

Package: hydroTSM (via r-universe)

October 23, 2026

Type Package

Title Time Series Management and Analysis for Hydrological Modelling

Version 0.8-26

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 <https://hzambran.r-universe.dev>

Date/Publication 2026-10-23 01:57:46 UTC

RemoteUrl <https://github.com/hzambran/hydrotsm>

RemoteRef HEAD

RemoteSha 6f51652c7b7fc5f96df614582e35a313b38a9df9

Contents

hydroTSM-package	3
(sub)daily2annual	6
(sub)daily2monthly	10
annualfunction	14
baseflow	17
calendarHeatmap	20
Cauquenes7336001	23
change_tz	24
climograph	25
cmv	30
daily2weekly	34
dip	38
diy	39
dm2seasonal	40
drawTimeAxis	43
dwdays	44
dwi	45
EbroPPtsMonthly	47
extract	48
fdc	50
fdcu	53
hip	57
hydropairs	59
hydroplot	60
infillxy	65
isComplete	66
istdx	70
izoo2rzoo	72
KarameaAtGorgeQts	76
ma	77
MaquehueTemuco	78
matrixplot	79
mip	81
monthlyfunction	82
nperiods	85
OcaEnOnaQts	87
plot_pq	87
precipQC	92
precipQC-class	93
precipQC_daily	94
precipQC_subdaily	100

precipQC_tests	105
rm1stchar	110
SanMartinoPPts	111
seasonalfunction	112
sfreq	114
shiftyears	116
si	117
smry	119
sname2ts	121
spi	123
stdx	126
streamflowQC	127
streamflowQC-class	129
streamflowQC_daily	130
streamflowQC_subdaily	134
streamflowQC_tests	138
subdaily2daily	142
subdaily2weekly	145
subhourly2hourly	148
subhourly2nhourly	151
subhourly2nminutes	153
tempQC	156
tempQC-class	157
tempQC_daily	158
tempQC_subdaily	163
tempQC_tests	166
time2season	170
vector2zoo	172
weeklyfunction	173
yip	176
zoo2RHtest	177
Index	179

hydroTSM-package	<i>Management, analysis, and plot of hydrological time series, with focus on hydrological modelling</i>
------------------	---

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.
[OcaEn0naQts](#) 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.
[submonthly2monthly](#) Aggregation from hourly, daily, weekly or pseudo-weekly 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.
[subhourly2nhourly](#) Aggregation from subhourly or hourly to n-hourly.
[subhourly2nminutes](#) Aggregation 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 object.
`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.
`spei` Standardized Precipitation-Evapotranspiration Index.
`spi` Standardized Precipitation Index for monthly precipitation.
`sname2ts` Station Name -> Time Series.
`zoo2RHtest` Zoo object -> RHTest.

Miscellaneous 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 stream-flows 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 <code>na.rm.max</code> the corresponding annual value will be NA.
<code>out.fmt</code>	Character indicating the date format for the output zoo object. See format in as.Date . Possible values are: -) <code>%Y</code> : only the year will be used for the time. Default option. (e.g., "1961" "1962" ...) -) <code>%Y-%m-%d</code> : a complete date format will be used for the time. (e.g., "1961-01-01" "1962-01-01" ...). See Details.
<code>start.month</code>	numeric in [1,...,12], representing the starting month (1:Jan, ..., 12:Dec) to be used in the computation of annual values. By default <code>start.month=1</code> .
<code>dates</code>	numeric, factor or Date object indicating how to obtain the dates for corresponding to each gauging station If <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates If <code>dates</code> is a factor, it is converted into Date class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of Date class, the code verifies that the number of days on it be equal to the number of element in <code>x</code>
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>dates</code> , e.g. <code>%Y-%m-%d</code> . See format in as.Date . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
<code>out.type</code>	Character that defines the desired type of output. Valid values are: -) <code>data.frame</code> : a data.frame, with as many columns as stations are included in <code>x</code> , and row names indicating the Year -) <code>db</code> : 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.
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	arguments additional to <code>na.rm</code> 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., if the year is 2022, the time attribute will be 2022-01-01). The only exception occurs 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 *Daily/Submonthly -> Monthly*

Description

Generic functions for transforming daily, sub-daily, weekly, or pseudo-weekly time series into a MONTHLY one.

Usage

```

daily2monthly(x, ...)
subdaily2monthly(x, ...)
submonthly2monthly(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, ...)

## Default S3 method:
submonthly2monthly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'zoo'
submonthly2monthly(x, FUN, na.rm=TRUE, na.rm.max=0, ...)

## S3 method for class 'data.frame'
submonthly2monthly(x, FUN, na.rm=TRUE, na.rm.max=0,
                  dates=1, date.fmt = "%Y-%m-%d", tz=NULL, out.type = "data.frame",
                  out.fmt="numeric", verbose=TRUE, ...)

## S3 method for class 'matrix'
submonthly2monthly(x, FUN, na.rm=TRUE, na.rm.max=0,
                  dates=1, date.fmt = "%Y-%m-%d", tz=NULL, out.type = "data.frame",
                  out.fmt="numeric", verbose=TRUE, ...)

```

Arguments

x	zoo, data.frame or matrix object, with submonthly 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.

<code>na.rm</code>	Logical. Should missing values be removed? -) TRUE : the monthly values are computed only for months with a percentage of missing values less than <code>na.rm.max</code> -) FALSE: if there is AT LEAST one NA within a month, the corresponding monthly values in the output object will be NA.
<code>na.rm.max</code>	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 <code>na.rm.max</code> the corresponding monthly value will be NA.
<code>dates</code>	numeric, factor or Date object indicating how to obtain the dates for each gauging station. <code>submonthly2monthly</code> also accepts character and POSIXt objects. If <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates If <code>dates</code> is a factor or character, it is converted into Date or POSIXct class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of Date or POSIXt class, the code verifies that the number of time steps on it be equal to the number of elements in <code>x</code>
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>dates</code>, e.g. <code>%Y-%m-%d</code>. See format in as.Date. ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code>.
<code>tz</code>	character, with the specification of the time zone used for parsing POSIXt dates when <code>x</code> is a matrix or data.frame.
<code>out.type</code>	Character that defines the desired type of output. Valid values are: -) <code>data.frame</code>: a data.frame, with as many columns as stations are included in <code>x</code>, and row names indicating the month and year for each value. -) <code>db</code> : 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.
<code>out.fmt</code>	OPTIONAL. Only used when <code>x</code> 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: <code>numeric</code>, <code>zoo</code>.
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	arguments additional to <code>na.rm</code> passed to FUN.

Details

`submonthly2monthly` follows the output conventions of `daily2monthly`, but it groups values directly by calendar month without requiring a regular sampling frequency. This allows hourly, daily, weekly, and pseudo-weekly zoo objects whose Date/POSIXt indexes are not regular enough for automatic frequency detection. For `na.rm.max`, the percentage of missing values is computed using the values observed within each month; missing unobserved regular time steps are not inserted.

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)

#####
## Ex4: Pseudo-weekly dates aggregated directly to monthly values
dates <- as.Date(c("2023-01-01", "2023-01-08", "2023-01-15",
                  "2023-01-23", "2023-02-01", "2023-02-10"))
x <- zoo(c(1, 2, 3, 4, 5, 6), dates)
submonthly2monthly(x, FUN=sum, 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 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 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 <i>dates</i> , 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", bfc col=c("blue", "darkcyan", "darkorange3"),
         pch=15, cex=0.3, ...)
```

Arguments

<code>x</code>	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.
<code>beta</code>	numeric representing the filter parameter. Default value is 0.925 as recommended by Arnold and Allen (1999)
<code>from</code>	Character indicating the starting date for subsetting <code>x</code> . It has to be in the format indicated by <code>date.fmt</code> . The default value corresponds to the date of the first element of <code>x</code> .
<code>to</code>	Character indicating the ending date for subsetting <code>x</code> . It has to be in the format indicated by <code>date.fmt</code> . The default value corresponds to the date of the last element of <code>x</code> .
<code>n.pass</code>	Integer indicating the number times the filter will be passed over the streamflow data stored in <code>x</code> . Default is 3.
<code>pass.start</code>	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
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>from</code> and <code>to</code> , e.g. <code>%Y-%m-%d</code> . See ‘Details’ section in strptime . By default, <code>date.fmt</code> is missing, and it is automatically set to <code>%Y-%m-%d</code> when <code>time(x)</code> is Date object, and set to <code>%Y-%m-%d %H:%M:%S</code> when <code>x</code> is a sub-daily zoo object.
<code>tz</code>	character, with the specification of the time zone used for <code>from</code> , <code>to</code> . System-specific (see time zones), but <code>""</code> is the current time zone, and <code>"GMT"</code> 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.

<code>na.fill</code>	Character indicating how to fill any NA present in <code>x</code> . 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
<code>out.type</code>	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.
<code>plot</code>	logical. Indicates if the baseflow should be plotted or not. If plotted, the original <code>x</code> values are plotted as well.
<code>xcol</code>	character, representing the color to be used for plotting the streamflow time series. Only used when <code>plot=TRUE</code> .
<code>bfcol</code>	character of length 3, representing the color(s) to be used for plotting 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 <code>plot=TRUE</code> .
<code>pch</code>	numeric, representing the symbols used for plotting the streamflow time series (both, the original series and the baseflow). Only used when <code>plot=TRUE</code> .
<code>cex</code>	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 <code>par("cex")</code> . See plot.default . Only used when <code>plot=TRUE</code> .
<code>...</code>	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:

$$q(t) = \text{Beta} * q(t-1) + [(1+\text{Beta})/2] * [Q(t) - Q(t-1)]$$

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 ($\text{Beta}=0.925$). The value $\text{Beta}=0.925$ 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's 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

<code>x</code>	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.
<code>from</code>	Character indicating the starting date for subsetting <code>x</code> . It has to be in the format indicated by <code>date.fmt</code> .
<code>to</code>	Character indicating the ending date for subsetting <code>x</code> . It has to be in the format indicated by <code>date.fmt</code> .
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>from</code> and <code>to</code> , e.g. <code>%Y-%m-%d</code> . See 'Details' section in strptime .
<code>main</code>	character, Main chart title.
<code>col</code>	A color palette, i.e. a vector of <code>n</code> contiguous colors generated by functions like <code>rainbow</code> , <code>heat.colors</code> , <code>topo.colors</code> , <code>bpy.colors</code> or one of your own making, perhaps using colorRampPalette . If none is provided, a color ramp with 8 colours is created using colorRampPalette .
<code>cuts</code>	Numeric, indicating the values used to divide the range of <code>x</code> in the legend of colours. If not provided, it is automatically selected as a function of <code>length(col)</code> .
<code>cuts.dec</code>	Number of decimal places used to present the numbers in the legend of colours.
<code>cuts.labels</code>	Character indicating the label to be used in the colour legend for each one of the values defined by <code>cuts</code> . If not provided, <code>as.character(cuts)</code> is used.
<code>cuts.style</code>	discarded because takes too much time or not always provide the required number of classes: "dpih", "headtails", "hclust", "jenks".
<code>legend.title</code>	text to be displayed above the legend of colours.
<code>legend.fontsize</code>	size of text (in points) used in the legend of colours.
<code>do.png</code>	Do you want to write the figure as a .png file? logical, set TRUE to save the image.
<code>png.fname</code>	character, indicating the name of the file (possibly with a meaningful file extension) that will be used to write the output file.
<code>png.width</code>	numeric, the width of the device.
<code>png.height</code>	numeric, the height of the device.
<code>png.pointsize</code>	integer, the default pointsize of plotted text, interpreted as big points (1/72 inch) at <code>res</code> ppi.
<code>png.res</code>	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.
<code>do.pdf</code>	Do you want to write the figure as a .pdf file? logical, set TRUE to save the image.

<code>pdf.fname</code>	character, indicating the name of the file (possibly with a significant extension) that will be used to write the output file.
<code>pdf.width</code>	numeric, the width of the device.
<code>pdf.height</code>	numeric, the height of the device.
<code>pdf.pointsize</code>	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.
<code>...</code>	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 slightly modified version of the function is also distributed under the GPL-2 licence, and the 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 main title of the output figure.
- 3) it allows a customisation of the color palette.
- 4) it uses a categorical legend instead of a continuous 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 maximum 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	<i>Hydrometeorological time series for "Cauquenes en El Arrayan" catchment</i>
------------------	--

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 streamflows, [mm], measured at the "Cauquenes en El Arrayan" (7336001) station.
-) *Qobs_m3s*: Daily streamflows, [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-Garretón, 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.

change_tz	<i>Change the time zone of a sub-daily zoo object</i>
-----------	---

Description

Changes the time zone used by the time index of a sub-daily zoo object whose index inherits from POSIXct or POSIXlt.

Usage

```
change_tz(x, ...)

## S3 method for class 'zoo'
change_tz(x, new.tz, old.tz=NULL, ...)
```

Arguments

x	zoo object with a sub-daily POSIXct or POSIXlt time index.
new.tz	character. Time zone to assign to the output time index.
old.tz	character or NULL. Time zone currently used by the time index. When NULL, it is inferred from attr(time(x), "tzone"). When provided, it must match the detected time zone when the time index already has one.
...	further arguments passed to or from other methods.

Details

Both old.tz and new.tz are checked against [OlsonNames](#).

change_tz() preserves the represented instants and changes their time zone for display and downstream calendar grouping. For example, midnight UTC is shown as the previous evening in "America/Santiago" when the applicable UTC offset is negative. It does not reinterpret clock labels in a new time zone.

The values and dimensions of x are preserved; only the time index is changed.

Value

A zoo object with the same values as x, but with its time index converted to new.tz.

Author(s)

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

See Also

[zoo](#), [OlsonNames](#), [as.POSIXct](#), [subhourly2hourly](#), [subhourly2nhourly](#), [subdaily2daily](#)

Examples

```
dates <- seq(as.POSIXct("2001-01-01 00:00:00", tz="UTC"),
             by="hour", length.out=3)
x <- zoo(1:3, dates)

y <- change_tz(x, new.tz="America/Santiago")
time(y)

