when a new day begins (e.g., for some rain gauge stations). -) All the values of x with a time attribute before start are considered as belonging to the day before the one indicated in the time attribute of those values. -) All the values of x with a time attribute equal to start are considered to be equal to "00:00:00" in the output zoo object. -) All the values of x with a time attribute after start are considered as belonging to the same day as the one indicated in the time attribute of those values. It is useful when the daily values start at a time different from "00:00:00". Use with caution. See examples.
start.fmt	character indicating the format in which the time is provided in start, By default date.fmt=%H:%M:%S. See format in as.POSIXct.
tz	character, with the specification of the time zone used in both x and start. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(x). This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.
dates	numeric, factor or Date object indicating how to obtain the dates for each gauging station If dates is a number (default), it indicates the index of the column in *x* that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days on it be equal to the number of elements in x
date.fmt	character indicating the format in which the DateTime objects are stored in *dates*, e.g. %Y-%m-%d %H:%M:%S. See format in as.POSIXct. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.fmt	OPTIONAL. Only used when x is a matrix or data.frame object /cr character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo.
verbose	logical; if TRUE, progress messages are printed
...	arguments additional to na.rm passed to FUN.

Value

a zoo object with weekly time frequency

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

cmv, subhourly2hourly, daily2monthly, daily2annual, subdaily2daily, weeklyfunction, hydroplot, vector2zoo, izoo2rzoo, as.Date

Examples

```
######################
## Ex1: Computation of WEEKLY values from HOURLY ts, only when the percentage of NAs in
#       each week is lower than a user-defined percentage (10% in this example).

## Loading the HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

# Subsetting 'x' to its first three weeks
# (31-December-1979 to 31-March-1980)
x <- window(x, end="1980-03-31 23:59:00")

## Transforming into NA the 10% of values in 'x'
set.seed(10) # for reproducible results
n           <- length(x)
n.nas       <- round(0.1*n, 0)
na.index    <- sample(1:n, n.nas)
x[na.index] <- NA

## Daily to Weekly, only for weeks with less than 10% of missing values
( w2 <- subdaily2weekly(x, FUN=sum, na.rm=TRUE, na.rm.max=0.1) )

# Verifying that the second and third month of 'x' had 10% or more of missing values
cmv(x, tscale="weekly")

## Not run:
######################
## Ex2: Computation of WEEKLY values from HOURLY ts, removing any missing value in 'x'
#       Loading the HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

# Sub-daily to weekly ts
subdaily2weekly(x, FUN=mean, na.rm=TRUE)

## End(Not run)
```

subhourly2hourly *Sub-hourly -> Hourly*

Description

Generic function for transforming a sub-HOURLY time series into an HOURLY one

Usage

```
subhourly2hourly(x, ...)

## Default S3 method:
subhourly2hourly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'zoo'
subhourly2hourly(x, FUN, na.rm=TRUE, na.rm.max=0, tz, ...)

## S3 method for class 'data.frame'
subhourly2hourly(x, FUN, na.rm=TRUE, na.rm.max=0,
           dates=1, date.fmt="%Y-%m-%d %H:%M:%S", out.fmt="zoo",
           verbose= TRUE, ...)

## S3 method for class 'matrix'
subhourly2hourly(x, FUN, na.rm=TRUE, na.rm.max=0,
           dates=1, date.fmt="%Y-%m-%d %H:%M:%S", out.fmt="zoo",
           verbose= TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with sub-hourly time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x represent the ID of each station.
FUN	Function that have to be applied for transforming from sub-hourly to hourly time step. (e.g., for precipitation FUN=sum and for temperature and streamflow ts, FUN=mean).
na.rm	Logical. Should missing values be removed? -) TRUE : the hourly values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA sub-hourly value within a day, the corresponding hourly value(s) will be NA as well
na.rm.max	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each hour to keep the hourly aggregated value in the output object of this function. In other words, if the percentage of missing values in a given hour is larger than na.rm.max the corresponding hourly value will be NA.

tz	character, with the specification of the time zone used for x. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(x). If tz is provided, it forces time(x) to be in the tome zone specified by tz, without modifying the the values (hours, minutes, seconds, etc). A list of valid time zones can be obtained by calling the base function OlsonNames(). This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for x. It should be used with caution, being well aware of the time zone of the data. See examples.
dates	numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of x (e.g., gauging station) If dates is a number, it indicates the index of the column in x that stores the date and times If dates is a factor, it is converted into POSIXct class, using the date format specified by date.fmt If dates is already of POSIXct or POSIXt class, the code verifies that the number of elements on it be equal to the number of elements in x
date.fmt	character indicating the format in which the dates are stored in dates, By default date.fmt=%Y-%m-%d %H:%M:%S. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.fmt	OPTIONAL. Only used when x is a matrix or data.frame object /cr character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo (default)
verbose	logical; if TRUE, progress messages are printed
...	further arguments passed to or from other methods.

Value

a zoo object with hourly time series

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

subhourly2nminutes,, subdaily2daily, subdaily2monthly, subdaily2annual, subdaily2seasonal, as.POSIXct, dm2seasonal, monthlyfunction, seasonalfunction, hydroplot, vector2zoo, izoo2rzoo

Examples