## The represented instants and the observed values are preserved.
identical(as.numeric(time(x)), as.numeric(time(y)))
identical(coredata(x), coredata(y))
```

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 <i>dates</i> , <i>from</i> and <i>to</i> . 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 <code>tmean</code> $\leq$ <code>pcp.solid.thr</code> the precipitation for that day is considered as solid precipitation.
<code>pcp.ylim</code>	[OPTIONAL] numeric of length 2 with the the range used for the precipitation axis. The second value should be larger than the first one.
<code>temp.ylim</code>	[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.
<code>pcp.col</code>	Color used in the legend to represent the monthly average precipitation.
<code>pcp.solid.col</code>	Color used in the legend to represent the monthly average solid precipitation.
<code>tmean.col</code>	Color used in the legend to represent the monthly average temperature.
<code>tmn.col</code>	Color used in the legend to represent the monthly minimum temperature.
<code>tmx.col</code>	Color used in the legend to represent the monthly maximum temperature.
<code>pcp.labels</code>	logical. Should monthly precipitation values to be shown above the bars?. By default <code>pcp.labels=TRUE</code> .
<code>tmean.labels</code>	logical. Should monthly mean temperature values to be shown above the lines?. By default <code>tmean.labels=TRUE</code> .
<code>tmx.labels</code>	logical. Should monthly maximum temperature values to be shown above the lines?. By default <code>tmx.labels=TRUE</code> .
<code>tmn.labels</code>	logical. Should monthly minimum temperature values to be shown above the lines?. By default <code>tmn.labels=TRUE</code> .
<code>pcp.labels.cex</code>	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 <code>pcp.labels=TRUE</code> .
<code>temp.labels.cex</code>	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 <code>tmean.labels=TRUE</code> or <code>tmx.labels=TRUE</code> or <code>tmn.labels=TRUE</code> .
<code>pcp.labels.dx</code>	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 <code>pcp.labels=TRUE</code> . Lengths smaller than 12 are recycled and larger lengths are not used.
<code>pcp.labels.dy</code>	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 <code>pcp.labels=TRUE</code> . Lengths smaller than 12 are recycled and larger lengths are not used.
<code>temp.labels.dx</code>	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 <code>tmean.labels=TRUE</code> or <code>tmx.labels=TRUE</code> or <code>tmn.labels=TRUE</code> . Lengths smaller than 12 are recycled and larger lengths are not used.
<code>temp.labels.dy</code>	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 <code>tmean.labels=TRUE</code> or <code>tmx.labels=TRUE</code> or <code>tmn.labels=TRUE</code> . Lengths smaller than 12 are recycled and larger lengths are not used.

<code>plot.pcp.probs</code>	logical used to decide whether to show uncertainty values around the monthly mean precipitation values. By default <code>plot.pcp.probs=TRUE</code> . When <code>plot.pcp.probs=TRUE</code> the <code>pcp.probs</code> argument is used to define the values of the lower an upper uncertainty bounds.
<code>pcp.probs</code>	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 <code>pcp.probs=c(0.25, 0.75)</code> , 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 <code>pcp</code> is a (sub)daily zoo object, it is first aggregated into monthly values using <code>mean</code> , and then the <code>pcp.probs</code> quantiles are computed over all the monthly values belonging to a calendar month.
<code>plot.temp.probs</code>	logical used to decide whether to show uncertainty values around the monthly mean temperature values. By default <code>plot.temp.probs=TRUE</code> . When <code>plot.temp.probs=TRUE</code> the <code>temp.probs</code> argument is used to define the values of the lower an upper uncertainty bounds.
<code>temp.probs</code>	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 <code>tmx</code> and <code>tmn</code> are provided, then <code>temp.probs</code> are used to compute the lower an upper uncertainty bounds for each one of the 12 monthly maximum/minimum temperature values. By default <code>temp.probs=c(0.25, 0.75)</code> , 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 <code>tmx/tmn</code> is provided and is a (sub)daily zoo object, it is first aggregated into monthly values using <code>mean</code> , and then the <code>temp.probs</code> quantiles are computed over all the monthly values belonging to a calendar month.
<code>temp.probs.col</code>	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 <code>tmx</code> and <code>tmn</code> are not provided by the user, the second element of <code>temp.probs.col</code> will still be used to define the color of the uncertainty band around the mean monthly values of air temperature.
<code>temp.probs.alpha</code>	numeric of length 1, with the factor used to modify the opacity of <code>temp.probs.col</code> . Typically in <code>[0,1]</code> , with 0 indicating a completely transparent colour and 1 indicating no transparency.
<code>lat</code>	[OPTIONAL] numeric or character used to show the latitude for which the climograph was plotted for.
<code>lon</code>	[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 precipitation 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

<code>x</code>	zoo, data.frame or matrix object, with the time series to be analysed. Measurements at several gauging stations can be stored in a data.frame or matrix object, and in that case, each column of <code>x</code> represents the time series measured in a gauging station, and the column names of <code>x</code> have to correspond to the ID of each station (starting by a letter).
<code>tscale</code>	character with the temporal scale to be used for analysing the missing data. Valid values are: -) hourly: the percentage/amount of missing values will be given for each hour and, therefore, the expected time frequency of <code>x</code> must be sub-hourly. -) daily: the percentage/amount of missing values will be given for each day and, therefore, the expected time frequency of <code>x</code> 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 <code>x</code> 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 <code>x</code> 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 <code>x</code> 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 <code>?time2season</code>) and, therefore, the expected time frequency of <code>x</code> 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 <code>x</code> must be sub-annual (i.e., seasonal, monthly, daily, hourly or sub-hourly).
<code>dec</code>	integer indicating the amount of decimal places included in the output. It is only used when <code>out.type == 'percentage'</code>.
<code>start</code>	character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by <code>start.fmt</code>. This value is used to define the time when a new day begins (e.g., for some rain gauge stations). -) All the values of <code>x</code> with a time attribute before <code>start</code> are considered as belonging to the day before the one indicated in the time attribute of those values. -) All the values of <code>x</code> with a time attribute equal to <code>start</code> are considered to be equal to "00:00:00" in the output zoo object. -) All the values of <code>x</code> with a time attribute after <code>start</code> 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.
<code>start.fmt</code>	character indicating the format in which the time is provided in <code>start</code> , By default <code>date.fmt=%H:%M:%S</code> . See format in as.POSIXct .
<code>start.month</code>	numeric in [1,...,12], representing the starting month (1:Jan, ..., 12:Dec) to be used in the computation of annual values. By default <code>start.month=1</code> .
<code>tz</code>	character, with the specification of the time zone used in both <code>x</code> and <code>start</code> . 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 <code>tz</code> is missing (the default), it is automatically set to the time zone used in <code>time(x)</code> . This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.
<code>dates</code>	numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of <code>x</code> (e.g., gauging station). If <code>dates</code> is a number, it indicates the index of the column in <code>x</code> that stores the date and times. If <code>dates</code> is a factor, it is converted into POSIXct class, using the date format specified by <code>date.fmt</code> . If <code>dates</code> 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 <code>x</code> .
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>dates</code> , By default <code>date.fmt=%Y-%m-%d %H:%M:%S</code> . See format in as.Date . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
<code>out.type</code>	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.
<code>...</code>	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	<i>Daily -> Weekly</i>
--------------	---------------------------

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,
             week.date.format = "%Y-%W", week.grouping = "calendar", ...)

## S3 method for class 'zoo'
daily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0,
             week.date.format = "%Y-%W", week.grouping = "calendar", ...)

## S3 method for class 'data.frame'
daily2weekly(x, FUN, na.rm=TRUE, na.rm.max=0, dates=1,
             date.fmt = "%Y-%m-%d", week.date.format = "%Y-%W",
             week.grouping = "calendar", 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", week.date.format = "%Y-%W",
             week.grouping = "calendar", 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).
na.rm	FUN MUST accept the na.rm argument, because na.rm is passed to FUN. 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 corresponding 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".
week.date.format	character indicating how weekly dates are stored in the output object. Valid values are: -) %Y-%W: year and Monday-based week number, preserving the historical behaviour. -) %Y-%m-%d: date of the first day available in each weekly aggregation group, returned as a Date index in zoo outputs and displayed as YYYY-MM-DD.
week.grouping	character indicating how daily values are grouped into weekly values. Valid values are: -) calendar: calendar year-week groups defined by %Y-%W, preserving the historical behaviour. Weeks crossing a calendar-year boundary are split. -) sequential: consecutive 7-day groups starting at the first date in x, equivalent to seq(start(x), end(x), by="weeks") for a complete daily series.
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 weekly label 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 (Week) stores the week number when week.date.format="%Y-%W" and the weekly date when week.date.format="%Y-%m-%d" The fourth column (Value) stores the numerical values corresponding to the val-

ues specified in the three previous columns.

<code>out.fmt</code>	OPTIONAL. Only used when <code>x</code> 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: <code>numeric</code> , <code>zoo</code> .
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	arguments additional to <code>na.rm</code> passed to FUN.

Details

The argument `week.grouping` defines how observations are assigned to weekly aggregation groups before FUN is applied. It is independent from `week.date.format`, which only controls how the weekly groups are labelled in the output object.

When `week.grouping="calendar"` (default), `daily2weekly` groups the input dates using `format(time(x), "%Y-%W")`. This is the historical behaviour of the function. The `%W` conversion uses Monday as the first day of the week and numbers weeks within each calendar year. Days before the first Monday of a calendar year belong to week "00". Therefore, a Monday-Sunday week that crosses from December into January can be split into two aggregation groups, one for the ending calendar year and one for week "00" or "01" of the new calendar year.

When `week.grouping="sequential"`, `daily2weekly` groups the input dates into consecutive 7-day periods starting at the first date in `x`. For a complete daily series, the resulting weekly labels with `week.date.format="%Y-%m-%d"` correspond to `seq(start(x), end(x), by="weeks")`. This option is useful when the user wants uninterrupted 7-day blocks over the full record instead of calendar year-week groups.

The argument `week.date.format` controls only the time index returned by `daily2weekly`. With `week.date.format="%Y-%W"`, the output is labelled with year-week strings. With `week.date.format="%Y-%m-%d"`, the output is labelled with the first date of each weekly group. The missing-value filter controlled by `na.rm.max` is applied to the same groups defined by `week.grouping`.

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

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

## Same aggregation, with the weekly zoo index shown as YYYY-MM-DD dates
w.dates <- daily2weekly(x, FUN=sum, na.rm=TRUE, week.date.format="%Y-%m-%d")

## Aggregation using consecutive 7-day groups starting at the first date in 'x'
w.seq <- daily2weekly(x, FUN=sum, na.rm=TRUE, week.grouping="sequential",
                      week.date.format="%Y-%m-%d")

#####
## 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" ##
diy("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.
diy("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

<code>year</code>	numeric, the year for which the sequence of days will be generated
<code>out.type</code>	Character indicating the type of result that is given by this function. Valid values are: -) <code>seq</code> : a vectorial sequence with all the days within the given year -) <code>nmbr</code>: 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	<i>(sub)Daily/Monthly -> Seasonal Values</i>
-------------	---

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 <i>dates</i>, 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	<i>Customized Time Axis</i>
--------------	-----------------------------

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	<i>Amount of dry/wet days in a time series</i>
--------	--

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 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 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 <i>wet days</i> , whereas precipitation values lower to thr are classified as <i>dry days</i> .

type	character, indicating if the daily values have to be classified as dry or wet days. It works linked to the values specified in <code>thr</code> . 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 <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates If <code>dates</code> is a factor, it is converted into Date class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of Date class, the code verifies that the number of days in <code>dates</code> be equal to the number of element in <code>x</code>
date.fmt	character indicating the format in which the dates are stored in <code>dates</code> , e.g. <code>%Y-%m-%d</code> . See format in as.Date . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
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 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 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 <i>dates</i>, 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. Month-vector order is relevant: c(9,10,11,12,1,2,3) extracts September to March, with January, February and March belonging to the

next calendar year, whereas `c(1,2,3,9,10,11,12)` extracts all those months within each calendar year.

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. Numeric vectors must contain either only months or only years.

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

-) SONDJFM: September, October, November, December, January, February, March, with January, February and March belonging to the next calendar year

... 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 values from September to March
extract(x, trgt=c(9,10,11,12,1,2,3))

## Extracting January-March and September-December within the same calendar year
extract(x, trgt=c(1,2,3,9,10,11,12))

## Extracting all values from September to March using the SONDJFM shortcut
extract(x, trgt="SONDJFM")
```

```
## 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 stream-flow 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 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 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 be 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 .

<code>lty</code>	The line type, see <code>par</code> . See <code>plot.default</code> .
<code>cex</code>	See <code>plot.default</code> . 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 <code>par("cex")</code> . 'NULL' and 'NA' are equivalent to '1.0'. Note that this does not affect annotation
<code>cex.axis</code>	magnification of axis annotation relative to 'cex'.
<code>cex.lab</code>	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
<code>leg.txt</code>	vector with the names that have to be used for each column of x.
<code>leg.cex</code>	numeric, indicating the character expansion factor for the legend, *relative* to current <code>par("cex")</code> . Default value = 1
<code>leg.pos</code>	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 <code>legend</code> .
<code>thr.pos</code>	keyword to be used to position the streamflow values corresponding to the user-defined thresholds <code>lQ.thr</code> and <code>hQ.thr</code> . It is only used when <code>thr.shw=TRUE</code> . 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 <code>legend</code> .
<code>verbose</code>	logical; if TRUE, progress messages are printed (when x is a matrix or data.frame).
<code>thr.shw</code>	logical, indicating if the streamflow values corresponding to the user-defined thresholds <code>lQ.thr</code> and <code>hQ.thr</code> have to be shown in the plot.
<code>new</code>	logical, if TRUE (default), a new plotting window is created.
<code>...</code>	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)
```

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.

<code>xat</code>	Only used when <code>log="x"</code> . 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 <code>at</code> argument in Axis .
<code>col</code>	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.
<code>pch</code>	See plot.default . A vector of plotting characters or symbols: see points .
<code>lwd</code>	See plot.default . The line width, see par .
<code>lty</code>	See plot.default . The line type, see par .
<code>cex</code>	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 <code>par("cex")</code> . 'NULL' and 'NA' are equivalent to '1.0'. Note that this does not affect annotation.
<code>cex.axis</code>	magnification of axis annotation relative to 'cex'.
<code>cex.lab</code>	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
<code>leg.txt</code>	vector with the names that have to be used for each column of x.
<code>leg.cex</code>	numeric, indicating the character expansion factor for the legend, *relative* to current <code>par("cex")</code> . Default value = 1
<code>leg.pos</code>	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 <code>leg.pos="auto"</code> , the legend provided by <code>leg.txt</code> is located on the 'bottomleft' when <code>log="y"</code> and on the 'topright' otherwise
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>thr.shw</code>	logical, indicating if the streamflow values corresponding to the user-defined thresholds <code>lQ.thr</code> and <code>hQ.thr</code> have to be shown in the plot. When <code>leg.pos="auto"</code> , the legend with the threshold values is located on the 'topright' when <code>log="y"</code> and on the 'bottomleft' otherwise
<code>border</code>	See polygon . The color to draw the border of the polygon with the uncertainty bounds. The default, 'NA', means to omit borders.
<code>bands.col</code>	See polygon . The color for filling the polygon. The default, 'NA', is to leave polygons unfilled, unless <code>bands.density</code> is specified. If <code>bands.density</code> is specified with a positive value this gives the color of the shading lines.
<code>bands.density</code>	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 <code>bands.density</code> means no shading nor filling whereas negative values (and 'NA') suppress shading (and so allow color filling).
<code>bands.angle</code>	See polygon . The slope of shading lines for the polygon with the uncertainty bounds, given as an angle in degrees (counter-clockwise).
<code>new</code>	logical, if TRUE, a new plotting window is created.
<code>...</code>	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 to 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 <code>cor</code> . 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 <code>cor</code> . 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 <code>pairs</code> function.

Value

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

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=c("ts+boxplot+hist", "ts+boxplot", "ts+hist", "ts"),
          pfreq=c("dma", "dm", "ma", "seasonal", "o"),
          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=c("ts+boxplot+hist", "ts+boxplot", "ts+hist", "ts"),
          pfreq=c("dma", "dm", "ma", "seasonal", "o"),
          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=c("ts+boxplot+hist", "ts+boxplot", "ts+hist", "ts"),
  pfreq=c("dma", "dm", "ma", "seasonal", "o"),
  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 boxplots 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.

<code>win.len2</code>	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.
<code>tick.tstep</code>	Character indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: -) days, -) months, -) years
<code>lab.tstep</code>	Character indicating the time step that have to be used for putting the labels on the time axis. Valid values are: -) days, -) months, -) years
<code>lab.fmt</code>	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 <code>lab.tstep=="days"</code> , "%b" when <code>lab.tstep=="month"</code> , and "%Y" when <code>lab.tstep=="year"</code> .
<code>cex</code>	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. (See par).
<code>cex.main</code>	The magnification to be used for main titles relative to the current setting of <code>cex</code> (See par).
<code>cex.lab</code>	The magnification to be used for x and y labels relative to the current setting of <code>cex</code> (See par).
<code>cex.axis</code>	The magnification to be used for axis annotation relative to the current setting of <code>cex</code> (See par).
<code>col</code>	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 <code>pfreq="o"</code> , only one character element is needed. See plot.default).
<code>dates</code>	ONLY required when <code>x</code> is a data frame. It is a numeric, factor or Date object indicating how to obtain the dates corresponding to the <code>sname</code> station. If <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates If <code>dates</code> is a factor, it is converted into Date class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of Date class, the code verifies that the number of days in <code>dates</code> be equal to the number of element in <code>x</code>
<code>date.fmt</code>	Character indicating the format in which the dates are stored in <code>dates</code> , <code>from</code> and <code>to</code> . See format in as.Date . It is used whenever character values are supplied in <code>dates</code> , <code>from</code> , or <code>to</code> . For sub-daily series, it must include the time components used by those arguments.
<code>tz</code>	character, with the specification of the time zone used for <code>x</code> , <code>from</code> , and <code>to</code> . 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 <code>tz</code> is NULL (the default), it is automatically set to the time zone used in <code>time(x)</code> .

If `tz` is provided, it forces `time(x)` to be in the time zone specified by `tz`, without modifying 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.

<code>from</code>	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 <code>date.fmt</code> .
<code>to</code>	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 <code>date.fmt</code> .
<code>stype</code>	OPTIONAL, only used when <code>pfreq=seasonal</code> . 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
<code>season.names</code>	OPTIONAL, only used when <code>pfreq=seasonal</code> . character of length 4 indicating the names of each one of the weather seasons defined by <code>stype</code> . These names are only used for plotting purposes
<code>h</code>	OPTIONAL, only used when <code>pfreq=seasonal</code> , 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).
<code>...</code>	further arguments passed to the <code>plot.zoo</code> and <code>axis</code> 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

<code>x</code>	numeric, data.frame or matrix in which some values are missing (NA).
<code>sim</code>	numeric, data.frame or matrix, with the same dimension of <code>x</code> , which contains the values that will be used for infilling the missing (NA) values in <code>x</code>
<code>...</code>	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

<code>x</code>	zoo object with a time attribute that should have a regular time frequency (e.g., hourly, daily, monthly, annual).
<code>tz</code>	character, indicating the time zone in which the date of <code>x</code> 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 time zone specified by `tz`, without modifying 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.

<code>out.type</code>	Character indicating the type of result that is given by this function. Valid values are: -) <code>single</code> => only a logical output is returned. See details in the Value section. -) <code>all</code> => a list with 3 elements is returned. See details in the Value section.
<code>verbose</code>	logical; if TRUE, informative messages are shown in the screen. When <code>verbose=TRUE</code> and <code>x</code> has some missing temporal layers, the <code>DateTime(s)</code> with missing layer(s) are shown in the screen.
<code>...</code>	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`. It is an empty vector when no DateTime is missing.

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)

```

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 $z = \text{re-scale}(x) = x * [x_{\max} - x_{\min}] + x_{\min}$.

Usage

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

Arguments

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

<code>x</code>	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)
<code>from</code>	Character indicating the starting date for creating the regularly spaced zoo object. The default value corresponds to the date of the first element of <code>x</code> . It has to be in the format indicated by <code>date.fmt</code> .
<code>to</code>	Character indicating the ending date for creating the regularly spaced zoo object. The default value corresponds to the date of the last element of <code>x</code> . It has to be in the format indicated by <code>date.fmt</code> .
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>from</code> and <code>to</code> , e.g. <code>%Y-%m-%d</code> . See 'Details' section in strptime . By default, <code>date.fmt</code> is missing, and it is automatically set to <code>%Y-%m-%d</code> when <code>time(x)</code> is Date object, and set to <code>%Y-%m-%d %H:%M:%S</code> when <code>x</code> is a sub-daily zoo object.
<code>tstep</code>	character, indicating the time step used for creating the time sequence going from <code>from</code> to <code>to</code> that will be used as <code>time(x)</code> Valid values are (but not limited to) hours, days, months, years. By default, <code>tstep</code> is missing, and it is automatically set to "minutes" when <code>sfreq(x)</code> is min, to "hours" when <code>sfreq(x)</code> is hourly, to "days" when <code>sfreq(x)</code> is daily, to "weeks" when <code>sfreq(x)</code> is weekly, to "months" when <code>sfreq(x)</code> is monthly, to "quarters" when <code>sfreq(x)</code> is quarterly, and to "years" when <code>sfreq(x)</code> is annual.
<code>tz</code>	character, with the specification of the time zone used for <code>x</code> , <code>from</code> , and <code>to</code> . 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 <code>tz</code> is missing (the default), it is automatically set to the time zone used in <code>time(x)</code> . If <code>tz</code> is provided, it forces <code>time(x)</code> to be in the time zone specified by <code>tz</code> , without modifying the the values (hours, minutes, seconds, etc). A list of valid time zones can be obtained by calling the base function <code>OlsonNames()</code> . This argument can be used when working with sub-daily zoo objects to force using time zones other than the local time zone for <code>from</code> and <code>to</code> . It should be used with caution, being well aware of the time zone of the data. See examples.
<code>na.action</code>	character indicating whether to keep 'NA' values in <code>x</code> (default) or interpolate them with linear or spline interpolation using the na.approx or the na.spline function of the zoo package.
<code>...</code>	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' (there 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 omitted
(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 omitted
(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 omitted 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 treated 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 set 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 occurred. However, due to CRAN requirements, in January 2024 (for hydroTSM v0.7-1), the time zone of this data set was changed from "GMT+12" to UTM, in order to avoid missing datetimes at times where daylight saving time occurred. 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	<i>Moving Average</i>
----	-----------------------

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 minimum 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

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

Details

The default `levelplot` panel draws dotted grey lines along cell boundaries, making individual matrix values easier to distinguish.

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	<i>Months in Period</i>
-----	-------------------------

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	<i>Monthly Function</i>
-----------------	-------------------------

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 <i>dates</i>, e.g. %Y-%m-%d. See format in as.Date. ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code>.
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)

```

nperiods

Number of Years, Months, Weeks, Days, or Hours in a zoo Object

Description

Generic functions for computing the temporal span covered by a [zoo](#) object, expressed as years, months, weeks, days, or hours.

Usage

```

nyears(x, ...)
nmonths(x, ...)
nweeks(x, ...)
ndays(x, ...)
nhours(x, ...)

## S3 method for class 'zoo'
nyears(x, only.integer.output=FALSE, tz, verbose=TRUE, ...)
## S3 method for class 'zoo'
nmonths(x, only.integer.output=FALSE, tz, verbose=TRUE, ...)
## S3 method for class 'zoo'
nweeks(x,
        week.grouping="calendar",
        verbose=TRUE, ...)
## S3 method for class 'zoo'
ndays(x, only.integer.output=FALSE, tz, verbose=TRUE, ...)
## S3 method for class 'zoo'
nhours(x, only.integer.output=FALSE, tz, verbose=TRUE, ...)

```

Arguments

x A zoo object with a regular time index.

<code>only.integer.output</code>	logical; if FALSE (default), decimal values are returned when the last year, month, day, or hour is not complete. If TRUE, only the number of whole years, months, days, or hours is returned, and a warning is issued when the last period is incomplete.
<code>week.grouping</code>	character indicating how weeks are counted by <code>nweeks</code> . Valid values are: -) calendar: distinct calendar year-week groups defined by %Y-%W. Weeks crossing a calendar-year boundary are split. -) sequential: consecutive 7-day groups starting at the first date in <code>x</code> . This argument is fully compatible with daily2weekly .
<code>tz</code>	character, indicating the time zone in which the time index of <code>x</code> is located. If missing, the time zone stored in the index is used when available. See isComplete .
<code>verbose</code>	logical; if TRUE, informative messages are shown in the screen. The functions call isComplete and, when <code>x</code> has missing date/time values, they are reported by <code>isComplete</code> .
<code>...</code>	further arguments passed to or from other methods.

Details

The returned value is computed from the first date/time to the inferred end of the last observation. The end of the last observation is inferred from the regular time step of `x`. For example, a daily object from 2020-01-01 to 2020-01-31 returns 1 month and 31 days.

Before computing the temporal span, the functions call `isComplete(x, out.type="all")`. Therefore, incomplete objects are still accepted when their time index has a regular frequency, but missing date/time values are reported when `verbose=TRUE`.

For `nweeks`, the result is the number of weekly groups that `daily2weekly` would produce with the same `week.grouping`.

Value

A numeric value with the temporal span of `x` expressed in the requested unit. `nweeks` returns the integer number of weekly groups. For the other functions, an integer value is returned when `only.integer.output=TRUE`.

Author(s)

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

See Also

[isComplete](#), [daily2weekly](#), [zoo](#)

Examples

```
dates <- as.Date("2020-01-01") + 0:30
x <- zoo::zoo(seq_along(dates), dates)

nyears(x, verbose=FALSE)
```

```

nmonths(x, verbose=FALSE)
nweeks(x, verbose=FALSE)
ndays(x, verbose=FALSE)
nhours(x, verbose=FALSE)

y <- x[1:15]
ndays(y, verbose=FALSE)
ndays(y, only.integer.output=TRUE, verbose=FALSE)

```

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 <code>p.probs=c(0.25, 0.75)</code>, 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 <code>p</code> is a (sub)daily zoo object, it is first aggregated into monthly values using <code>mean</code>, and then the <code>p.probs</code> 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 <code>p.col</code>. Typically in <code>[0,1]</code>, 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 <code>plot.q.probs=TRUE</code>. When <code>plot.q.probs=TRUE</code> the <code>q.probs</code> 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 <code>q</code> is a (sub)daily zoo object, it is first aggregated into monthly values using <code>FUN</code>, and then the <code>q.probs</code> quantiles are computed over all the monthly values belonging to a calendar month.. By default <code>q.probs=c(0.25, 0.75)</code>, 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 <code>q</code> is provided and is a (sub)daily zoo object, it is first aggregated into monthly values using <code>FUN</code>, and then the <code>q.probs</code> 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 <code>q.probs.col</code>. Typically in <code>[0,1]</code>, 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 <code>labels=TRUE</code>.
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 <code>labels=TRUE</code>. 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 <code>labels=TRUE</code>. 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)
```

precipQC

*Frequency-aware quality control of precipitation time series***Description**

Detects the sampling frequency of a precipitation zoo object with `sfreq` and dispatches to `precipQC_daily` or `precipQC_subdaily`. No files are written.

Usage

```
precipQC(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"), ...,
  elevation=NULL, elevation.scale=500, verbose=TRUE
)
```

Arguments

<code>x</code>	A univariate or multivariate numeric zoo object. Each column is one precipitation station and must have a unique name.
<code>metadata, station.id, coords, elevation, elevation.scale</code>	Optional station metadata, field-name selectors, and elevation-decay control passed to the selected workflow. See <code>precipQC_daily</code> .
<code>...</code>	Arguments passed unchanged to the selected QC function. Because the daily and sub-daily functions have resolution-specific arguments, only arguments accepted by the detected workflow should be supplied.
<code>verbose</code>	Logical. If TRUE, informative messages report the detected frequency, the active QC stages, spatial scales, and the final QC summary. If FALSE, these informational messages are suppressed; warnings and errors are unaffected.

Details

`sfreq(x)` returning "daily" selects `precipQC_daily(x, ...)`. Values "minute" and "hourly" select `precipQC_subdaily(x, ...)`. Weekly, monthly, quarterly, annual, or other frequencies are rejected with an informative error because the implemented thresholds and tests are not valid at those resolutions. Progress is emitted with `message()`, so it can be captured or suppressed without redirecting returned values.

Value

An object of class "precipQC"; see [precipQC-class](#).

Author(s)

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

See Also

[sfreq](#), [precipQC_daily](#), [precipQC_subdaily](#), [plot.precipQC](#)

Examples

```
dates <- seq(as.Date("2020-01-01"), by="day", length.out=40)
x <- zoo(cbind(A=rep(0, 40), B=rep(0, 40)), dates)

qc <- precipQC(
  x, min.years=0,
  checks=c(climatology=FALSE, spatial=FALSE, dryspell=FALSE,
            breakpoint=FALSE)
)
qc$settings$resolution
```

```
precipQC-class
```

```
Precipitation quality-control result
```

Description

Print and plot methods for objects returned by [precipQC](#), [precipQC_daily](#), and [precipQC_subdaily](#).

Usage

```
## S3 method for class 'precipQC'
print(x, ...)

## S3 method for class 'precipQC'
plot(
  x, max.stations=20L,
  col=c("#2878B5", "#D55E00", "#E69F00"), ...
)
```

Arguments

x	An object of class "precipQC".
max.stations	Maximum number of stations displayed in the missing/suspicious percentage panel. Stations with the largest suspicious and missing fractions are shown.
col	At least three colours for accepted stations, discarded/rejected values, and missing/review flags.

... For plot, additional arguments passed to the final timeline [plot](#) call. Ignored by print.

Details

The plot contains four panels: numbers of accepted and discarded stations; missing and confirmed-suspicious percentages for the most affected stations; flag counts by individual test; and the number of confirmed suspicious values through time.

Value

Both methods return `x` invisibly. The print method writes a compact summary to the console and the plot method draws on the active graphics device.

Author(s)

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

See Also

[precipQC](#), [precipQC_daily](#), [precipQC_subdaily](#), [precipQC_range](#)

```
precipQC_daily
```

```
Quality control of daily precipitation time series
```

Description

Applies a conservative, multi-test quality-control workflow to one or more daily precipitation series in a zoo object. Individual observations are flagged for review or rejection, stations are recommended for acceptance or discarding, and no files are written.

Usage

```
precipQC_daily(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  checks=c(range=TRUE, duplicate=TRUE, frequency=TRUE, gap=TRUE,
            climatology=TRUE, persistence=TRUE, accumulation=TRUE,
            weekday=TRUE, spatial=TRUE, dryspell=TRUE, breakpoint=TRUE),
  lower=0, upper=1825, wet.threshold=0.1,
  duplicate.min.month=20L, duplicate.min.year=300L,
  duplicate.min.nonzero=3L,
  frequency.window=10L, frequency.min.samples=20L,
  gap.threshold=300, gap.min.samples=30L,
  climatology.window=15L, climatology.prob=0.999,
  climatology.z=8, climatology.min.samples=30L,
  persistence.threshold=10, persistence.run=5L,
  weekday.min.wet=20L, weekday.alpha=0.001,
  weekday.ratio=0.5,
```

```

    spatial.days=c(1L, 3L, 7L), spatial.cr=3,
    n.neighbours=10L, max.distance=400, min.neighbours=2L,
    min.overlap=30L, min.correlation=0,
    dryspell.days=15L, neighbour.wet.days=3L,
    neighbour.fraction=1,
    correction=c("none", "set_na", "spatial"),
    min.evidence=2L, max.missing=0.2, max.suspicious=0.05,
    min.years=1, discard.breakpoint=FALSE,
    elevation=NULL, elevation.scale=500, verbose=TRUE
  )

```

Arguments

<code>x</code>	A zoo object with a daily Date or POSIXt index. Each column is one precipitation station and must have a unique name. Values are assumed to be precipitation depths in mm per day.
<code>metadata</code>	NULL, or a data.frame with one row per station. Extra columns are preserved in the accepted and discarded metadata outputs. When supplied, it must contain the identifier, longitude, and latitude fields selected by <code>station.id</code> and <code>coords</code> . Rows may be in any order and extra stations are allowed. When NULL, spatial tests retain the correlation-only behaviour and generated station metadata are returned.
<code>station.id</code>	Character string naming the metadata column that contains the column names of <code>x</code> .
<code>coords</code>	Two character strings naming longitude and latitude columns in metadata, in that order. Both columns must be numeric and expressed in decimal degrees. Individual entries may be NA; a station has a usable location only when both entries are present. Non-missing longitude and latitude values must be finite and lie in $[-180, 180]$ and $[-90, 90]$, respectively.
<code>elevation</code>	NULL, or a character string naming an optional numeric metadata column of station elevations in metres. It is used only when explicitly selected.
<code>elevation.scale</code>	Positive elevation-decay scale in metres. With elevation metadata, candidate neighbours and their spatial weights are penalized by $\exp(-\text{abs(dz)} / \text{elevation.scale})$. A smaller value gives elevation difference more influence. The argument has no effect when <code>elevation=NULL</code> .
<code>checks</code>	Named logical vector. Any subset of "range", "climatology", "duplicate", "frequency", "gap", "persistence", "accumulation", "weekday", "spatial", "dryspell", and "breakpoint" can be supplied to override the default of running every check.
<code>duplicate.min.month</code> , <code>duplicate.min.year</code> , <code>duplicate.min.nonzero</code>	Minimum complete paired month positions, complete paired year positions, and non-zero totals in both blocks for copied-block detection.
<code>frequency.window</code> , <code>frequency.min.samples</code>	Number of consecutive non-zero reports examined for clustered identical values and minimum calendar-month wet sample used for their percentile screen.

<code>gap.threshold, gap.min.samples</code>	Minimum adjacent separation in a sorted calendar-month wet distribution and minimum wet sample required for that check.
<code>lower, upper</code>	Physical lower and upper limits, in mm/day. The default upper limit is the historical 24-hour world record used by Hamada et al. (2011). Regional limits should be supplied when authoritative values are available.
<code>wet.threshold</code>	Depth in mm separating dry and wet observations.
<code>climatology.window</code>	Odd or even positive integer giving the width, in calendar days, of the moving day-of-year climatological window.
<code>climatology.prob</code>	Upper empirical probability used by the climatological test.
<code>climatology.z</code>	Number of robust standard deviations above the median on the log1p-transformed scale.
<code>climatology.min.samples</code>	Minimum number of observations needed to estimate a climatological limit.
<code>persistence.threshold</code>	Minimum repeated non-zero value, in mm/day, to which <code>persistence.run</code> is applied.
<code>persistence.run</code>	Minimum run length for repeated values at or above <code>persistence.threshold</code> . The paper-based default flags five or more identical values of at least 10 mm/day.
<code>weekday.min.wet</code>	Minimum number of wet days required in a station-year before the weekday false-zero test is attempted.
<code>weekday.alpha</code>	Significance level for the exposure-adjusted chi-squared comparison of wet-day occurrence among the seven weekdays.
<code>weekday.ratio</code>	Maximum ratio of observed to expected wet days used to identify an under-recorded weekday after the overall weekday test is significant. It must be in (0, 1).
<code>spatial.days</code>	Positive integers giving the rolling accumulation durations, in days, at which spatial consistency is tested. A flag at an accumulated endpoint is expanded to every contributing daily value.
<code>spatial.cr</code>	Critical ratio for transformed observed-minus-neighbour-estimate residuals. The default follows the initial value used by El Hachem et al. (2022).
<code>n.neighbours</code>	Maximum number of neighbours used for each target station.
<code>max.distance</code>	Maximum station-neighbour distance in km when coordinates are available.
<code>min.neighbours</code>	Minimum number of simultaneous valid neighbours needed for a spatial estimate.
<code>min.overlap</code>	Minimum number of paired observations required when selecting neighbours.
<code>min.correlation</code>	Minimum Spearman correlation preferred when selecting neighbours. When too few preferred neighbours remain, the best eligible neighbours are used.
<code>dryspell.days</code>	Length of a target-station dry window to compare with neighbours.

<code>neighbour.wet.days</code>	Number of wet days a neighbour must have during a target-station dry window.
<code>neighbour.fraction</code>	Fraction of available neighbours that must corroborate a suspicious dry window. It must be in $(0, 1]$.
<code>correction</code>	Correction policy. "none" (the default) preserves all observations; "set_na" replaces confirmed suspicious observations by NA; "spatial" uses the native-scale spatial estimate when available and NA otherwise.
<code>min.evidence</code>	Number of independent non-hard flags required to reject an observation. Range, copied-block, clustered frequent-value, distribution-gap, persistence, accumulated-total, and spatially corroborated dry-spell flags are treated as hard evidence. A value with less evidence remains in the suspicious table with action "review".
<code>max.missing</code>	Maximum missing fraction allowed for a station recommendation of "accept". Missing timestamps inside the record span are included.
<code>max.suspicious</code>	Maximum fraction of confirmed suspicious observations allowed for a station recommendation of "accept".
<code>min.years</code>	Minimum record length in years required for a station recommendation of "accept". The default is intended for general QC; analyses of extremes usually require substantially longer records.
<code>discard.breakpoint</code>	Logical. If TRUE, a large significant breakpoint also causes a station recommendation of "discard". It is FALSE by default because an apparent change may be climatic rather than instrumental.
<code>verbose</code>	Logical. If TRUE, informative messages identify the workflow start, each enabled test and spatial scale, evidence combination, and the final station/observation summary. If FALSE, only these informational messages are suppressed; warnings and errors remain visible.

Details

The workflow combines tests that address different error mechanisms instead of treating one statistical anomaly as proof of error.

1. *Physical range*: negative depths and values above a defensible record limit are impossible or require review. Record limits are screening limits, not estimates of local return levels. This is the gross-error stage used by Hamada et al. (2011) and Estevez et al. (2022).
2. *Climatological plausibility*: precipitation is transformed with $\log_{1p}$ to reduce positive skew. Within a moving 15-day calendar window, an upper limit combines an empirical quantile with a robust median/MAD limit. The largest exceedance is removed iteratively before the limit is recomputed. This is a robust adaptation of the calendar-window outlier test of Hamada et al. (2011), consistent with the robust and month-aware limits discussed by Stepanek et al. (2009), Estevez et al. (2022), and Golkhatmi and Farzandi (2024).
3. *Persistence*: identical non-zero values repeated for several days may indicate a stuck sensor, copied record, or truncated measurement. The defaults reproduce the daily criterion in Hamada et al. (2011), while Vicente-Serrano et al. (2010) and Estevez et al. (2022) independently use repeated-value screening in daily databases.

4. *Copied blocks, clustered identical values, and distribution gaps*: complete identical years or months, tiered frequent identical values among ten non-zero reports, and a wet-distribution tail separated by at least 300 mm are flagged as hard recording-integrity evidence. These tests reproduce the daily precipitation duplicate, frequent-value, and gap checks of Durre et al. (2010), with explicit minimum samples to avoid declaring sparse blocks identical.
5. *Accumulated totals*: a large value immediately after missing days is flagged as a possible multi-day accumulation. Automatic redistribution is not attempted because the period and total must be known reliably. Scherrer et al. (2011) redistributed known accumulated totals proportionally to spatial estimates.
6. *Systematic weekday false zeros*: within each station-year, the distribution of wet days is compared with the weekday exposure. If the distribution is highly inconsistent and a weekday records at most the requested fraction of its expected wet days, dry values on that weekday are flagged for review. This operationalizes the weekday-zero quality component of Estevez et al. (2022), with exposure adjustment to avoid mistaking uneven missingness for under-recording.
7. *Multi-scale spatial consistency*: each station is removed in turn, its transformed precipitation is estimated from up to ten nearby stations, and the residual is standardized by local and historical robust dispersion. The test is repeated for 1-, 3-, and 7-day rolling totals by default. Geographic distance limits the candidate set. If elevation is supplied, height similarity also affects neighbour ranking and the inverse-distance weights, which avoids treating a nearby station across a large orographic contrast as equally representative. A station missing either coordinate remains testable: its neighbours are selected by temporal correlation. For a located target, eligible located neighbours are preferred and unlocated neighbours are conservative fallbacks that are not subjected to max.distance because their distance is unknown. This combines the neighbour logic of Scherrer et al. (2011) with the multi-aggregation critical-ratio idea of El Hachem et al. (2022). Relative candidate/reference comparisons are also supported by Stepanek et al. (2009) and Vicente-Serrano et al. (2010).
8. *Spatially corroborated dry spells*: a long target-station dry period is flagged only when the selected neighbours repeatedly record wet days, adapting the GSDR-QC dry-neighbour rule of Lewis et al. (2021) and the suspect-zero comparisons of Vicente-Serrano et al. (2010) and Golkhatmi and Farzandi (2024).
9. *Homogeneity*: Pettitt rank tests are applied to sufficiently complete annual precipitation totals, wet-day counts, maxima, and counts above the station's wet-day 99th percentile. P-values are Holm-adjusted and the strongest diagnostic is reported. The multi-indicator design follows Vicente-Serrano et al. (2010); the cautious interpretation and use of multiple evidence follows Stepanek et al. (2009). The breakpoint is diagnostic by default because station moves, equipment changes, and true climatic changes can produce similar signals.

Convective precipitation can be genuinely local. Therefore a spatial flag by itself is labelled for review, and default correction is disabled. Confirmed rejection requires either a hard physical/recording test or at least min.evidence coincident flags. Correlation-only results for stations without coordinates should be interpreted more cautiously because geographic proximity cannot be verified.

Value

An object of class "precipQC", which is a list containing:

- `accepted.metadata` and `discarded.metadata`: station metadata plus QC diagnostics and the recommendation reason;

- `accepted.data`: unmodified daily series for accepted stations;
- `accepted.corrected`: accepted-station series after the requested correction policy;
- `suspicious`: one row per flagged station-time value, including the tests, evidence count, action, spatial estimate, and correction;
- `corrections`: an audit table of changed values;
- `flags`: named logical zoo objects returned by the individual tests;
- `flag.count`, `rejected`, `station.summary`, `breakpoint`, `spatial.diagnostics`, `neighbours`, and `settings`.

Author(s)

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

References

- Durre, I., Menne, M. J., Gleason, B. E., Houston, T. G. and Vose, R. S. (2010). Comprehensive automated quality assurance of daily surface observations. *Journal of Applied Meteorology and Climatology*, 49, 1615–1633.
- Hamada, A., Arakawa, O. and Yatagai, A. (2011). An automated quality control method for daily rain-gauge data. *Global Environmental Research*, 15, 183–192.
- Scherrer, S. C., Begert, M., Croci-Maspoli, M. and Appenzeller, C. (2011). Operational quality control of daily precipitation using spatio-climatological plausibility testing. *Meteorologische Zeitschrift*, 20, 397–407.
- El Hachem, A. et al. (2022). Space-time statistical quality control of extreme precipitation observations. *Hydrology and Earth System Sciences*, 26, 6137–6146.
- Lewis, E. et al. (2021). Quality control of a global hourly rainfall dataset. *Environmental Modelling and Software*, 144, 105169.
- Stepanek, P., Zahradnicek, P. and Skalak, P. (2009). Data quality control and homogenization of air temperature and precipitation series in the area of the Czech Republic in the period 1961–2007. *Advances in Science and Research*, 3, 23–26.
- Vicente-Serrano, S. M., Begueria, S., Lopez-Moreno, J. I., Garcia-Vera, M. A. and Stepanek, P. (2010). A complete daily precipitation database for northeast Spain: reconstruction, quality control, and homogeneity. *International Journal of Climatology*, 30, 1146–1163.
- Estevez, J., Llabres-Brustenga, A., Casas-Castillo, M. C., Garcia-Marin, A. P., Kirchner, R. and Rodriguez-Sola, R. (2022). A quality control procedure for long-term series of daily precipitation data in a semiarid environment. *Theoretical and Applied Climatology*, 149, 1029–1041.
- Golkhatmi, N. S. N. and Farzandi, M. (2024). Enhancing rainfall data consistency and completeness: a spatiotemporal quality control approach and missing data reconstruction using MICE on large precipitation datasets. *Water Resources Management*, 38, 815–833.
- Serrano-Notivol, R. and Tejedor, E. (2021). From rain to data: a review of the creation of monthly and daily station-based gridded precipitation datasets. *WIREs Water*, 8, e1555.

See Also

[precipQC](#), [precipQC_subdaily](#), [precipQC_range](#), [precipQC_weekday](#), [precipQC_spatial](#), [plot.precipQC](#), [zoo](#)

Examples

```
set.seed(1)
dates <- seq(as.Date("2018-01-01"), by="day", length.out=400)
values <- matrix(stats::rgamma(length(dates) * 3, 0.4, 1),
                 ncol=3)
values[values < 1] <- 0
values[20, 1] <- -1
colnames(values) <- c("A", "B", "C")
pcp <- zoo(values, dates)

meta <- data.frame(code=colnames(pcp),
                  longitude=c(-71.2, -71.1, -71.0),
                  latitude=c(-33.2, -33.1, -33.0),
                  altitude_m=c(100, 250, 400))

qc <- precipQC_daily(pcp, metadata=meta, station.id="code",
                   coords=c("longitude", "latitude"),
                   elevation="altitude_m", min.years=0,
                   checks=c(breakpoint=FALSE))

qc$station.summary
head(qc$suspicious)
plot(qc)
```

```
precipQC_subdaily
```

```
Quality control of sub-daily precipitation time series
```

Description

Applies physical, climatological, temporal, accumulation, spatial, dry-spell, and homogeneity checks to one or more sub-daily precipitation series in a zoo object. It returns auditable point flags, optional corrections, and station acceptance recommendations without writing files.

Usage