```
## Creating a 15-min time sequence and counting its length
dt  <- seq( from=as.POSIXct("2021-06-30 00:15"), to=as.POSIXct("2021-06-30 23:45"), by="15 min" )
ndt <- length(dt)

## Creating a dummy 15-min zoo object, with 1 as the only value in each time period
x <- zoo( rep(1, ndt), dt)


## sub-hourly to hourly
h1 <- subhourly2hourly(x, FUN=sum, na.rm=TRUE)

## Aggregation of 3 sub-hourly ts (i.e., a zoo matrix) into an hourly one
X  <- cbind(x, x, x)
h2 <- subhourly2hourly(X, FUN=sum, na.rm=TRUE)
```

---

subhourly2nminutes *Sub-hourly -> n-minutes*

Description

Generic function for aggregating a sub-hourly time series into a "n-minutes' one.

Usage

```
subhourly2nminutes(x, ...)

## Default S3 method:
subhourly2nminutes(x, nminutes, FUN, na.rm=TRUE,
                                     from=start(x), to=end(x), ...)

## S3 method for class 'zoo'
subhourly2nminutes(x, nminutes, FUN, na.rm=TRUE,
                                  from=start(x), to=end(x), tz, ...)

## S3 method for class 'data.frame'
subhourly2nminutes(x, nminutes, FUN, na.rm=TRUE,
        from=start(x), to=end(x), dates=1, date.fmt="%Y-%m-%d %H:%M:%S",
        out.fmt="zoo", verbose= TRUE, ...)

## S3 method for class 'matrix'
subhourly2nminutes(x, nminutes, FUN, na.rm=TRUE,
        from=start(x), to=end(x), dates=1, date.fmt="%Y-%m-%d %H:%M:%S",
        out.fmt="zoo", verbose= TRUE, ...)
```

Arguments

x	zoo, data.frame or matrix object, with sub-hourly time series. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x represent the ID of each station.
nminutes	numeric, defining the amount of minutes to be used for aggregating x. nminutes must be larger than the amount of minutes between each x value (computed as time(x)[2]-time(x)[1]).
FUN	Function that have to be applied for aggregating from sub-hourly into n-minutes time step. (e.g., for precipitation FUN=sum and for temperature and streamflow ts, FUN=mean).
na.rm	Logical. Should missing values be removed? -) TRUE : the hourly values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA sub-hourly value within a day, the corresponding hourly value(s) will be NA as well
from	POSIX object indicating the starting time used to carry out the temporal aggregation. When from > start(x) then x is cut in time to the starting DateTime defined by from. When from < start(x) then x is extended backward with NAs to the starting DateTime defined by from.
to	POSIX object indicating the ending time used to carry out the temporal aggregation. When to < end(x) then x is cut in time to the ending DateTime defined by to. When to > end(x) then x is extended forward with NAs to the ending DateTime defined by from.
tz	character, with the specification of the time zone used for x, from, and to. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If tz is missing (the default), it is automatically set to the time zone used in time(x). If tz is provided, it forces time(x) to be in the tome zone specified by tz, without modifying the the values (hours, minutes, seconds, etc). A list of valid time zones can be obtained by calling the base function OlsonNames(). This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for from and to. It should be used with caution, being well aware of the time zone of the data. See examples.
dates	numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of x (e.g., gauging station)

	If dates is a number, it indicates the index of the column in x that stores the date and times If dates is a factor, it is converted into POSIXct class, using the date format specified by date.fmt If dates is already of POSIXct or POSIXt class, the code verifies that the number of elements on it be equal to the number of elements in x
date.fmt	character indicating the format in which the dates are stored in dates, By default date.fmt=%Y-%m-%d %H:%M:%S. See format in as.Date. ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.fmt	OPTIONAL. Only used when x is a matrix or data.frame object /cr character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo (default)
verbose	logical; if TRUE, progress messages are printed
...	further arguments passed to or from other methods.

Value

a zoo object with hourly time series

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

subhourly2hourly, subdaily2daily, subdaily2monthly, subdaily2annual, subdaily2seasonal, as.POSIXct, dm2seasonal, monthlyfunction, seasonalfunction, hydroplot, vector2zoo, izoo2rzoo

Examples