```
precipQC_subdaily(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  checks=c(range=TRUE, climatology=TRUE, persistence=TRUE,
           daily.accumulation=TRUE, monthly.accumulation=TRUE,
           weekday=TRUE, spatial=TRUE, dryspell=TRUE, breakpoint=TRUE),
  lower=0, max.rate=401, wet.threshold=0.1,
  climatology.prob=0.999, climatology.z=8,
  climatology.min.samples=100L,
  persistence.threshold=NULL, persistence.high.run=2L,
  persistence.long.hours=24,
  accumulation.factor=2,
  weekday.min.wet=20L, weekday.alpha=0.001,
  weekday.ratio=0.5, weekday.min.coverage=0.9,
  spatial.hours=c(0, 1, 3, 6, 24), spatial.cr=3,
```

```

n.neighbours=10L, max.distance=50, min.neighbours=2L,
min.overlap=100L, min.correlation=0,
dryspell.days=15L, neighbour.wet.days=3L,
neighbour.fraction=1,
correction=c("none", "set_na", "spatial"),
min.evidence=2L, max.missing=0.2, max.suspicious=0.05,
min.years=1, discard.breakpoint=FALSE,
elevation=NULL, elevation.scale=500, verbose=TRUE
)

```

Arguments

- x** A zoo object with a strictly increasing POSIXt index and a modal interval shorter than one day. Each column is a station and must have a unique name. Values are assumed to be precipitation depths in mm for each interval. Gaps are allowed when their durations are integer multiples of the modal interval.
- metadata, station.id, coords, elevation, elevation.scale** Station metadata and its identifier/coordinate column names. See [precipQC_daily](#). Metadata are optional. When supplied, identifier, longitude, and latitude field names are required; elevation in metres is optional and is activated by naming its field. Individual longitude or latitude entries may be NA; see the fallback rules in [precipQC_daily](#).
- checks** Named logical vector. Any subset of "range", "climatology", "persistence", "daily.accumulation", "monthly.accumulation", "weekday", "spatial", "dryspell", and "breakpoint" can override the default of running every check.
- lower** Physical lower limit in mm per interval.
- max.rate** Upper screening rate in mm/hour. It is multiplied by the detected interval length, so the range test uses a depth appropriate to the native time step. The default is the conservative world-record hourly value used by Lewis et al. (2021), not a local design-rainfall estimate.
- wet.threshold** Depth in mm separating dry and wet intervals.
- climatology.prob, climatology.z, climatology.min.samples** Empirical probability, robust transformed-scale multiplier, and minimum sample size used to estimate month-specific upper climatological limits.
- persistence.threshold** Minimum high value to test for short repeated runs. If NULL, twice the station mean wet-day depth is used.
- persistence.high.run** Minimum number of consecutive identical high values.
- persistence.long.hours** Duration at which any repeated wet value is suspicious. It is converted to a number of native intervals.
- accumulation.factor** Multiplier of the station mean wet-day depth used to screen possible daily and monthly totals recorded in one sub-daily interval.

<code>weekday.min.wet</code> , <code>weekday.alpha</code> , <code>weekday.ratio</code>	Minimum annual wet-day count, significance level, and maximum observed-to-expected wet-day ratio for the weekday false-zero test. These have the same interpretation as the corresponding arguments of precipQC_daily .
<code>weekday.min.coverage</code>	Minimum fraction of expected native intervals that must be observed before a sub-daily calendar day is included in the weekday test. It must be in $(0, 1]$.
<code>spatial.hours</code>	Accumulation durations in hours for spatial tests. Zero denotes the native interval. Positive values must be integer multiples of the detected interval. Flags on accumulated windows are expanded to their contributing native observations.
<code>spatial.cr</code>	Critical ratio for the transformed observation-neighbour residual.
<code>n.neighbours</code> , <code>max.distance</code> , <code>min.neighbours</code> , <code>min.overlap</code> , <code>min.correlation</code>	Controls for selecting up to <code>n.neighbours</code> , normally within <code>max.distance</code> km, and for calculating spatial estimates. See precipQC_daily .
<code>dryspell.days</code> , <code>neighbour.wet.days</code> , <code>neighbour.fraction</code>	Controls for flagging target-station dry windows that selected neighbours do not corroborate.
<code>correction</code>	"none" preserves all data; "set_na" replaces confirmed suspicious observations by NA; "spatial" uses native-scale spatial estimates where available and NA otherwise. No correction is made by default.
<code>min.evidence</code>	Number of coincident non-hard flags required to reject an observation. Physical range, persistence, accumulation, and corroborated dry-spell flags are hard evidence. The weekday test is non-hard evidence and therefore remains for review unless another active test corroborates it.
<code>max.missing</code> , <code>max.suspicious</code> , <code>min.years</code>	Maximum missing fraction, maximum confirmed-suspicious fraction, and minimum record length used for station acceptance recommendations.
<code>discard.breakpoint</code>	Logical. Whether a large significant homogeneity breakpoint should by itself cause a discard recommendation. The conservative default is FALSE.
<code>verbose</code>	Logical. If TRUE, informative messages identify the workflow start, each enabled test and spatial scale, evidence combination, and the final station/observation summary. If FALSE, only these informational messages are suppressed; warnings and errors remain visible.

Details

The sub-daily workflow follows a layered rule base:

1. *Range and temporal-resolution validation.* Negative depths and depths exceeding a duration-adjusted screening record are flagged. The time index is checked for strict ordering and a consistent modal interval. This addresses impossible values, duplicated timestamps, and mixed frequencies discussed by Fileni et al. (2023) and Villalobos-Herrera et al. (2022), and implements the gross-error and range stages of Jurczyk et al. (2020).

2. *Month-specific climatological plausibility.* A $\log_{10}$ -scale empirical/robust upper limit is estimated iteratively for each calendar month. The transformation reduces the strong positive skew of sub-daily rainfall, following the motivation for Box–Cox transformation and month-specific plausibility limits in El Hachem et al. (2022).
3. *Repeated values.* Two or more identical values above a high threshold, or any identical wet value repeated for at least 24 hours, are flagged. These rules adapt the streak checks of Lewis et al. (2021) and the blocked-sensor consistency check of Jurczyk et al. (2020).
4. *Daily and monthly accumulations.* A large isolated interval preceded and followed by dry intervals is tested as a possible longer-period total placed at one timestamp. The daily test uses 23-hour flanks; the monthly test uses a preceding 28-day dry period and a 23-hour following flank. Blenkinsop et al. (2017) and Lewis et al. (2021) show why surrounding dryness is important for avoiding false flags.
5. *Systematic weekday false zeros.* Native observations are aggregated in memory to daily totals only when at least `weekday.min.coverage` of the expected intervals is present. The daily wet-day occurrence test of `precipQC_weekday` is then applied, and suspicious dry days are expanded back to their finite dry native intervals. This adapts the weekday-zero quality component of Estevez et al. (2022) without treating partially observed days as zero-rainfall days.
6. *Multi-scale spatial consistency.* Leave-one-station-out robust neighbour estimates and critical ratios are calculated at the native, 1-, 3-, 6-, and 24-hour scales by default. Multi-scale testing follows El Hachem et al. (2022), who showed that an error may be visible at one accumulation but hidden at another; station-neighbour corroboration also follows the spatial consistency stage of Jurczyk et al. (2020). When elevation is supplied, neighbour ranking and inverse-distance weights also favour stations with similar height. Missing target coordinates trigger correlation-based selection; a located target uses unlocated neighbours only after eligible located neighbours.
7. *Spatial dry-spell consistency.* Fifteen-day dry windows are flagged only when neighbours record at least three wet days, adapting the GSDR-QC rule described by Lewis et al. (2021) and the regional suspect-zero logic of Golkhatmi and Farzandi (2024).
8. *Homogeneity.* Sub-daily values are first aggregated to daily totals. Pettitt tests are then applied to annual totals, wet-day counts, maxima, and extreme-day counts, with Holm-adjusted p-values. This is a diagnostic rather than a default discard rule, consistent with the multi-test caution of Stepanek et al. (2009).

Sub-hourly tip-time tests cannot be reconstructed from already accumulated zoo values. In particular, the inter-tip statistic and 1-minute thresholds of Villalobos-Herrera et al. (2022) require raw tip timestamps. The present function instead checks the finest interval actually supplied and then aggregates it to multiple durations.

Spatial disagreement is evidence for review, not proof of error, because short-duration convective rainfall can be highly localized. By default an observation is rejected only by a hard test or by agreement of at least two non-hard tests.

Value

An object of class "precipQC". It contains accepted and discarded metadata data.frames, unmodified and optionally corrected zoo series for accepted stations, a long-form suspicious table, a corrections audit table, logical flag series for every active test, station diagnostics, spatial diagnostics, neighbours, and settings. See `precipQC-class` for the complete structure.

Author(s)

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

References

- Blenkinsop, S., Lewis, E., Chan, S. C. and Fowler, H. J. (2017). Quality-control of an hourly rainfall dataset and climatology of extremes for the UK. *International Journal of Climatology*, 37, 722–740.
- Lewis, E. et al. (2021). Quality control of a global hourly rainfall dataset. *Environmental Modelling and Software*, 144, 105169.
- El Hachem, A. et al. (2022). Space-time statistical quality control of extreme precipitation observations. *Hydrology and Earth System Sciences*, 26, 6137–6146.
- Villalobos-Herrera, R. et al. (2022). Sub-hourly resolution quality control of rain-gauge data significantly improves regional sub-daily return level estimates. *Quarterly Journal of the Royal Meteorological Society*, 148, 3252–3271.
- Pritchard, D. et al. (2023). An observation-based dataset of global sub-daily precipitation indices (GSDR-I). *Scientific Data*, 10, 393.
- Fileni, F., Fowler, H. J., Lewis, E., McLay, F. and Yang, L. (2023). A quality-control framework for sub-daily flow and level data for hydrological modelling in Great Britain. *Hydrology Research*, 54, 1357–1367.
- Jurczyk, A., Szturc, J., Otop, I., Osrodka, K. and Struzik, P. (2020). Quality-based combination of multi-source precipitation data. *Remote Sensing*, 12, 1709.
- Stepanek, P., Zahradnicek, P. and Skalak, P. (2009). Data quality control and homogenization of air temperature and precipitation series in the area of the Czech Republic in the period 1961–2007. *Advances in Science and Research*, 3, 23–26.
- Golkhatmi, N. S. N. and Farzandi, M. (2024). Enhancing rainfall data consistency and completeness: a spatiotemporal quality control approach and missing data reconstruction using MICE on large precipitation datasets. *Water Resources Management*, 38, 815–833.
- Estevez, J., Llabres-Brustenga, A., Casas-Castillo, M. C., Garcia-Marin, A. P., Kirchner, R. and Rodriguez-Sola, R. (2022). A quality control procedure for long-term series of daily precipitation data in a semiarid environment. *Theoretical and Applied Climatology*, 149, 1029–1041.

See Also

[precipQC](#), [precipQC_daily](#), [precipQC_persistence](#), [precipQC_accumulation](#), [precipQC_weekday](#), [precipQC_spatial](#), [plot.precipQC](#), [subhourly2nhourly](#)

Examples

```
set.seed(2)
times <- seq(as.POSIXct("2020-01-01 00:00:00", tz="UTC"),
             by="hour", length.out=24 * 40)
values <- matrix(stats::rexp(length(times) * 3), ncol=3)
values[values < 2] <- 0
values[10, 1] <- -1
colnames(values) <- c("A", "B", "C")
pcp <- zoo(values, times)
```

```

meta <- data.frame(station=colnames(pcp),
                  lon=c(-71.2, -71.1, -71.0),
                  lat=c(-33.2, -33.1, -33.0))

qc <- precipQC_subdaily(
  pcp, metadata=meta, min.years=0,
  spatial.hours=c(0, 1, 3),
  checks=c(monthly.accumulation=FALSE, breakpoint=FALSE)
)
qc$station.summary
head(qc$suspicious)
plot(qc)

```

precipQC_tests	<i>Individual precipitation quality-control tests</i>
----------------	---

Description

Individual tests used by [precipQC_daily](#) and [precipQC_subdaily](#). They are exported so that each source of QC evidence can be inspected or tuned independently.

Usage

```

precipQC_range(x, lower=0, upper=Inf)

precipQC_persistence(
  x, high.threshold=10, high.run=5L,
  long.run=Inf, wet.threshold=0.1
)

precipQC_duplicate(
  x, min.month.values=20L, min.year.values=300L,
  min.nonzero=3L, wet.threshold=0.1
)

precipQC_frequency(
  x, window=10L, counts=c(9L, 8L, 7L, 5L),
  probs=c(0.3, 0.5, 0.7, 0.9),
  min.samples=20L, wet.threshold=0.1
)

precipQC_gap(x, gap=300, min.samples=30L, wet.threshold=0.1)

precipQC_weekday(
  x, wet.threshold=0.1, min.wet.days=20L,
  alpha=0.001, underreporting.ratio=0.5
)

```

```

precipQC_climatology(
  x, group=c("dayofyear", "month"), window=15L,
  prob=0.999, z=8, min.samples=30L, wet.threshold=0.1
)

precipQC_accumulation(
  x, interval.hours, preceding.hours=23, following.hours=23,
  threshold.factor=2, wet.threshold=0.1, missing.only=FALSE
)

precipQC_spatial(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  n.neighbours=10L, max.distance=Inf, min.neighbours=2L,
  min.overlap=30L, min.correlation=0, cr.threshold=3,
  wet.threshold=0.1, elevation=NULL, elevation.scale=500
)

precipQC_dryspell(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  n.neighbours=10L, max.distance=Inf, window.days=15L,
  neighbour.wet.days=3L, neighbour.fraction=1,
  wet.threshold=0.1, elevation=NULL, elevation.scale=500
)

precipQC_breakpoint(
  x, wet.threshold=0.1, min.years=5L, alpha=0.05,
  min.relative.change=0.5, min.completeness=0.8,
  indicators=c("total", "wet.days", "maximum", "extreme.days")
)

```

Arguments

<code>x</code>	A numeric zoo object. Each column is one station. Spatial and dry-spell tests require at least two neighbours for the default spatial calculation and at least one neighbour for the dry-spell calculation.
<code>lower, upper</code>	Physical lower and upper precipitation limits. Each may be scalar or contain one value per station.
<code>high.threshold</code>	Scalar or station-specific lower bound for repeated high values.
<code>high.run</code>	Minimum number of consecutive identical high values.
<code>long.run</code>	Minimum run length for any repeated wet value, or Inf to disable the long-run component.
<code>min.month.values, min.year.values</code>	Minimum number of complete corresponding calendar positions for copied-month and copied-year comparisons.
<code>min.nonzero</code>	Minimum non-zero values required in both blocks before precipitation duplication is flagged.

counts, probs	Paired repeated-value counts and minimum climatological probabilities for the clustered frequent-value rules.
gap	Minimum separation in mm between adjacent sorted wet values in one calendar month distribution.
min.wet.days	Minimum annual number of wet days required for the weekday test.
underreporting.ratio	Maximum observed-to-expected wet-day ratio used to identify an under-recorded weekday after the overall weekday occurrence test is significant.
wet.threshold	Precipitation depth that separates dry and wet observations.
group	Climatological grouping: a moving calendar-day window or calendar months.
window	Calendar-day window width. It is ignored for monthly grouping.
prob, z, min.samples	Empirical upper probability, robust standard-deviation multiplier on the log1p scale, and required reference sample size. For frequent-value and gap checks only min.samples is used.
interval.hours	Native time-step length in hours for the accumulation test.
preceding.hours, following.hours	Durations on either side of a possible accumulated total that must be dry or missing.
threshold.factor	Multiplier of station mean wet-day precipitation used as the lower screen for an accumulated total.
missing.only	Logical. If TRUE, only preceding missing values or an index gap satisfy the accumulation test; if FALSE, preceding dry values are required.
metadata	NULL, or a data.frame with one row per station. When supplied, the selected identifier, longitude, and latitude fields are required.
station.id	Name of the metadata station-identifier column.
coords	Names of numeric longitude and latitude columns in decimal degrees, in that order. Individual entries may be NA; non-missing values are range-checked.
elevation	NULL, or the name of an optional numeric elevation column in metres.
elevation.scale	Positive elevation-decay scale in metres. Height similarity is $\exp(-\text{abs}(\text{dz}) / \text{elevation.scale})$ and influences neighbour ranking and spatial weights. It is ignored when elevation=NULL.
n.neighbours	Maximum number of neighbours.
max.distance	Maximum great-circle distance in km when coordinates are available. Pairs for which either station lacks a coordinate are not distance-filtered.
min.neighbours	Minimum simultaneous valid neighbours for a spatial estimate.
min.overlap	Minimum paired observations for neighbour selection.
min.correlation	Preferred minimum Spearman correlation between target and neighbour.
cr.threshold	Critical ratio for the absolute transformed residual divided by robust local and historical dispersion.

<code>window.days</code>	Length of target-station dry windows.
<code>neighbour.wet.days</code>	Minimum number of wet days during the same window at a neighbour.
<code>neighbour.fraction</code>	Fraction of selected neighbours that must meet <code>neighbour.wet.days</code> .
<code>min.years</code>	Minimum number of annual wet-period medians for the Pettitt test.
<code>alpha</code>	Significance level for the weekday test or the Holm-adjusted Pettitt p-value.
<code>min.relative.change</code>	Minimum absolute relative difference between pre- and post-break medians.
<code>min.completeness</code>	Minimum fraction of expected daily observations required to retain an annual indicator in breakpoint testing.
<code>indicators</code>	Unique subset of "total", "wet.days", "maximum", and "extreme.days" to test for breakpoints.

Details

`precipQC_range` flags values outside physical limits.

`precipQC_persistence` run-length encodes each series. Repeated high values and exceptionally long runs of any wet value are flagged. Exact repetition is intentional because this test targets copied, stuck, or artificially disaggregated records.

`precipQC_duplicate` compares corresponding positions in entire years, different months of the same year, and the same calendar month in different years. Both blocks must be complete over the comparison and contain at least `min.nonzero` wet values. This implements the precipitation duplicate checks of Durre et al. (2010).

`precipQC_frequency` scans consecutive non-zero reports, skipping dry and missing observations, for the tiered 9/10, 8/10, 7/10, and 5/10 identical-value rules of Durre et al. (2010). The repeated amount must also exceed its calendar-month climatological percentile, which reduces false positives from measurement resolution.

`precipQC_gap` sorts wet observations separately by calendar month. When an adjacent gap is at least `gap`, it flags the upper tail. The 300-mm default follows the daily precipitation distribution-gap test of Durre et al. (2010).

`precipQC_weekday` requires at most one observation per calendar day and compares station-year wet-day counts among weekdays, after adjusting the expected counts for the number of available observations on each weekday. Only when the overall chi-squared test is highly significant does it flag dry values on strongly under-recorded weekdays. It is intended to detect false zeros caused by systematic missed manual observations. `precipQC_subdaily` applies it after coverage-screened daily aggregation and expands the resulting flags back to native intervals.

`precipQC_climatology` works on `log1p` precipitation. For each calendar group, it takes the more conservative of an upper empirical quantile and a robust median-plus-MAD threshold. A largest value above the limit is temporarily removed and the threshold recomputed until stable. This protects the reference distribution from gross errors while retaining a low false positive rate.

`precipQC_accumulation` first calculates the mean wet-day total of each station. A candidate must exceed `threshold.factor` times that mean and have the requested dry/missing flanks. It identifies candidates only; it cannot prove that a reported depth is a longer-period accumulation.

`precipQC_spatial` applies a leave-one-station-out test after $\log_{1p}$ transformation. Coordinates, overlap, and correspondence select up to ten neighbours. If elevation is selected, its exponential similarity factor penalizes neighbours separated by large height differences. The estimate is a distance- and elevation-weighted median when usable metadata weights exist and an ordinary median otherwise. A located target ranks eligible located neighbours first and uses correlation-qualified unlocated neighbours only as fallbacks. A target missing either coordinate uses correlation-based selection and remains fully testable. The critical ratio uses the larger of the simultaneous neighbour dispersion, historical target residual dispersion, and a small numerical floor. Only positive residuals are flagged here; isolated dryness is handled separately.

`precipQC_dryspell` aggregates native observations to wet/dry calendar days, locates long dry runs, and flags target values only where the requested fraction of neighbours records enough wet days in the same window.

`precipQC_breakpoint` first aggregates sub-daily input to daily totals. It applies the rank-based Pettitt statistic separately to sufficiently complete annual precipitation totals, wet-day counts, maxima, and counts above the station wet-day 99th percentile. P-values are Holm-adjusted across the selected indicators. The strongest breakpoint, its adjusted p-value, pre/post relative change, and number of significant indicators are returned because a statistically significant change is not necessarily a measurement error.

Value

`precipQC_range`, `precipQC_persistence`, `precipQC_duplicate`, `precipQC_frequency`, `precipQC_gap`, `precipQC_weekday`, `precipQC_climatology`, `precipQC_accumulation`, and `precipQC_dryspell` return logical zoo objects matching `x`.

`precipQC_spatial` returns a list with logical flags, numeric scores, neighbour estimate series, and selected neighbours.

`precipQC_breakpoint` returns one data.frame row per station with the breakpoint year, selected indicator, adjusted p-value, relative change, number of significant indicators, and logical flag.

Author(s)

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

References

- Hamada, A., Arakawa, O. and Yatagai, A. (2011). An automated quality control method for daily rain-gauge data. *Global Environmental Research*, 15, 183–192.
- Scherrer, S. C. et al. (2011). Operational quality control of daily precipitation using spatio-climatological plausibility testing. *Meteorologische Zeitschrift*, 20, 397–407.
- El Hachem, A. et al. (2022). Space-time statistical quality control of extreme precipitation observations. *Hydrology and Earth System Sciences*, 26, 6137–6146.
- Durre, I., Menne, M. J., Gleason, B. E., Houston, T. G. and Vose, R. S. (2010). Comprehensive automated quality assurance of daily surface observations. *Journal of Applied Meteorology and Climatology*, 49, 1615–1633.
- Lewis, E. et al. (2021). Quality control of a global hourly rainfall dataset. *Environmental Modelling and Software*, 144, 105169.

Stepanek, P., Zahradnicek, P. and Skalak, P. (2009). Data quality control and homogenization of air temperature and precipitation series in the area of the Czech Republic in the period 1961–2007. *Advances in Science and Research*, 3, 23–26.

Vicente-Serrano, S. M., Begueria, S., Lopez-Moreno, J. I., Garcia-Vera, M. A. and Stepanek, P. (2010). A complete daily precipitation database for northeast Spain: reconstruction, quality control, and homogeneity. *International Journal of Climatology*, 30, 1146–1163.

Estevez, J. et al. (2022). A quality control procedure for long-term series of daily precipitation data in a semiarid environment. *Theoretical and Applied Climatology*, 149, 1029–1041.

Golkhatmi, N. S. N. and Farzandi, M. (2024). Enhancing rainfall data consistency and completeness: a spatiotemporal quality control approach and missing data reconstruction using MICE on large precipitation datasets. *Water Resources Management*, 38, 815–833.

See Also

```
precipQC_daily, precipQC_subdaily, plot.precipQC
```

Examples

```
dates <- seq(as.Date("2020-01-01"), by="day", length.out=60)
x <- zoo(cbind(A=rep(0, 60), B=rep(0, 60), C=rep(0, 60)), dates)
x[10, "A"] <- -1
x[20:24, "B"] <- 15

precipQC_range(x)
precipQC_persistence(x)
precipQC_weekday(x)
precipQC_breakpoint(x)
```

rm1stchar	<i>Remove First Character(s)</i>
-----------	----------------------------------

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	<i>Seasonal Function</i>
------------------	--------------------------

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 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 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

<code>ltime</code>	vector with the date/times of each element of a zoo object.
<code>lstart.month</code>	numeric in [1,...,12], representing the starting month to be used in the computation of annual values. By default <code>lstart.month=1</code> (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 <i>dates</i> , <i>from</i> and <i>to</i> . 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:

$$si = (1/R) * \sum_{i=1, i=12, \dots} (abs(xi - R/12))$$

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 occurs 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. $IQR(x) = \text{quantile}(x, 3/4) - \text{quantile}(x, 1/4)$
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	<i>Station Name -> Time Series</i>
----------	---------------------------------------

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

<code>date.fmt</code>	character indicating the format in which the dates are stored in <i>dates</i> , e.g. <code>%Y-%m-%d</code> . See format in as.Date . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
<code>var.type</code>	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
<code>tstep.out</code>	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
<code>FUN</code>	ONLY required when <code>var.type</code> is missing and <code>tstep</code> 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)
<code>na.rm</code>	a logical value indicating whether 'NA' values should be stripped before the computation proceeds.
<code>from</code>	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 <code>date.fmt</code> .
<code>to</code>	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 <code>date.fmt</code> .

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")
```

spi	<i>Standardized Precipitation and Precipitation-Evapotranspiration Indices</i>
-----	--

Description

Computes the Standardized Precipitation Index (SPI) or the Standardized Precipitation-Evapotranspiration Index (SPEI) for one or more complete monthly zoo series using base-R distribution and optimisation functions.

Usage

```
spi(x, scale,
    distribution=c("gamma", "gumbel", "logis", "llogis",
                  "lnorm", "norm", "weibull"),
    fit=c("max-lik", "ub-pwm", "pp-pwm"), zero.threshold=0,
    kernel=list(type="rectangular", shift=0), ref.start=NULL, ref.end=NULL,
    params=NULL, start.fun=NULL, start.fun.fix=FALSE,
    p0=TRUE, p0.center.mass=FALSE, scaling=c("sd", "no", "max"),
    sci.limit=Inf, na.rm=FALSE, out.type=c("zoo", "numeric"),
    verbose=FALSE, warn=TRUE, ...)

spei(x, scale, distribution=c("genlog", "gev", "norm", "pe3"),
     fit=c("max-lik", "ub-pwm", "pp-pwm"),
     kernel=list(type="rectangular", shift=0), ref.start=NULL, ref.end=NULL,
     params=NULL, start.fun=NULL, start.fun.fix=FALSE,
     p0=FALSE, p0.center.mass=FALSE, scaling=c("sd", "no", "max"),
     sci.limit=Inf, na.rm=FALSE, out.type=c("zoo", "numeric"),
     verbose=FALSE, warn=TRUE, ...)
```

Arguments

x	numeric zoo object containing one or more monthly precipitation series for spi, or climatic water-balance series (precipitation minus potential evapotranspiration) for spei. Its time index must inherit from Date, POSIXt, or yearmon, and it must contain one value for every consecutive month.
scale	positive integer indicating the number of months in the backward-looking accumulation period. This argument has no default and must be provided.
distribution	character string naming the distribution to fit. For spi, valid values are "gamma", "gumbel", "logis" (logistic), "llogis" (log-logistic), "lnorm" (lognormal), "norm", and "weibull". For spei, valid values are "genlog" (generalized logistic), "gev", "norm", and "pe3" (Pearson type III). These are the candidate distributions evaluated by Stagge et al. (2015). When omitted, the first value is used: "gamma" for SPI and "genlog" for SPEI.
fit	character string specifying the parameter-estimation method: "max-lik" for maximum likelihood, "ub-pwm" for unbiased probability-weighted moments,

	or "pp-pwm" for plotting-position probability-weighted moments. The default is "max-lik".
zero.threshold	non-negative numeric value. For spi, precipitation values strictly below this threshold are changed to zero before temporal accumulation and distribution fitting. The default zero leaves all positive values unchanged.
kernel	list with elements type and shift defining the backward-looking accumulation kernel. Valid kernel types are "rectangular", "triangular", "circular", and "gaussian". shift must be an integer between zero and scale - 1. The default is an unshifted rectangular kernel.
ref.start, ref.end	optional Date objects or character strings in "YYYY-MM" or "YYYY-MM-DD" format defining the reference period used to estimate the distribution parameters. The day component, when supplied, is used only to identify the calendar month. By default, the full period in x is used.
params	optional numeric distribution parameters that override parameter fitting. A vector supplies one parameter set for every month and series; a matrix with dimensions npar by 12 supplies monthly parameters that are reused for every series; and an array with dimensions npar by nseries by 12 supplies parameters for every month and series, as in SPEI . Parameter names may be omitted when the documented order is used. The default NULL estimates parameters from x.
start.fun	optional function used to calculate initial parameters for maximum-likelihood estimation, following the interface used by SCI . It is called as start.fun(x, distr) for each calendar month and series and must return a numeric parameter vector. The default NULL uses hydroTSM's internal base-R starting-value estimator.
start.fun.fix	logical. If TRUE and maximum-likelihood optimisation fails, the initial parameters returned by start.fun (or by the internal estimator) are retained. If FALSE, the default, the affected month's parameters and drought-index values are set to NA.
p0	logical. If TRUE, the probability of zero values is represented by a mixed distribution. This is the default for SPI and is disabled by default for SPEI.
p0.center.mass	logical. If TRUE, the probability assigned to zero precipitation is estimated with the centre-of-probability-mass estimator described by Stagge et al. (2015).
scaling	character string indicating whether the accumulated precipitation is divided by its standard deviation ("sd", the default), left unchanged ("no"), or divided by its maximum ("max") before parameter estimation. Scaling can stabilise maximum-likelihood estimation.
sci.limit	non-negative numeric value used to truncate the absolute SPI or SPEI values. The default Inf disables truncation.
na.rm	logical. If FALSE, the default, x must not contain missing values. If TRUE, missing values are omitted during parameter fitting; accumulation windows containing a missing value remain NA.
out.type	character string indicating whether the result is returned as a "zoo" object (the default) or as a numeric vector or matrix.

verbose	logical indicating whether informative, bracketed progress messages are emitted with <code>message</code> . The default FALSE keeps package calls quiet.
warn	logical indicating whether warnings should be issued when parameter estimation or transformation fails for a calendar month.
...	additional maximum-likelihood fitting controls, notably <code>mledist.par</code> .

Details

The functions form a backward-looking accumulation at the selected scale, fit a separate distribution for each calendar month over the reference period, evaluate the fitted cumulative probabilities, and transform them to standard-normal variates. The SPI functions default to a Gamma distribution and a mixed probability at zero. The SPEI functions default to the generalized logistic distribution and retain negative climatic-water-balance values.

The seven SPI distributions and four SPEI distributions listed under `distribution` follow Section 2.4 and Appendices B and C of Stagge et al. (2015). Their density, cumulative-probability, L-moment conversion, and maximum-likelihood calculations use only base R.

For `fit="ub-pwm"`, the first three probability-weighted moments use the unbiased order-statistic estimators. For `fit="pp-pwm"`, non-exceedance probabilities are $F_i = (i - 0.35)/n$, corresponding to the plotting-position constants $A = -0.35$ and $B = 0$ used by **SPEI**. At least four non-missing, non-constant calibration values are required for either PWM method.

Accumulation is performed before the reference period is selected. Therefore, an accumulated value at the start of the reference period can use preceding observations from `x`. Kernel values are normalised and multiplied by `scale`, following the magnitude-preserving convention used by the **SPEI** package.

For SPI, `zero.threshold` is applied to individual monthly precipitation values before accumulation. At scales larger than one, the zero probability is therefore estimated from accumulated periods whose threshold-adjusted total is zero.

Maximum-likelihood estimation uses `optim` and unbiased-PWM starting values. Generalized-logistic, log-logistic, Pearson type III, GEV, and Gumbel distribution functions are implemented internally. Neither **SCI**, **SPEI**, **lmomco**, nor **evd** is required.

When `params` is supplied, parameter fitting is skipped and the supplied parameters are applied to the unscaled accumulated series; consequently, `fit`, `scaling`, `start.fun`, and `start.fun.fix` do not affect the result. When `p0=TRUE`, the zero probability is still estimated separately for each calendar month from the reference data. The parameter orders are: `gamma=(shape, rate)`, `gumbel=(loc, scale)`, `logis=(location, scale)`, `llogis=(shape, scale)`, `lnorm=(meanlog, sdlog)`, `norm=(mean, sd)`, `weibull=(shape, scale)`, `genlog=(shape, scale, location)`, `gev=(loc, scale, shape)`, and `pe3=(shape, scale, location)`.

`start.fun` is used only with `fit="max-lik"`. Its result can be named, in which case the names must match those listed above, or unnamed in the documented order. Retaining initial values with `start.fun.fix=TRUE` can produce a complete index when optimisation fails, but the retained values are starting estimates rather than a converged maximum-likelihood fit.

Value

A zoo object by default, or a numeric vector or matrix when `out.type="numeric"`. The result has the same time index, length, and column names as `x`. The first `scale - 1` values are NA when `scale > 1`.

Author(s)

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

References

McKee, T. B., Doesken, N. J., and Kleist, J. (1993). The relationship of drought frequency and duration to time scales. *Proceedings of the 8th Conference on Applied Climatology*, 17–22 January, Anaheim, California, 179–184.

Stagge, J. H., Tallaksen, L. M., Gudmundsson, L., Van Loon, A. F., and Stahl, K. (2015). Candidate distributions for climatological drought indices (SPI and SPEI). *International Journal of Climatology*, 35, 4027–4040. doi:10.1002/joc.4267.

See Also

[sfreq](#)

Examples

```
data(EbroPPtsMonthly)

pcp <- zoo(EbroPPtsMonthly$P9001, EbroPPtsMonthly$Date)
spi3 <- spi(pcp, scale=3, zero.threshold=0.1, warn=FALSE)
head(spi3)

spi3.pwm <- spi(pcp, scale=3, fit="ub-pwm", warn=FALSE)
head(spi3.pwm)

## Artificial monthly climatic water balance for illustrating 'spei'
pet <- zoo(60 + 40*sin(2*pi*(seq_along(pcp)-1)/12), time(pcp))
spei3 <- spei(pcp - pet, scale=3, warn=FALSE)
head(spei3)

## A supplied vector is reused for all calendar months
spei.fixed <- spei(pcp - pet, scale=1, distribution="norm",
  params=c(mean=0, sd=50), warn=FALSE)
head(spei.fixed)
```

stdx

Standardization

Description

Standardizes 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]))
```

streamflowQC

Frequency-aware quality control of streamflow time series

Description

Detects the sampling frequency of a numeric zoo object and dispatches daily data to [streamflowQC_daily](#) or minute/hourly data to [streamflowQC_subdaily](#). No files are written.

Usage

```
streamflowQC(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"), ...,
  area=NULL, elevation=NULL, elevation.scale=500, verbose=TRUE
)
```

Arguments

<code>x</code>	A numeric zoo object with one streamflow station per column.
<code>metadata</code>	NULL, or a data.frame with one row per station.
<code>station.id</code>	Name of the station-identifier field in metadata.
<code>coords</code>	Names of longitude and latitude fields in metadata, in that order.
<code>...</code>	Arguments passed unchanged to the selected workflow.
<code>area</code>	NULL, or the name of a positive numeric catchment-area field. Units may be chosen by the user but must be consistent among stations.
<code>elevation</code>	NULL, or the name of a numeric station-elevation field.
<code>elevation.scale</code>	Positive elevation-decay scale used in spatial neighbour ranking.
<code>verbose</code>	Logical. If TRUE, informative messages report the detected frequency, enabled QC stages, evidence combination, and final QC summary. If FALSE, these messages are suppressed; warnings and errors remain visible.

Details

The wrapper changes only the resolution-specific workflow. Test activation, thresholds, station metadata, evidence requirements, correction policy, and station recommendations remain explicit user controls. Daily and sub-daily observations must be in consistent discharge units. Duplicate timestamps are rejected because selecting among contradictory measurements would require provenance or agency-quality information not contained in a numeric zoo object.

The design combines the point plausibility and homogeneity principles of Gudmundsson et al. (2018), the real-time hierarchy of U.S. IOOS (2018), and the sub-daily artefact and event-corroboration framework of Fileni et al. (2026).

Progress is emitted with `message()`, so it can be captured or suppressed without redirecting the returned object.

Value

An object of class "streamflowQC"; see [streamflowQC-class](#).

References

- Fileni, F. et al. (2026). UK-Flow15-QC: A quality control framework for better river flow data in hydrological research. *EGUsphere preprint*. doi:10.5194/egusphere-2026-277.
- Gudmundsson, L., Do, H. X., Leonard, M. and Westra, S. (2018). The Global Streamflow Indices and Metadata Archive (GSIM) - Part 2: quality control, time-series indices and homogeneity assessment. *Earth System Science Data*, 10, 787–804. doi:10.5194/essd-10-787-2018.
- U.S. Integrated Ocean Observing System (2018). *Manual for Real-Time Quality Control of Stream Flow Data Version 1.0*. doi:10.25923/gszc-ha43.

See Also

[streamflowQC_daily](#), [streamflowQC_subdaily](#), [streamflowQC_tests](#), [precipQC](#), [tempQC](#)

Examples

```

dates <- as.Date("2020-01-01") + 0:29
x <- zoo::zoo(cbind(A=10 + sin(seq_along(dates)),
                    B=12 + sin(seq_along(dates))), dates)
qc <- streamflowQC(
  x, checks=c(range=TRUE, duplicate=FALSE, gap=FALSE,
               climatology=FALSE, flatline=FALSE, spike=FALSE,
               rate=FALSE, highflow=FALSE, spatial=FALSE,
               breakpoint=FALSE),
  min.years=0, max.missing=1, max.review=1, max.suspicious=1
)
qc$station.summary

```

streamflowQC-class	<i>Streamflow quality-control result</i>
--------------------	--

Description

Structure, print method, and four-panel summary plot for objects returned by [streamflowQC_daily](#), [streamflowQC_subdaily](#), and [streamflowQC](#).

Usage

```

## S3 method for class 'streamflowQC'
print(x, ...)

## S3 method for class 'streamflowQC'
plot(
  x, max.stations=20L,
  col=c("#2878B5", "#D55E00", "#E69F00"), ...
)

```

Arguments

<code>x</code>	An object inheriting from class "streamflowQC".
<code>max.stations</code>	Maximum number of stations in the affected-station panel.
<code>col</code>	At least three plotting colours.
<code>...</code>	Additional arguments; the plot method passes them to its final time-series panel.

Details

The plot summarizes station recommendations, missing/review/rejected fractions, counts by active test, and the time distribution of confirmed suspicious observations. Plotting is diagnostic and does not alter the result or write a file unless the caller explicitly opens a file device.

Value

Both methods return `x` invisibly. A "streamflowQC" object is a list with:

- `accepted.metadata` and `discarded.metadata`: original station metadata plus completeness, evidence, homogeneity, recommendation, and reason fields;
- `accepted.data` and `discarded.data`: original zoo series split by the station recommendation;
- `accepted.corrected` and `discarded.corrected`: the same partitions after the requested correction policy;
- `suspicious`: one row per point flagged for review or rejection, including all contributing test names and any spatial estimate;
- `corrections`: an audit table containing every changed value;
- `flags`, `flag.count`, and `rejected`: individual logical flag objects, evidence counts, and confirmed decisions;
- `station.summary`: station completeness, review/rejected fractions, breakpoint diagnostics, recommendation, and reason;
- `breakpoint`: annual-indicator Pettitt-test diagnostics;
- `spatial.estimate`, `spatial.score`, `spatial.support`, and `neighbours`: spatial-test diagnostics;
- `highflow` and `return.period`: component high-flow flags, GEV parameters, and estimated return periods; and
- `settings`: all resolved workflow controls.

References

Fileni, F. et al. (2026). UK-Flow15-QC: A quality control framework for better river flow data in hydrological research. *EGUsphere preprint*. doi:10.5194/egusphere-2026-277.

See Also

[streamflowQC_daily](#), [streamflowQC_subdaily](#), [streamflowQC_tests](#)

streamflowQC_daily	<i>Quality control of daily streamflow time series</i>
--------------------	--

Description

Applies individually selectable physical, temporal, high-flow, spatial, and homogeneity tests to one or more daily streamflow series. It returns point flags and station acceptance recommendations without writing files.

Usage