```
## Creating a 5-min time sequence and counting its length
dt <- seq( from=as.POSIXct("2021-06-30 00:00"), to=as.POSIXct("2021-06-30 23:55"), by="5 min" )
ndt <- length(dt)

## Creating a dummy 5-min zoo object, with 1 as the only value in each time period
x <- zoo( rep(1, ndt), dt)


## Aggregation from 5-minute single ts into 10-minute ts
h1 <- subhourly2nminutes(x, nminutes= 10, FUN=sum, na.rm=TRUE)

## Aggregation of 3 ts with 5-minute time frequency (i.e., a zoo matrix)
## into a 30-minute zoo object.
X  <- cbind(x, x, x)
h2 <- subhourly2nminutes(X, nminutes= 30, FUN=sum, na.rm=TRUE)
```

`time2season` *Date/DateTime character -> Seasonal character*

Description

This function transforms a character vector of Dates or DateTimes into a character vector of seasons (summer, winter, autumn, spring), depending on the value of `type`:

When `type=default` -) winter = DJF: December, January, February
-) spring = MAM: March, April, May
-) summer = JJA: June, July, August
-) autumn = SON: September, October, November

When `type=FrenchPolynesia` -) winter = DJFM: December, January, February, March
-) spring = AM : April, May
-) summer = JJAS: June, July, August, September
-) autumn = ON : October, November

Usage

```
time2season(x, out.fmt = "months", type="default")
```

Arguments

`x`	vector with the dates that have to be transformed. class(x) must be Date
`out.fmt`	character, indicating the format of the output seasons. Possible values are: -) `seasons` => c("winter", "spring", "summer", autumn") -) `months` => c("DJF", "MAM", "JJA", SON") or c("DJFM", "AM", "JJAS", ON")
`type`	character, indicating which weather seasons will be used for computing the output. Possible values are: -) `default` => "winter"= Dec, Jan, Feb; "spring"= Mar, Apr, May; "summer"=Jun, Jul, Aug; "autumn"= Sep, Oct, Nov -) `FrenchPolynesia` => "winter"= Dec, Jan, Feb, Mar; "spring"= Apr, May; "summer"=Jun, Jul, Aug, Sep; "autumn"= Oct, Nov

Value

character vector with the weather season to which each date in `x` belongs

Note

Weather seasons corresponding to French Polynesia were defined following a comment from Lydie Sichoix

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

dm2seasonal, seasonalfunction, extract, dip, mip

Examples

```
## Sequence of daily dates between "1961-01-01" and "1961-12-31"
t <- dip("1961-01-01", "1961-12-31")
time2season(t)

## Sequence of monthly dates between "1961-01-01" and "1961-12-31"
t <- mip("1961-01-01", "1961-12-31")
time2season(t)
time2season(t, out.fmt="seasons")
```

---

vector2zoo *Vector -> Zoo*

Description

Transform a numeric vector and its corresponding dates into a zoo object.

Usage

```
vector2zoo(x, dates, date.fmt = "%Y-%m-%d")
```

Arguments

x	numeric vector
dates	character, factor, Date or POSIXct object with the dates corresponding to each element of x. Valid object class for dates are: character, factor, Date, POSIXct
date.fmt	character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d. See 'Details' section in strptime. ONLY required when class(dates)=="factor" or class(dates)=="character".

Value

a zoo object, with values given by x and time stamps given by dates

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

zoo, izoo2rzoo, dip, mip, yip

Examples

```
##
## Example1: creating daily data

# Generating a numeric variable (e.g., read from the outputs of an hydrological model)
x <- 1:31

# Generating the dates corresponding to the previous values
dates <- dip("1961-01-01", "1961-01-31")

## Transforming from 'numeric' to 'zoo' class
z <- vector2zoo(x, dates)

##
## Example2: creating hourly data

# Generating a numeric variable
x <- rnorm(7)

# Generating hourly times since 17:00:00 up to 23:00:00 for 2012-Oct-15
dates <- ISOdatetime(2012, 10, 15, 17:23, 00, 0)

## Transforming from 'numeric' to 'zoo' class
z <- vector2zoo(x, dates)
```

---

weeklyfunction *Weekly Function*

Description

Generic function for obtaining 52 weekly values of a zoo object, by applying any R function to ALL the values in the object belonging to each one of the 52 calendar weeks (starting on Monday).

Usage

```
weeklyfunction(x, ...)