```
streamflowQC_daily(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  checks=c(range=TRUE, duplicate=TRUE, gap=TRUE, climatology=TRUE,
    flatline=TRUE, spike=TRUE, rate=TRUE, highflow=TRUE,
    spatial=TRUE, breakpoint=TRUE),
  lower=0, upper=Inf,
  duplicate.min.month=20L, duplicate.min.year=300L,
  duplicate.min.nonzero=1L,
  climatology.window=5L, climatology.z=6,
  climatology.min.samples=30L, log.offset=0.01,
  flatline.run=11L, flatline.tolerance=0, flag.zero=FALSE,
  spike.factor=5, spike.prob=0.99,
  rate.z=6, rate.min.samples=30L,
  rate.max.rise=Inf, rate.max.fall=Inf, rate.drop.factor=5,
  highflow.log.z=6, highflow.amax.ratio=2,
  highflow.return.period=1000, highflow.min.years=10L,
  spatial.window.days=3L, spatial.high.prob=0.99,
  spatial.support.prob=0.95, spatial.f=3,
  n.neighbours=5L, max.distance=400, min.neighbours=2L,
  min.support=1L, min.overlap=90L, min.correlation=0.5,
  breakpoint.min.years=10L, breakpoint.alpha=0.01,
  breakpoint.min.relative.change=0.5,
  breakpoint.min.completeness=0.8,
  correction=c("none", "set_na", "spatial"),
  min.evidence=3L, max.missing=0.2, max.review=0.2,
  max.suspicious=0.05, min.years=1, discard.breakpoint=FALSE,
  area=NULL, elevation=NULL, elevation.scale=500, verbose=TRUE
)
```

Arguments

<code>x</code>	A numeric daily zoo object. Each column is one station and the index is Date or POSIXt.
<code>metadata</code>	NULL, or a data.frame containing every station in <code>x</code> . Extra fields are preserved in the output.
<code>station.id</code>	Name of the unique station-identifier field in <code>metadata</code> . Identifiers are matched to <code>colnames(x)</code> .
<code>coords</code>	Names of numeric longitude and latitude fields, in that order. Missing coordinates are allowed; non-missing coordinates are range-checked.
<code>checks</code>	Named logical vector. Omitted names retain their defaults; all daily tests are enabled by default.
<code>lower, upper</code>	Physical limits, scalar or station-specific. The default rejects negative flow but deliberately imposes no universal upper limit.
<code>duplicate.min.month, duplicate.min.year</code>	Minimum paired calendar positions for exact copied-month and copied-year checks.

<code>duplicate.min.nonzero</code>	Minimum positive values required in each candidate copied block.
<code>climatology.window</code>	Width in calendar days of the circular seasonal reference window.
<code>climatology.z</code>	Standard-deviation multiplier for the log-flow climatology test.
<code>climatology.min.samples</code>	Minimum seasonal reference observations.
<code>log.offset</code>	Positive constant added before taking logarithms.
<code>flatline.run</code>	Minimum number of consecutive equal or near-equal daily values. The default flags runs of more than 10 days.
<code>flatline.tolerance</code>	Maximum consecutive numerical difference within a flat line.
<code>flag.zero</code>	Whether repeated zero-flow runs may be flagged. The default protects legitimate no-flow periods.
<code>spike.factor</code>	Multiplicative isolated peak/dip threshold.
<code>spike.prob</code>	Station quantile used as the absolute isolated-spike threshold.
<code>rate.z</code>	Robust standardized first-difference threshold.
<code>rate.min.samples</code>	Minimum monthly first-difference reference sample.
<code>rate.max.rise, rate.max.fall</code>	Optional absolute consecutive rise and fall limits, scalar or station-specific.
<code>rate.drop.factor</code>	Multiplicative abrupt-fall threshold.
<code>highflow.log.z</code>	Standard-deviation multiplier for unusually high global log-flow.
<code>highflow.amax.ratio</code>	Ratio of the largest annual maximum to the second largest that generates a high-flow flag.
<code>highflow.return.period</code>	GEV return-period threshold in years.
<code>highflow.min.years</code>	Minimum annual maxima required for GEV fitting.
<code>spatial.window.days</code>	Number of days before and after a high-flow event searched for neighbouring support.
<code>spatial.high.prob, spatial.support.prob</code>	Target and neighbour high-flow quantiles.
<code>spatial.f</code>	Minimum standardized spatial-regression residual.
<code>n.neighbours</code>	Maximum selected neighbours per station.
<code>max.distance</code>	Maximum target-neighbour distance in km.
<code>min.neighbours</code>	Minimum simultaneous spatial predictions.
<code>min.support</code>	Minimum neighbouring high-flow events needed to corroborate a target event.
<code>min.overlap</code>	Minimum paired observations for neighbour selection and pairwise regression.

min.correlation	Minimum Spearman correlation on log-flow for a preferred neighbour.
breakpoint.min.years	Minimum complete annual indicators for a Pettitt test.
breakpoint.alpha	Holm-adjusted breakpoint significance level.
breakpoint.min.relative.change	Minimum relative pre/post median change.
breakpoint.min.completeness	Minimum daily completeness for an annual indicator.
correction	"none" preserves all values, "set_na" removes confirmed points, and "spatial" uses an available spatial estimate and otherwise NA.
min.evidence	Number of independent flag families needed to confirm a point that did not fail a hard test.
max.missing, max.review, max.suspicious	Maximum station fractions of missing expected values, any flagged observations, and confirmed suspicious observations.
min.years	Minimum record duration for station acceptance.
discard.breakpoint	Whether a significant large breakpoint discards a station. It is diagnostic by default because real catchment change or regulation can cause inhomogeneity.
area	NULL, or the name of a positive numeric catchment-area field used to compare specific discharge spatially.
elevation	NULL, or the name of an optional numeric elevation field.
elevation.scale	Positive elevation-decay scale used in neighbour ranking.
verbose	Logical. If TRUE, informative messages identify the workflow start, each enabled QC stage, evidence combination, and the final station/observation summary. If FALSE, only these informational messages are suppressed; warnings and errors remain visible.

Details

The daily climatology test reproduces the GSIM structure: it compares $\log(Q + \log.offset)$ with a five-day calendar window using a six-standard deviation default. Negative/non-finite non-missing values and exact copied calendar blocks are hard failures. Other tests remain evidence for review; the default requires three independent families before point rejection because floods, regulation, and intermittent flow can legitimately trigger individual statistical rules.

High-flow diagnostics combine the six-standard-deviation, largest-AMAX ratio, and GEV return-period tests into one evidence family so correlated extreme statistics are not counted as independent proof. Spatial QC fits pairwise log-flow regressions, optionally on flow divided by catchment area, and flags a rare event only when its regression residual is large and too few selected neighbours show high flow within the search window.

Station acceptance requires all user-defined completeness, review, confirmed suspicion, duration, and optional breakpoint criteria. Point and station decisions are therefore related but distinct.

Value

An object of class "streamflowQC"; see [streamflowQC-class](#).

References

Crochemore, L. et al. (2020). Lessons learnt from checking the quality of openly accessible river flow data worldwide. *Hydrological Sciences Journal*, 65, 699–711. doi:10.1080/02626667.2019.1659509.

Gudmundsson, L. et al. (2018). The Global Streamflow Indices and Metadata Archive (GSIM) - Part 2: quality control, time-series indices and homogeneity assessment. *Earth System Science Data*, 10, 787–804. doi:10.5194/essd-10-787-2018.

U.S. Integrated Ocean Observing System (2018). *Manual for Real-Time Quality Control of Stream Flow Data Version 1.0*. doi:10.25923/gszc-ha43.

See Also

[streamflowQC_subdaily](#), [streamflowQC](#), [streamflowQC_tests](#)

Examples

```
dates <- as.Date("2020-01-01") + 0:59
x <- zoo::zoo(cbind(A=10 + sin(seq_along(dates) / 5),
                   B=12 + sin(seq_along(dates) / 5)), dates)
x[20, "A"] <- -1
qc <- streamflowQC_daily(
  x, checks=c(duplicate=FALSE, climatology=FALSE, flatline=FALSE,
              spike=FALSE, rate=FALSE, highflow=FALSE,
              spatial=FALSE, breakpoint=FALSE),
  min.years=0, max.missing=1, max.review=1, max.suspicious=1
)
qc$suspicious
```

streamflowQC_subdaily *Quality control of sub-daily streamflow time series*

Description

Applies individually selectable physical, temporal, high-flow, spatial, and homogeneity tests to one or more regular minute/hourly streamflow series. It returns point flags and station acceptance recommendations without writing files.

Usage

```
streamflowQC_subdaily(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  checks=c(range=TRUE, gap=TRUE, climatology=TRUE, flatline=TRUE,
           spike=TRUE, rate=TRUE, fluctuation=TRUE, highflow=TRUE,
           spatial=TRUE, breakpoint=TRUE),
```

```

lower=0, upper=Inf,
climatology.z=6, climatology.min.samples=100L, log.offset=0.01,
flatline.hours=168, high.flatline.hours=24,
flatline.high.prob=0.99, flatline.tolerance=0, flag.zero=FALSE,
spike.factor=5, spike.prob=0.99,
rate.z=6, rate.min.samples=100L,
rate.max.rise=Inf, rate.max.fall=Inf, rate.drop.factor=5,
fluctuation.window=16L, fluctuation.minimum=8L,
fluctuation.low.diff.prob=0.05,
highflow.log.z=6, highflow.amax.ratio=2,
highflow.return.period=1000, highflow.min.years=10L,
spatial.window.hours=72, spatial.high.prob=0.99,
spatial.support.prob=0.95, spatial.f=3,
n.neighbours=5L, max.distance=200, min.neighbours=2L,
min.support=1L, min.overlap=100L, min.correlation=0.5,
breakpoint.min.years=10L, breakpoint.alpha=0.01,
breakpoint.min.relative.change=0.5,
breakpoint.min.completeness=0.8,
correction=c("none", "set_na", "spatial"),
min.evidence=3L, max.missing=0.2, max.review=0.2,
max.suspicious=0.05, min.years=1, discard.breakpoint=FALSE,
area=NULL, elevation=NULL, elevation.scale=500, verbose=TRUE
)

```

Arguments

<code>x</code>	A numeric regular sub-daily zoo object with a POSIXt index and one station per column.
<code>metadata</code>	NULL, or a data.frame containing every station in <code>x</code> . Extra fields are retained.
<code>station.id</code>	Name of the unique metadata station-identifier field.
<code>coords</code>	Names of longitude and latitude fields, in that order. Individual coordinates may be missing.
<code>checks</code>	Named logical vector. Omitted names retain their defaults; all sub-daily tests are enabled by default.
<code>lower, upper</code>	Physical limits, scalar or station-specific. The default rejects negative flow without imposing a region-specific maximum.
<code>climatology.z</code>	Standard-deviation multiplier for monthly log-flow.
<code>climatology.min.samples</code>	Minimum monthly reference observations.
<code>log.offset</code>	Positive constant added before taking logarithms.
<code>flatline.hours</code>	Duration in hours of a low/ordinary-flow flat line.
<code>high.flatline.hours</code>	Shorter flat-line duration applied above the station high-flow quantile.
<code>flatline.high.prob</code>	High-flow quantile for the shorter flat-line rule.

flatline.tolerance	Maximum consecutive numerical difference within a flat line.
flag.zero	Whether repeated zero-flow runs may be flagged.
spike.factor	Multiplicative isolated peak/dip threshold.
spike.prob	Station quantile used as the absolute isolated-spike threshold.
rate.z	Robust standardized first-difference threshold.
rate.min.samples	Minimum monthly first-difference reference sample.
rate.max.rise, rate.max.fall	Optional absolute consecutive rise and fall limits, scalar or station-specific.
rate.drop.factor	Multiplicative abrupt-fall threshold.
fluctuation.window	Number of observations in the moving sign-reversal window.
fluctuation.minimum	Number of derivative sign reversals that the window must exceed.
fluctuation.low.diff.prob	Quantile of absolute changes treated as measurement noise before counting reversals.
highflow.log.z	Standard-deviation multiplier for unusually high global log-flow.
highflow.amax.ratio	Ratio of the largest annual maximum to the second largest that generates a flag.
highflow.return.period	GEV return-period threshold in years.
highflow.min.years	Minimum annual maxima required for GEV fitting.
spatial.window.hours	Hours before and after a rare event searched for neighbouring support.
spatial.high.prob, spatial.support.prob	Target and neighbour high-flow quantiles.
spatial.f	Minimum standardized spatial-regression residual.
n.neighbours	Maximum selected neighbours per station.
max.distance	Maximum target-neighbour distance in km.
min.neighbours	Minimum simultaneous spatial predictions.
min.support	Minimum neighbouring high-flow events needed for corroboration.
min.overlap	Minimum paired observations for neighbour selection and regression.
min.correlation	Minimum preferred Spearman log-flow correlation.
breakpoint.min.years	Minimum complete annual indicators for a Pettitt test.
breakpoint.alpha	Holm-adjusted breakpoint significance level.

breakpoint.min.relative.change	Minimum relative pre/post median change.
breakpoint.min.completeness	Minimum daily completeness for an annual indicator.
correction	"none" preserves values, "set_na" removes confirmed points, and "spatial" uses an available spatial estimate and otherwise NA.
min.evidence	Independent flag families needed to confirm a non-hard-failure point.
max.missing, max.review, max.suspicious	Maximum station fractions of missing expected values, any flagged observations, and confirmed suspicious observations.
min.years	Minimum record duration for station acceptance.
discard.breakpoint	Whether a significant large breakpoint discards a station.
area	NULL, or the name of a positive numeric catchment-area field used to compare specific discharge spatially.
elevation	NULL, or an optional numeric elevation field.
elevation.scale	Positive elevation-decay scale used in neighbour ranking.
verbose	Logical. If TRUE, informative messages identify the workflow start, each enabled QC stage, evidence combination, and the final station/observation summary. If FALSE, only these informational messages are suppressed; warnings and errors remain visible.

Details

The defaults generalize the 15-minute UK-Flow15-QC rules to the actual modal interval. Thus, ordinary flat lines use 168 hours and high-flow flat lines use 24 hours rather than fixed counts of 672 and 96. Relative spikes and abrupt falls use a factor of five; absolute spikes use the station 0.99 quantile; and rapid fluctuations use more than eight sign reversals over 16 observations.

High-flow diagnostics include unusually large log-flow, an annual-maximum ratio, and a base-R maximum-likelihood GEV fit after at least 10 years. These correlated diagnostics contribute one evidence family. A spatial flag requires a rare target event, a large leave-one-station-out regression residual, and insufficient high-flow support among selected neighbours within the default 72-hour window. Catchment area, coordinates, and elevation are optional; correlation-based spatial comparison remains available without metadata.

Only physical range failure is hard evidence by default. The three-family default reduces the risk of rejecting genuine flash floods, reservoir operations, tidal behaviour, intermittent zeros, or high-baseflow plateaus. The review fraction can still recommend discarding a station with pervasive single-test artefacts.

Value

An object of class "streamflowQC"; see [streamflowQC-class](#).

References

- Fileni, F., Fowler, H. J., Lewis, E., McLay, F. and Yang, L. (2023). A quality-control framework for sub-daily flow and level data for hydrological modelling in Great Britain. *Hydrology Research*, 54, 1357–1367. doi:10.2166/nh.2023.045.
- Fileni, F. et al. (2026). UK-Flow15-QC: A quality control framework for better river flow data in hydrological research. *EGUsphere preprint*. doi:10.5194/egusphere-2026-277.
- U.S. Integrated Ocean Observing System (2018). *Manual for Real-Time Quality Control of Stream Flow Data Version 1.0*. doi:10.25923/gszc-ha43.
- Zhao, Q. et al. (2018). Research on the Data-Driven Quality Control Method of Hydrological Time Series Data. *Water*, 10, 1712. doi:10.3390/w10121712.

See Also

[streamflowQC_daily](#), [streamflowQC](#), [streamflowQC_tests](#)

Examples

```
times <- seq(as.POSIXct("2020-01-01", tz="UTC"),
             by="hour", length.out=72)
x <- zoo::zoo(cbind(A=10 + sin(seq_along(times) / 5),
                   B=12 + sin(seq_along(times) / 5)), times)
x[20, "A"] <- -1
qc <- streamflowQC_subdaily(
  x, checks=c(climatology=FALSE, flatline=FALSE, spike=FALSE,
              rate=FALSE, fluctuation=FALSE, highflow=FALSE,
              spatial=FALSE, breakpoint=FALSE),
  min.years=0, max.missing=1, max.review=1, max.suspicious=1
)
qc$suspicious
```

streamflowQC_tests	<i>Individual quality-control tests for streamflow</i>
--------------------	--

Description

Individual tests called by [streamflowQC_daily](#) and [streamflowQC_subdaily](#). They are exported so each source of evidence can be run, inspected, deactivated, and tuned independently.

Usage

```
streamflowQC_range(x, lower=0, upper=Inf)

streamflowQC_duplicate(
  x, min.month.values=20L, min.year.values=300L, min.nonzero=1L
)

streamflowQC_gap(x, interval=NULL)
```

```

streamflowQC_climatology(
  x, group=c("dayofyear", "month", "month_slot"), window=5L,
  z=6, min.samples=30L, offset=0.01
)

streamflowQC_flatline(
  x, run=11L, high.run=Inf, high.prob=0.99,
  tolerance=0, flag.zero=FALSE
)

streamflowQC_spike(x, factor=5, prob=0.99)

streamflowQC_rate(
  x, group=c("month", "month_slot"), z=6, min.samples=30L,
  max.rise=Inf, max.fall=Inf, drop.factor=5, offset=0.01
)

streamflowQC_fluctuation(
  x, window=16L, minimum=8L, low.diff.prob=0.05
)

streamflowQC_highflow(
  x, log.z=6, amax.ratio=2, return.period=1000,
  min.years=10L, offset=0.01
)

streamflowQC_spatial(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  area=NULL, elevation=NULL, elevation.scale=500,
  n.neighbours=5L, max.distance=400, min.neighbours=2L,
  min.support=1L, min.overlap=90L, min.correlation=0.5,
  window=3L, high.prob=0.99, support.prob=0.95, f=3
)

streamflowQC_breakpoint(
  x, min.years=10L, alpha=0.01, min.relative.change=0.5,
  min.completeness=0.8,
  indicators=c("mean", "sd", "minimum", "maximum")
)

```

Arguments

<code>x</code>	A numeric zoo object with one station per column.
<code>lower, upper</code>	Physical limits, scalar or station-specific.
<code>min.month.values, min.year.values</code>	Minimum paired calendar positions for exact copied-month and copied-year comparisons.

min.nonzero	Minimum positive observations in a candidate copied block.
interval	Expected interval: days for a Date index or seconds for a POSIXt index. NULL uses the modal difference.
group	Seasonal grouping. "month_slot" preserves the calendar month and hour:minute slot.
window	For climatology, circular calendar-day window width; for fluctuation, number of observations; for spatial QC, observations before and after the target event.
z	Standardized climatology or first-difference threshold.
min.samples	Minimum climatology or first-difference reference sample.
offset	Positive constant added before taking logarithms.
run, high.run	Minimum ordinary and high-flow flat-line lengths in observations.
high.prob	Target high-flow quantile.
tolerance	Maximum consecutive difference inside a flat line.
flag.zero	Whether repeated zero-flow runs may be flagged.
factor	Multiplicative isolated peak/dip threshold.
prob	Quantile used as the absolute spike threshold.
max.rise, max.fall	Optional absolute consecutive-change limits, scalar or station-specific.
drop.factor	Multiplicative abrupt-fall threshold.
minimum	Number of sign reversals that a fluctuation window must exceed.
low.diff.prob	Quantile of absolute changes treated as noise.
log.z	Global log-flow standard-deviation threshold.
amax.ratio	Largest/second-largest annual-maximum ratio threshold.
return.period	GEV return-period threshold in years.
min.years	Minimum annual values required for GEV or breakpoint analysis.
metadata	NULL, or station metadata containing all stations.
station.id	Name of the unique station identifier field.
coords	Names of longitude and latitude fields, in that order.
area	NULL, or the name of a positive numeric catchment-area field.
elevation	NULL, or the name of an optional numeric elevation field.
elevation.scale	Positive elevation-decay scale used in neighbour ranking.
n.neighbours	Maximum selected neighbours per station.
max.distance	Maximum geographic distance in km.
min.neighbours	Minimum simultaneous spatial predictions.
min.support	Minimum neighbours with a supporting high-flow event.
min.overlap	Minimum paired observations for selection and regression.
min.correlation	Minimum preferred Spearman log-flow correlation.

support.prob	Neighbour high-flow quantile.
f	Minimum standardized spatial residual.
alpha	Holm-adjusted Pettitt-test significance level.
min.relative.change	Minimum relative pre/post median change.
min.completeness	Minimum daily completeness of an annual indicator.
indicators	Unique subset of "mean", "sd", "minimum", and "maximum".

Details

streamflowQC_range treats negative discharge and optional station-specific upper violations as physical errors. Missing values are handled by the workflow completeness calculation. Non-finite non-missing values fail the range check.

streamflowQC_duplicate finds exactly copied calendar months or years. It is intended for daily data; requiring a positive value prevents ordinary all-zero intermittent-flow blocks from being flagged by default.

streamflowQC_gap marks the first finite observation after a missing value or missing interval. It complements, rather than replaces, station completeness. A numeric zoo cannot retain transmission syntax, so the QARTOD syntax test is enforced structurally by the workflow validators.

streamflowQC_climatology applies the GSIM transformation $\log(Q + 0.01)$ and mean plus/minus six standard deviations. Daily references use a circular calendar-day window; sub-daily references may use month or month-slot groups.

streamflowQC_flatline flags every member of an extended near-constant run. Daily defaults implement more than 10 equal positive values from GSIM. The sub-daily workflow converts the UK-Flow15-QC seven-day and one-day high-flow rules from hours to input observations.

streamflowQC_spike requires an isolated centre value to change and reverse by a factor of five or by more than the station 0.99 quantile. streamflowQC_rate combines robust monthly log-difference screening, optional absolute limits, and the five-fold rapid-drop rule. streamflowQC_fluctuation flags windows with recurrent first-derivative sign reversals after suppressing the smallest changes.

streamflowQC_highflow returns one combined flag plus separate six-standard-deviation, AMAX-ratio, and GEV components. The GEV is fitted to annual maxima by base-R maximum likelihood. The combined flag deliberately counts as one evidence family because all three statistics describe the same extreme event.

streamflowQC_spatial selects neighbours using correlation and optional geographic/elevation proximity, fits target-on-neighbour regressions in log-specific-discharge space when catchment area is available, and combines predictions using inverse-error weighted medians. A rare target event is flagged only when its residual score exceeds f and fewer than min.support neighbours exceed their support quantile in the search window.

streamflowQC_breakpoint aggregates sub-daily input to daily means, calculates complete annual indicators, applies Pettitt rank tests, and Holm-adjusts p-values. Its station flag is separate from point QC because an inhomogeneity may represent real regulation or catchment change.

Value

Range, duplicate, gap, climatology, flatline, spike, rate, and fluctuation tests return a logical zoo matching x.

streamflowQC_highflow returns a list with combined flags, component flag objects, return.period, and station fit diagnostics.

streamflowQC_spatial returns a list with flags, standardized scores, flow estimate, neighbour-event support.count, and selected neighbours.

streamflowQC_breakpoint returns one data.frame row per station.

References

Fileni, F. et al. (2026). UK-Flow15-QC: A quality control framework for better river flow data in hydrological research. *EGUsphere preprint*. doi:10.5194/egusphere-2026-277.

Gudmundsson, L. et al. (2018). The Global Streamflow Indices and Metadata Archive (GSIM) - Part 2: quality control, time-series indices and homogeneity assessment. *Earth System Science Data*, 10, 787–804. doi:10.5194/essd-10-787-2018.

U.S. Integrated Ocean Observing System (2018). *Manual for Real-Time Quality Control of Stream Flow Data Version 1.0*. doi:10.25923/gszc-ha43.

Zhao, Q. et al. (2018). Research on the Data-Driven Quality Control Method of Hydrological Time Series Data. *Water*, 10, 1712. doi:10.3390/w10121712.

See Also

[streamflowQC_daily](#), [streamflowQC_subdaily](#)

Examples

```
dates <- as.Date("2020-01-01") + 0:19
x <- zoo::zoo(cbind(A=c(rep(10, 11), 11:19),
                    B=seq(5, 24)), dates)
streamflowQC_flatline(x)
streamflowQC_range(x)
```

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

<code>x</code>	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 <code>x</code> represents the time series measured in each gauging station, and the column names of <code>x</code> should correspond to the ID of each station (starting by a letter).
<code>FUN</code>	Function that have to be applied for aggregating from sub-daily into daily time step (e.g., for precipitation <code>FUN=sum</code> and for temperature and streamflows <code>ts FUN=mean</code>). <code>FUN</code> MUST accept the <code>na.rm</code> argument, because <code>na.rm</code> is passed to <code>FUN</code>.
<code>na.rm</code>	Logical. Should missing values be removed? -) <code>TRUE</code> : the daily values are computed only for days with a percentage of missing values less than <code>na.rm.max</code> -) <code>FALSE</code>: if there is AT LEAST one NA within a day, the corresponding daily value in the output object will be NA.
<code>na.rm.max</code>	Numeric in <code>[0, 1]</code>. 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 <code>na.rm.max</code> the corresponding daily value will be NA.
<code>start</code>	character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by <code>start.fmt</code>. 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.

<code>start.fmt</code>	character indicating the format in which the time is provided in <code>start</code> , By default <code>date.fmt=%H:%M:%S</code> . See format in as.POSIXct .
<code>tz</code>	character, with the specification of the time zone used in both <code>x</code> and <code>start</code> . 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 <code>tz</code> is missing (the default), it is automatically set to the time zone used in <code>time(x)</code> . This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.
<code>dates</code>	numeric, factor, <code>POSIXct</code> or <code>POSIXt</code> object indicating how to obtain the dates and times for each column of <code>x</code> (e.g., gauging station) If <code>dates</code> is a number, it indicates the index of the column in <code>x</code> that stores the date and times If <code>dates</code> is a factor, it is converted into <code>POSIXct</code> class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of <code>POSIXct</code> or <code>POSIXt</code> class, the code verifies that the number of elements on it be equal to the number of elements in <code>x</code>
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>dates</code> , By default <code>date.fmt=%Y-%m-%d %H:%M:%S</code> . See format in as.Date . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
<code>out.fmt</code>	OPTIONAL. Only used when <code>x</code> 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: <code>numeric</code> , <code>zoo</code> (default)
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	arguments additional to <code>na.rm</code> 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

## Aggregating 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: Computation 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 corresponding 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.

<code>start.fmt</code>	character indicating the format in which the time is provided in <code>start</code> , By default <code>date.fmt=%H:%M:%S</code> . See format in as.POSIXct .
<code>tz</code>	character, with the specification of the time zone used in both <code>x</code> and <code>start</code> . 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 <code>tz</code> is missing (the default), it is automatically set to the time zone used in <code>time(x)</code> . This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.
<code>dates</code>	numeric, factor or Date object indicating how to obtain the dates for each gauging station If <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates If <code>dates</code> is a factor, it is converted into Date class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of Date class, the code verifies that the number of days on it be equal to the number of elements in <code>x</code>
<code>date.fmt</code>	character indicating the format in which the DateTime objects are stored in <code>dates</code> , e.g. %Y-%m-%d %H:%M:%S. See format in as.POSIXct . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
<code>out.fmt</code>	OPTIONAL. Only used when <code>x</code> 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.
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	arguments additional to <code>na.rm</code> 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)
```

subhourly2nhourly *Sub-hourly or hourly -> n-hourly*

Description

Generic function for aggregating a sub-hourly or hourly time series into a user-defined n-hourly one.

Usage

```
subhourly2nhourly(x, ...)

## Default S3 method:
subhourly2nhourly(x, FUN, n.hours, na.rm=TRUE, na.rm.max=0,
                  start="00:00:00", start.fmt="%H:%M:%S", tz, ...)

## S3 method for class 'zoo'
subhourly2nhourly(x, FUN, n.hours, na.rm=TRUE, na.rm.max=0,
                  start="00:00:00", start.fmt="%H:%M:%S", tz, ...)

## S3 method for class 'data.frame'
subhourly2nhourly(x, FUN, n.hours, 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'
subhourly2nhourly(x, FUN, n.hours, 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-hourly or 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 represents the time series measured in each gauging station, and the column names of x should correspond to the ID of each station.
FUN	Function that have to be applied for aggregating from sub-hourly or hourly into n-hourly 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.
n.hours	positive integer indicating the length, in hours, of each output aggregation period.

<code>na.rm</code>	Logical. Should missing values be removed? -) TRUE : the n-hourly values are computed only for periods with a percentage of missing values less than or equal to <code>na.rm.max</code> -) FALSE: if there is AT LEAST one NA within an n-hour period, the corresponding n-hourly value in the output object will be NA.
<code>na.rm.max</code>	Numeric in [0, 1]. It is used to define the maximum percentage of missing values allowed in each n-hour period to keep the aggregated value in the output object of this function. In other words, if the percentage of missing values in a given n-hour period is larger than <code>na.rm.max</code>, the corresponding n-hourly value will be NA.
<code>start</code>	character, indicating the starting time used for anchoring the n-hour aggregation periods. It MUST be provided in the format specified by <code>start.fmt</code>.
<code>start.fmt</code>	character indicating the format in which the time is provided in <code>start</code>, By default <code>date.fmt=%H:%M:%S</code>. See format in as.POSIXct.
<code>tz</code>	character, with the specification of the time zone used in both <code>x</code> and <code>start</code>. 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 <code>tz</code> is missing (the default), it is automatically set to the time zone used in <code>time(x)</code>. If <code>tz</code> is provided, it forces <code>time(x)</code> to be in the time zone specified by <code>tz</code>, without modifying the values (hours, minutes, seconds, etc).
<code>dates</code>	numeric, factor, character, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of <code>x</code> (e.g., gauging station) If <code>dates</code> is a number, it indicates the index of the column in <code>x</code> that stores the date and times If <code>dates</code> is a factor or character, it is converted into POSIXct class, using the date format specified by <code>date.fmt</code> If <code>dates</code> 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 <code>x</code>
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>dates</code>, By default <code>date.fmt=%Y-%m-%d %H:%M:%S</code>. See format in as.Date. ONLY required when <code>class(dates)=="factor"</code>, <code>class(dates)=="character"</code> or <code>class(dates)=="numeric"</code>.
<code>out.fmt</code>	OPTIONAL. Only used when <code>x</code> is a matrix or data.frame object /or character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: numeric, zoo (default)
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	arguments additional to <code>na.rm</code> passed to FUN.

Details

This function assumes that `x` has an hourly or sub-hourly temporal frequency. The output time index corresponds to the beginning of each n-hour aggregation period.

Value

a zoo object with n-hourly time series

Author(s)

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

See Also

[subhourly2hourly](#), [subhourly2nminutes](#), [subdaily2daily](#), [subdaily2monthly](#), [subdaily2annual](#), [subdaily2seasonal](#), [as.POSIXct](#), [hydroplot](#), [vector2zoo](#), [izoo2rzoo](#)

Examples

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

## Creating a dummy 30-min zoo object, with 1 as the only value in each time period
x <- zoo(rep(1, ndt), dt)

## Aggregation from 30-minute single ts into 3-hourly ts
h1 <- subhourly2nhourly(x, FUN=sum, n.hours=3, na.rm=TRUE)

## Aggregation of 3 ts with 30-minute time frequency (i.e., a zoo matrix)
## into a 6-hourly zoo object.
X <- cbind(x, x, x)
h2 <- subhourly2nhourly(X, FUN=sum, n.hours=6, na.rm=TRUE)

## Hourly input into 6-hourly output
h <- subhourly2hourly(x, FUN=sum)
h3 <- subhourly2nhourly(h, FUN=sum, n.hours=6, 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 <code>time(x)[2]-time(x)[1]</code>).
FUN	Function that have to be applied for aggregating from sub-hourly into n-minutes time step. (e.g., for precipitation <code>FUN=sum</code> and for temperature and streamflow <code>ts</code> , <code>FUN=mean</code>).
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 <code>from > start(x)</code> then x is cut in time to the starting DateTime defined by from. When <code>from < start(x)</code> 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 <code>to < end(x)</code> then x is cut in time to the ending DateTime defined by to. When <code>to > end(x)</code> 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 time zone specified by `tz`, without modifying 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.

<code>dates</code>	numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each column of <code>x</code> (e.g., gauging station) If <code>dates</code> is a number, it indicates the index of the column in <code>x</code> that stores the date and times If <code>dates</code> is a factor, it is converted into POSIXct class, using the date format specified by <code>date.fmt</code> If <code>dates</code> 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 <code>x</code>
<code>date.fmt</code>	character indicating the format in which the dates are stored in <code>dates</code>, By default <code>date.fmt=%Y-%m-%d %H:%M:%S</code>. See format in as.Date. ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code>.
<code>out.fmt</code>	OPTIONAL. Only used when <code>x</code> is a matrix or data.frame object /or character, for selecting if the result will be a matrix/data.frame or a zoo object. Valid values are: <code>numeric</code>, <code>zoo</code> (default)
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	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)

```

tempQC

Frequency-aware quality control of air-temperature time series

Description

Detects the sampling frequency of a numeric zoo object with [sfreq](#) and dispatches it to [tempQC_daily](#) or [tempQC_subdaily](#). No files are written.

Usage

```

tempQC(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"), ...,
  elevation=NULL, elevation.scale=500, verbose=TRUE
)

```

Arguments

<code>x</code>	A numeric zoo object. Each column is one station.
<code>metadata, station.id, coords, elevation, elevation.scale</code>	Optional station metadata, field-name selectors, and elevation-decay control passed to the selected workflow. See tempQC_daily .
<code>...</code>	Arguments passed unchanged to the selected workflow.
<code>verbose</code>	Logical. If TRUE, informative messages report the detected frequency, active QC stages, and final QC summary. If FALSE, these informational messages are suppressed; warnings and errors are unaffected.

Details

Daily input is sent to [tempQC_daily](#); minute or hourly input is sent to [tempQC_subdaily](#). Other frequencies are rejected explicitly. Automatic dispatch changes only the workflow: test activation, thresholds, metadata, correction policy, and station recommendation criteria remain under user control. Progress is reported with `message()`, so it can be captured or suppressed independently of the returned result. The staged design follows the gross-error, tolerance, temporal, inter-variable, and inter-station QC categories described by Brunet et al. (2020).

Value

An object of class "tempQC"; see [tempQC-class](#).

References

Brunet, M. et al. (2020). *Best Practice Guidelines for Climate Data and Metadata Formatting, Quality Control and Submission*. Copernicus Climate Change Service.

See Also

[tempQC_daily](#), [tempQC_subdaily](#), [tempQC_tests](#), [precipQC](#)

Examples

```
dates <- as.Date("2020-01-01") + 0:29
x <- zoo::zoo(cbind(A=10 + sin(seq_along(dates)),
                    B=11 + sin(seq_along(dates))), dates)
qc <- tempQC(x, checks=c(range=TRUE, duplicate=FALSE,
                        climatology=FALSE, persistence=FALSE,
                        step=FALSE, spike=FALSE, spatial=FALSE,
                        breakpoint=FALSE), min.years=0)

qc$station.summary
```

tempQC-class	<i>Air-temperature quality-control result</i>
--------------	---

Description

Structure, print method, and four-panel summary plot for objects returned by [tempQC_daily](#), [tempQC_subdaily](#), and [tempQC](#).

Usage

```
## S3 method for class 'tempQC'
print(x, ...)

## S3 method for class 'tempQC'
plot(
  x, max.stations=20L,
  col=c("#2878B5", "#D55E00", "#E69F00"), ...
)
```

Arguments

- x An object inheriting from class "tempQC".
- max.stations Maximum number of stations in the affected-station panel.
- col At least three plotting colours.
- ... Additional arguments; the plot method passes them to its final time-series panel.

Details

The plot summarizes station recommendations, missing and confirmed-suspicious fractions, counts by active test, and the time distribution of rejected observations. It is diagnostic and does not alter the result.

Value

Both methods return x invisibly. A "tempQC" object is a list with:

- `accepted.metadata` and `discarded.metadata`: station metadata plus recommendation diagnostics;
- `accepted.data`: original zoo series for stations recommended for acceptance;
- `accepted.corrected`: the same accepted stations after the requested correction policy;
- `suspicious`: one row per point flagged for review or rejection;
- `corrections`: an audit table of changed values;
- `flags`, `flag.count`, and `rejected`: individual logical flag objects, evidence counts, and confirmed decisions;
- `station.summary`: completeness, flag fractions, breakpoint diagnostics, recommendation, and reason;
- `breakpoint`: station homogeneity diagnostics;
- `spatial.estimate`, `spatial.score`, and `neighbours`: spatial-test diagnostics; and
- `settings`: resolved workflow settings.

References

Brunet, M. et al. (2020). *Best Practice Guidelines for Climate Data and Metadata Formatting, Quality Control and Submission*. Copernicus Climate Change Service.

See Also

[tempQC_daily](#), [tempQC_subdaily](#), [tempQC_tests](#)

tempQC_daily	<i>Quality control of daily air-temperature time series</i>
--------------	---

Description

Applies a conservative, multi-test quality-control workflow to one or more daily air-temperature station series in a zoo object. It returns auditable point flags, optional corrections, and station acceptance or discard recommendations without writing to disk.

Usage

```
tempQC_daily(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  checks=c(range=TRUE, duplicate=TRUE, climatology=TRUE,
           persistence=TRUE, step=TRUE, spike=TRUE,
           spatial=TRUE, breakpoint=TRUE),
  lower=-89.4, upper=57.7,
  duplicate.min.month=20L, duplicate.min.year=300L,
  climatology.window=15L, climatology.z=6,
  climatology.min.samples=100L,
  persistence.run=7L, persistence.tolerance=0,
  step.z=3, step.min.samples=20L,
  step.max.rise=Inf, step.max.fall=Inf,
  spike.threshold=25,
  n.neighbours=5L, max.distance=200, min.neighbours=2L,
  min.overlap=90L, min.group.overlap=20L,
  min.correlation=0.6, spatial.f=4, spatial.min.difference=8,
  breakpoint.min.years=10L, breakpoint.alpha=0.05,
  breakpoint.min.shift=1, breakpoint.min.completeness=0.8,
  correction=c("none", "set_na", "spatial"),
  min.evidence=2L, max.missing=0.2, max.suspicious=0.05,
  min.years=1, discard.breakpoint=FALSE,
  elevation=NULL, elevation.scale=500, verbose=TRUE
)
```

Arguments

x	A numeric daily zoo object with one uniquely named station column per series. Values are assumed to be degrees Celsius.
metadata	NULL, or a data.frame with one row per station. Additional columns are retained in both metadata outputs. When supplied, the selected identifier, longitude, and latitude fields are required. Rows are matched to colnames(x) and need not be pre-sorted.
station.id	Name of the metadata column containing the column names of x.
coords	Names of numeric longitude and latitude metadata columns in decimal degrees, in that order. Entries may be NA; a station is located only when both coordinates are present. Non-missing values are checked against the valid longitude and latitude ranges.
elevation	NULL, or the name of an optional numeric metadata column containing elevation in metres.
elevation.scale	Positive elevation-decay scale in metres. It controls how strongly height differences reduce neighbour suitability and regression weights. It is ignored when elevation=NULL.
checks	Named logical vector used to activate or deactivate any of "range", "duplicate", "climatology", "persistence", "step", "spike", "spatial", and "breakpoint". All are active by default.

lower, upper	Physical screening limits. Each can be scalar or contain one value per station. Regionally authoritative limits should replace the defaults when available.
duplicate.min.month, duplicate.min.year	Minimum number of complete, paired calendar positions needed to declare two months or years identical.
climatology.window	Width in days of the circular day-of-year reference window.
climatology.z	Absolute robust standardized anomaly threshold.
climatology.min.samples	Minimum reference sample size for a calendar-window climatology.
persistence.run	Minimum number of consecutive values in a flat line.
persistence.tolerance	Maximum range within a flat-line window.
step.z	Robust standardized threshold for a consecutive-day change.
step.min.samples	Minimum monthly sample of first differences.
step.max.rise, step.max.fall	Optional absolute rise and fall limits; Inf leaves the month-specific statistical limits in control.
spike.threshold	Minimum absolute difference from both adjacent days, with the two differences having the same sign, for an isolated spike or dip.
n.neighbours	Maximum number of spatial-regression neighbours.
max.distance	Maximum target-neighbour great-circle distance in km.
min.neighbours	Minimum number of simultaneous regression estimates.
min.overlap	Minimum paired observations for initial neighbour selection.
min.group.overlap	Minimum paired observations within a calendar month for fitting an individual regression.
min.correlation	Minimum Pearson correlation for an eligible neighbour.
spatial.f	Minimum standardized spatial-regression residual.
spatial.min.difference	Minimum absolute observed-estimated difference in degrees Celsius. Both spatial criteria must be exceeded.
breakpoint.min.years	Minimum complete annual indicators for a Pettitt test.
breakpoint.alpha	Holm-adjusted significance level for breakpoint evidence.
breakpoint.min.shift	Minimum absolute difference between median annual indicators before and after a breakpoint.

breakpoint.min.completeness	Minimum fraction of daily observations in an annual indicator.
correction	"none" preserves values, "set_na" replaces confirmed suspicious values by NA, and "spatial" uses the spatial estimate where available and NA otherwise.
min.evidence	Number of coincident non-hard flags needed to reject a point. Physical range, copied-block, flat-line, and isolated-spike flags are hard evidence.
max.missing, max.suspicious	Maximum missing and confirmed-suspicious fractions for recommending a station for acceptance.
min.years	Minimum record length for station acceptance.
discard.breakpoint	Logical. If TRUE, a flagged breakpoint causes a discard recommendation. The default is conservative because a change may be climatic.
verbose	Logical. If TRUE, informative messages identify the workflow start, each enabled test, evidence combination, and the final station/observation summary. If FALSE, only these informational messages are suppressed; warnings and errors remain visible.