## Default S3 method:
weeklyfunction(x, FUN, na.rm=TRUE, na.rm.max=0, start="00:00:00",
                                  start.fmt= "%H:%M:%S", tz, ...)

## S3 method for class 'zoo'
weeklyfunction(x, FUN, na.rm=TRUE, na.rm.max=0, start="00:00:00",
                              start.fmt= "%H:%M:%S", tz, ...)
```

```
## S3 method for class 'data.frame'
weeklyfunction(x, FUN, na.rm=TRUE, na.rm.max=0, start="00:00:00",
                       start.fmt= "%H:%M:%S", tz, dates=1, date.fmt="%Y-%m-%d",
                        out.type="data.frame", verbose=TRUE,...)

## S3 method for class 'matrix'
weeklyfunction(x, FUN, na.rm=TRUE, na.rm.max=0, start="00:00:00",
                       start.fmt= "%H:%M:%S", tz, dates=1, date.fmt="%Y-%m-%d",
                        out.type="data.frame", verbose=TRUE,...)
```

Arguments

x	zoo, xts, data.frame or matrix object, with daily or monthly time series. Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter).
FUN	Function that will be applied to ALL the values in x belonging to each one of the 12 months of the year (e.g., FUN can be some of mean, sum, max, min, sd).
na.rm	Logical. Should missing values be removed? -) TRUE : the monthly values and FUN are computed considering only those values in x different from NA -) FALSE: if there is AT LEAST one NA within a month, the corresponding monthly value will be NA
na.rm.max	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each month to keep the weekly aggregated value in the output object of this function. In other words, if the percentage of missing values in a given month is larger or equal than na.rm.max the corresponding weekly value will be NA.
start	character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by start.fmt. This value is used to define the time when a new day begins (e.g., for some rain gauge stations). -) All the values of x with a time attribute before start are considered as belonging to the day before the one indicated in the time attribute of those values. -) All the values of x with a time attribute equal to start are considered to be equal to "00:00:00" in the output zoo object. -) All the values of x with a time attribute after start are considered as belonging to the same day as the one indicated in the time attribute of those values. It is useful when the daily values start at a time different from "00:00:00". Use with caution. See examples.
start.fmt	character indicating the format in which the time is provided in start, By default date.fmt=%H:%M:%S. See format in as.POSIXct.
tz	character, with the specification of the time zone used in both x and start. System-specific (see time zones), but "" is the current time zone, and "GMT" is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct.

If tz is missing (the default), it is automatically set to the time zone used in time(x).
This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.

dates — It is only used when x is not a zoo object.
numeric, factor, Date indicating how to obtain the dates.
If dates is a number (default), it indicates the index of the column in x that stores the dates
If dates is a factor, it is converted into 'Date' class, using the date format specified by date.fmt
If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of elements in x

date.fmt — It is only used when x is not a zoo object.
character indicating the format in which the dates are stored in *dates*, e.g. %Y-%m-%d.
See format in as.Date.
ONLY required when class(dates)=="factor" or class(dates)=="numeric".

out.type — It is only used when x is a matrix or data.frame.
Character defining the desired type of output. Valid values are:
-) data.frame: a data.frame, with 12 columns representing the months, and as many rows as gauging stations are included in x
-) db : a data.frame, with 4 columns will be produced. Useful for a posterior boxplot
The first column ('StationID') will store the ID of the station,
The second column ('Year') will store the year,
The third column ('Month') will store month,
The fourth column ('Value') will contain the monthly value corresponding to the three previous columns.

verbose — Logical; if TRUE, progress messages are printed

... — further arguments passed to or from other methods

Value

When x is a zoo object, a numeric vector with 12 elements representing the computed monthly value for each month.
When x is a data.frame which columns represent measurements at different gauging stations, the resulting object is a data.frame with 12 columns and as many rows as gauging stations are in x, each row storing the computed 12 monthly value for each gauging station.

Note

Due to the fact that FUN is applied over all the elements in x belonging to a given calendar month, its result will depend on the sampling frequency of x and the type of function provided by FUN (**special attention have to be put when** FUN=sum)

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

annualfunction, seasonalfunction, dm2seasonal, daily2monthly, daily2annual

Examples

```
## Ex1: Computation of mean WEEKLY values from DAILY ts, removing any missing value in 'x'

# Loading DAILY streamflows (3 years) at the station
# Oca en Ona (Ebro River basin, Spain)
data(OcaEnOnaQts)
x <- OcaEnOnaQts

## Mean WEEKLY streamflows at station 'x'
weeklyfunction(x, FUN=mean, na.rm=TRUE)

######################
## Ex2: Computation of mean WEEKLY values from HOURLY ts, removing any missing value in 'x'

# Loading HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

## Mean WEEKLY streamflows at station 'x'. Each day starts at 00:00:00
weeklyfunction(x, FUN=mean, na.rm=TRUE)

######################
## Ex3: Computation of mean WEEKLY values from HOURLY ts, removing any missing value in 'x'
##      and starting each day at 08:00:00

# Loading HOURLY streamflows for the station Karamea at Gorge
data(KarameaAtGorgeQts)
x <- KarameaAtGorgeQts

## Mean WEEKLY streamflows at station 'x'. Each day starts at 00:00:00
weeklyfunction(x, FUN=mean, na.rm=TRUE, start="00:00:00")
```

---

yip *Years in Period*

Description

Given any starting and ending dates, it generates:
1) a vector of class Date with all the years between the two dates (both of them included), OR
2) the amount of years between the two dates

Usage

```
yip(from, to, date.fmt = "%Y-%m-%d", out.type = "seq")
```

Arguments

from	Character indicating the starting date for creating the sequence. It has to be in the format indicated by date.fmt.
to	Character indicating the ending date for creating the sequence. It has to be in the format indicated by date.fmt.
date.fmt	character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See format in as.Date.
out.type	Character indicating the type of result that is given by this function. Valid values are: -) seq => a vectorial sequence with all the years within the given dates. -) nmbr => the number of years within the given dates.

Value

Depending on the value of out.type, it returns:
1) seq : a vector of class Date with all the years between from and to (both of them included), OR
2) nmbr: a single numeric value with the amount of years between the two dates.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

See Also

dip, diy, mip

Examples

```
# Sequence of monthly dates between "1961-01-01" and "1961-12-31"
yip("1961-01-01", "1961-12-31")

## Computing the number of years between 1961 and 1975,
## by using "%d-%m-%Y" as date format   ##
yip("01-01-1961", "01-01-1975", date.fmt= "%d-%m-%Y", out.type = "nmbr")
```

---

zoo2RHtest *Zoo -> RHTest*

Description

It creates the input file to the '*RHtest_dlyPrcp.r*' script, used for testing the homogeneity of climatological time series (available at: https://etccdi.pacificclimate.org/software.shtml)

Usage

```
zoo2RHtest(x, fname="pcp.txt", tstep.out="daily", dec=".", na="-999.0")
```

Arguments

x	time series that will be written. class(x) must be a zoo object
fname	Character, with the filename of the precipitation time series
tstep.out	Character indicating the time step that have to be used for writing x into the output file
dec	the string to use for decimal points in numeric or complex columns: must be a single character.
na	character to be used for representing the missing values in the data

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>

References

https://etccdi.pacificclimate.org/software.shtml https://etccdi.pacificclimate.org/RHtest/RHtestsV4_UserManual_10Dec2014.pdf

Examples

```
## Loading the SanMartino precipitation data
data(SanMartinoPPts)
x <- SanMartinoPPts

#Getting the monthly ts
pcp.m <- daily2monthly(x, FUN=sum, na.rm=FALSE)

## Not run:
# From zoo to the input format required by 'FindU.dlyPrcp' function
zoo2RHtest(x=pcp.m, fname="pcp-monthly.txt", tstep.out="monthly", na="-999.0")

# Homogeneity analysis
FindU.dlyPrcp(InSeries="pcp-monthly.txt", output="pcp-monthly", MissingValueCode="-999.0",
GUI=FALSE, pthr=0, Mq=10, p.lev=0.95, Iadj=10000)

## End(Not run)
```

Index