Details

The tests represent independent error mechanisms and are deliberately combined as evidence instead of interpreting every statistical outlier as an error.

1. *Physical range and copied blocks.* Values beyond observed world limits and exactly duplicated months or years are gross-integrity failures. The limits and block comparisons follow Durre et al. (2010). The duplicate test requires complete paired calendar positions to limit false positives.
2. *Climatological plausibility.* For each station and calendar day, the median and MAD are calculated from a moving 15-day window. An absolute robust score of six is flagged. This is a resistant implementation of the 15-day, six-standard-deviation test in Durre et al. (2010) and the biweight calendar-window approach of Feng et al. (2004).
3. *Persistence, steps, and spikes.* Flat lines expose stuck sensors or copied reports (Feng et al., 2004; Boulanger et al., 2010). First differences are checked against month-specific robust limits, adapting the seasonal rate-of-change test of Hubbard et al. (2007). A value at least 25 degrees from both neighbours in the same direction implements the daily spike/dip criterion of Durre et al. (2010).
4. *Spatial regression.* Month-specific linear regressions predict the target from eligible correlated stations. The five regressions with the lowest proximity-adjusted errors are combined with proximity/inverse-error-variance weights. Proximity decays with horizontal distance and, when selected, absolute elevation difference. The pairwise regression intercept estimates systematic station offsets rather than imposing a universal lapse rate. A target missing either coordinate uses correlation and regression error without a distance filter. A located target ranks eligible located neighbours first; unlocated neighbours remain conservative fallbacks with reduced spatial weight. A point is flagged only if both its absolute residual and its standardized residual are large. This combines the SRT formulation of Hubbard et al. (2007) and Estevez et al. (2018) with the conservative dual threshold of Durre et al. (2010). Correlation-based selection is important in complex terrain (Xiong et al., 2022; Xiong et al., 2024).

5. *Homogeneity*. Pettitt tests are applied to sufficiently complete annual mean and variability series and adjusted by Holm's method. The result is diagnostic by default, reflecting the distinction between QC and homogenization emphasized by Boulanger et al. (2010) and Brunet et al. (2020).

The paired logical constraints $T_{min} \leq T_{max}$ and lagged minimum- maximum checks cannot be inferred from a single univariate station matrix. Use `tempQC_internal` when paired Tmin and Tmax zoo objects are available.

Value

An object of class "tempQC"; see `tempQC-class`.

References

- Boulanger, J.-P., Aizpuru, J., Leggieri, L. and Marino, M. (2010). A procedure for automated quality control and homogenization of historical daily temperature and precipitation data (APACH): part 1. *Climatic Change*, 98, 471–491.
- Brunet, M. et al. (2020). *Best Practice Guidelines for Climate Data and Metadata Formatting, Quality Control and Submission*. Copernicus Climate Change Service.
- Durre, I., Menne, M. J., Gleason, B. E., Houston, T. G. and Vose, R. S. (2010). Comprehensive automated quality assurance of daily surface observations. *Journal of Applied Meteorology and Climatology*, 49, 1615–1633.
- Estevez, J., Gavilan, P. and Garcia-Marin, A. P. (2018). Spatial regression test for ensuring temperature data quality in southern Spain. *Theoretical and Applied Climatology*, 131, 309–318.
- Feng, S., Hu, Q. and Qian, W. (2004). Quality control of daily meteorological data in China, 1951–2000. *International Journal of Climatology*, 24, 853–870.
- Hubbard, K. G., Guttman, N. B., You, J. and Chen, Z. (2007). An improved QC process for temperature in the daily cooperative weather observations. *Journal of Atmospheric and Oceanic Technology*, 24, 206–213.
- Xiong, X. et al. (2022). Research on quality control methods for surface temperature observations via spatial correlation analysis. *International Journal of Climatology*. doi:10.1002/joc.7897.
- Xiong, X. et al. (2024). Research on quality control method of surface temperature observations for complex physical geography. *Journal of Atmospheric and Oceanic Technology*, 41, 803–815.

See Also

`tempQC`, `tempQC_subdaily`, `tempQC_tests`, `plot.tempQC`

Examples

```
dates <- as.Date("2020-01-01") + 0:89
v <- 12 + 8 * sin(2 * pi * seq_along(dates) / 365)
x <- zoo::zoo(cbind(A=v, B=v + 0.3), dates)
qc <- tempQC_daily(
  x, checks=c(range=TRUE, duplicate=FALSE, climatology=FALSE,
              persistence=TRUE, step=FALSE, spike=TRUE,
              spatial=FALSE, breakpoint=FALSE),
```

```

    min.years=0, max.missing=1
  )
  qc$station.summary

```

tempQC_subdaily

Quality control of sub-daily air-temperature time series

Description

Applies physical, climatological, temporal, spatial, and homogeneity checks to minute or hourly air-temperature station series in a zoo object. The method preserves the native time index and writes no files.

Usage

```

tempQC_subdaily(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  checks=c(range=TRUE, climatology=TRUE, persistence=TRUE,
    step=TRUE, spike=TRUE, spatial=TRUE, breakpoint=TRUE),
  lower=-89.4, upper=57.7,
  climatology.z=6, climatology.min.samples=30L,
  persistence.hours=3, persistence.tolerance=0.05,
  step.z=5, step.min.samples=30L,
  step.max.rise=Inf, step.max.fall=Inf,
  spike.threshold=12,
  n.neighbours=5L, max.distance=50, min.neighbours=2L,
  min.overlap=100L, min.group.overlap=20L,
  min.correlation=0.7, spatial.f=4, spatial.min.difference=5,
  breakpoint.min.years=10L, breakpoint.alpha=0.05,
  breakpoint.min.shift=1, breakpoint.min.completeness=0.8,
  correction=c("none", "set_na", "spatial"),
  min.evidence=2L, max.missing=0.2, max.suspicious=0.05,
  min.years=1, discard.breakpoint=FALSE,
  elevation=NULL, elevation.scale=500, verbose=TRUE
)

```

Arguments

x	A numeric zoo object with a strictly increasing POSIXt index at minute or hourly resolution. Each column is a station in degrees Celsius.
metadata	NULL, or station metadata with one row per column. When supplied, it must include the selected identifier, longitude, and latitude fields.
station.id	Name of the station identifier column in metadata.
coords	Names of longitude and latitude metadata columns in decimal degrees, in that order. Individual entries may be NA; non-missing values are range-checked.
elevation	NULL, or the name of an optional numeric elevation column in metres.

<code>elevation.scale</code>	Positive height-decay scale in metres used in neighbour selection and weighting when elevation is supplied.
<code>checks</code>	Named logical vector that overrides any of "range", "climatology", "persistence", "step", "spike", "spatial", and "breakpoint".
<code>lower, upper</code>	Physical screening limits, scalar or station-specific.
<code>climatology.z</code>	Absolute robust anomaly threshold within each calendar-month and hour-of-day group.
<code>climatology.min.samples</code>	Minimum group sample size.
<code>persistence.hours</code>	Duration of an almost-flat period. The corresponding number of observations is derived from the modal sampling interval.
<code>persistence.tolerance</code>	Maximum temperature range within that period.
<code>step.z</code>	Robust standardized threshold for consecutive changes within each month and hour.
<code>step.min.samples</code>	Minimum first-difference sample per group.
<code>step.max.rise, step.max.fall</code>	Optional absolute consecutive rise and fall limits; use regional, interval-specific values when available.
<code>spike.threshold</code>	Minimum difference from both adjacent observations for an isolated spike or dip.
<code>n.neighbours, max.distance, min.neighbours</code>	Maximum neighbour count, maximum distance in km, and minimum simultaneous estimates.
<code>min.overlap, min.group.overlap</code>	Minimum overall and month-hour paired samples for neighbour selection and regression fitting.
<code>min.correlation</code>	Minimum eligible target-neighbour Pearson correlation.
<code>spatial.f</code>	Minimum standardized spatial residual.
<code>spatial.min.difference</code>	Minimum absolute spatial residual in degrees Celsius. Both spatial thresholds must be exceeded.
<code>breakpoint.min.years, breakpoint.alpha, breakpoint.min.shift, breakpoint.min.completeness</code>	Minimum annual sample size, Holm-adjusted significance level, minimum absolute median shift, and annual completeness for the homogeneity diagnostic.
<code>correction</code>	"none", "set_na", or "spatial"; see tempQC_daily .
<code>min.evidence</code>	Number of coincident non-hard tests needed to reject a point. Range, persistence, and isolated-spike flags are hard evidence.

max.missing, max.suspicious, min.years	Station acceptance limits.
discard.breakpoint	Whether a flagged breakpoint should cause a station discard recommendation.
verbose	Logical. If TRUE, informative messages identify the workflow start, each enabled test, evidence combination, and the final station/observation summary. If FALSE, only these informational messages are suppressed; warnings and errors remain visible.

Details

The sub-daily workflow uses the same evidence and audit model as [tempQC_daily](#), but its climatological, first-difference, and spatial-regression reference groups are conditioned jointly on calendar month and hour of day. Cheng et al. (2016) showed that hourly reference-station relations can change sharply by hour and season, especially across complex terrain.

1. *Range*: global physical limits screen gross coding and unit errors. Local limits may be supplied; elevation-adjusted regional limits are preferable where established (Cheng et al., 2016).
2. *Climatology*: robust median/MAD limits within month-hour groups preserve the diurnal and seasonal cycle (Cheng et al., 2016; Brunet et al., 2020).
3. *Persistence*: the default identifies a temperature range no greater than 0.05 degrees across three hours, adapting the 10-minute flat-line test of Beele et al. (2022) to the detected sampling interval.
4. *Steps and isolated spikes*: month-hour first differences are standardized robustly. Optional absolute rise and fall limits support local real-time criteria. Cheng et al. (2016) demonstrate why weather- and hour-dependent limits are safer than a universal hourly threshold; Beele et al. (2022) provide an operational 10-minute example.
5. *Spatial regression*: correlated neighbours are chosen by proximity-adjusted regression error and combined with proximity and inverse-error-variance weights. Geographic distance and, when supplied, elevation difference define proximity. A target missing either coordinate uses the correlation/regression fallback, while a located target prefers eligible located neighbours. Models are fitted separately by month and hour. This implements the SRT family evaluated by Hubbard et al. (2007), Cheng et al. (2016), and Estevez et al. (2018), while the correlation screen addresses the terrain sensitivity discussed by Xiong et al. (2022, 2024).
6. *Homogeneity*: native values are converted to daily means before annual mean and variability breakpoint diagnostics. The result is not a point flag and does not cause discard unless explicitly requested.

Value

An object of class "tempQC"; see [tempQC-class](#).

References

Beele, E., Reyniers, M., Aerts, R. and Somers, B. (2022). Quality control and correction method for air temperature data from a citizen science weather station network in Leuven, Belgium. *Earth System Science Data*, 14, 4681–4717.

- Brunet, M. et al. (2020). *Best Practice Guidelines for Climate Data and Metadata Formatting, Quality Control and Submission*. Copernicus Climate Change Service.
- Cheng, A. R., Lee, T. H., Ku, H. I. and Chen, Y. W. (2016). Quality control program for real-time hourly temperature observation in Taiwan. *Journal of Atmospheric and Oceanic Technology*, 33, 953–965.
- Estevez, J., Gavilan, P. and Garcia-Marin, A. P. (2018). Spatial regression test for ensuring temperature data quality in southern Spain. *Theoretical and Applied Climatology*, 131, 309–318.
- Hubbard, K. G., Guttman, N. B., You, J. and Chen, Z. (2007). An improved QC process for temperature in the daily cooperative weather observations. *Journal of Atmospheric and Oceanic Technology*, 24, 206–213.
- Xiong, X. et al. (2022). Research on quality control methods for surface temperature observations via spatial correlation analysis. *International Journal of Climatology*. doi:10.1002/joc.7897.
- Xiong, X. et al. (2024). Research on quality control method of surface temperature observations for complex physical geography. *Journal of Atmospheric and Oceanic Technology*, 41, 803–815.

See Also

[tempQC](#), [tempQC_daily](#), [tempQC_tests](#), [plot.tempQC](#)

Examples

```
times <- seq(as.POSIXct("2020-01-01", tz="UTC"), by="hour", length.out=72)
h <- as.integer(format(times, "%H"))
v <- 12 + 5 * sin(2 * pi * h / 24)
x <- zoo::zoo(cbind(A=v, B=v + 0.2), times)
qc <- tempQC_subdaily(
  x, checks=c(range=TRUE, climatology=FALSE, persistence=TRUE,
              step=FALSE, spike=TRUE, spatial=FALSE,
              breakpoint=FALSE),
  min.years=0, max.missing=1
)
qc$station.summary
```

tempQC_tests

Individual quality-control tests for air temperature

Description

Individual tests called by [tempQC_daily](#) and [tempQC_subdaily](#). They are exported so each source of evidence can be run, inspected, and tuned independently.

Usage

```
tempQC_range(x, lower=-89.4, upper=57.7)

tempQC_duplicate(x, min.month.values=20L, min.year.values=300L)
```

```

tempQC_persistence(x, run=7L, tolerance=0)

tempQC_climatology(
  x, group=c("dayofyear", "month", "month_hour"),
  window=15L, z=6, min.samples=30L
)

tempQC_step(
  x, group=c("month", "month_hour"), z=3,
  min.samples=20L, max.rise=Inf, max.fall=Inf
)

tempQC_spike(x, threshold=25)

tempQC_spatial(
  x, metadata=NULL, station.id="station", coords=c("lon", "lat"),
  group=c("month", "month_hour"), n.neighbours=5L,
  max.distance=200, min.neighbours=2L, min.overlap=90L,
  min.group.overlap=20L, min.correlation=0.6,
  f=4, min.difference=8, min.se=0.1, elevation=NULL,
  elevation.scale=500
)

tempQC_internal(tmin, tmax, lagged=TRUE, max.range=40)

tempQC_breakpoint(
  x, min.years=10L, alpha=0.05, min.shift=1,
  min.completeness=0.8, indicators=c("mean", "sd")
)

```

Arguments

<code>x</code>	A numeric zoo object with one station per column.
<code>lower, upper</code>	Physical limits, scalar or station-specific.
<code>min.month.values, min.year.values</code>	Minimum complete paired positions for copied-month and copied-year comparisons.
<code>run</code>	Minimum flat-line window length in observations.
<code>tolerance</code>	Maximum within-window temperature range.
<code>group</code>	Calendar grouping. "month_hour" uses month and integer hour of day.
<code>window</code>	Circular calendar-day window width.
<code>z</code>	Absolute robust standardized climatology or first-difference threshold.
<code>min.samples</code>	Minimum climatological or first-difference reference sample.
<code>max.rise, max.fall</code>	Optional absolute consecutive-change limits.

threshold	Absolute difference from both adjacent values required by the isolated spike/dip test.
metadata	NULL, or station metadata. When supplied, the selected identifier, longitude, and latitude fields are required.
station.id	Name of the metadata station identifier column.
coords	Names of numeric longitude and latitude metadata columns in decimal degrees, in that order. Individual entries may be NA; non-missing values are range-checked.
elevation	NULL, or the name of an optional numeric metadata column containing elevation in metres.
elevation.scale	Positive elevation-decay scale in metres used to penalize neighbours with large height differences.
n.neighbours	Maximum selected spatial-regression neighbours.
max.distance	Maximum target-neighbour distance in km.
min.neighbours	Minimum simultaneous valid regression estimates.
min.overlap	Minimum overall paired observations for neighbour selection.
min.group.overlap	Minimum paired observations in a model group.
min.correlation	Minimum target-neighbour Pearson correlation.
f	Standardized residual threshold for spatial regression.
min.difference	Absolute spatial residual threshold.
min.se	Positive lower bound on each regression standard error.
tmin, tmax	Numeric zoo objects with identical time indices and station columns containing paired minimum and maximum temperatures.
lagged	Whether to apply previous/current/following-day logical tests.
max.range	Maximum allowed lagged Tmin–Tmax separation.
min.years	Minimum complete annual indicators for breakpoint testing.
alpha	Holm-adjusted breakpoint significance level.
min.shift	Minimum absolute pre/post median change in degrees Celsius.
min.completeness	Minimum daily completeness of an annual indicator.
indicators	Unique subset of "mean", "sd", "minimum", and "maximum".

Details

tempQC_range is a gross-error test using the world-record screening limits compiled by Durre et al. (2010). Limits are intentionally configurable.

tempQC_duplicate compares corresponding calendar positions and flags complete months or years that are copied exactly. Durre et al. (2010) identify such duplication as a keying, transmission, or processing failure.

tempQC_persistence moves a window through each station and flags every observation in windows whose range does not exceed tolerance. Exact seven-day flat lines follow Feng et al. (2004); the adjustable tolerance and duration support sub-daily sensor tests such as Beele et al. (2022).

tempQC_climatology calculates a resistant median and MAD reference. Daily tests use a circular day-of-year window; sub-daily tests can preserve the seasonal and diurnal cycles with month-hour groups. The six-score default is based on Durre et al. (2010), with the resistant formulation motivated by Feng et al. (2004).

tempQC_step robustly standardizes consecutive changes within calendar groups and can also apply absolute rise/fall limits. This is the seasonal rate-of-change structure of Hubbard et al. (2007), extended to hour-specific groups following Cheng et al. (2016).

tempQC_spike requires the centre value to differ from both neighbours by at least threshold in the same direction. Its daily default is the 25-degree spike/dip rule of Durre et al. (2010).

tempQC_spatial fits target-on-neighbour linear regressions separately by calendar group, retains the lowest proximity-adjusted-error eligible neighbours, and combines their predictions with geographic/elevation proximity and inverse-error-variance weights. The pairwise regression intercept estimates systematic station offsets, including persistent height effects, without imposing a fixed lapse rate. Missing target coordinates trigger correlation/error-based selection; for a located target, eligible located neighbours rank ahead of unlocated fallbacks. A flag requires both an absolute and a standardized residual. The equations follow the spatial regression test of Hubbard et al. (2007) and Estevez et al. (2018); grouping and correlation screening incorporate findings of Cheng et al. (2016) and Xiong et al. (2022, 2024).

tempQC_internal flags Tmin above Tmax, possible interchange relative to the adjacent three-day window, and the 40-degree lagged range conditions in Durre et al. (2010). It returns separate flags because the erroneous member of an inconsistent pair is not identifiable automatically.

tempQC_breakpoint aggregates sub-daily data to daily means, calculates complete annual indicators, applies Pettitt rank tests, and adjusts p-values across indicators. Boulanger et al. (2010) and Brunet et al. (2020) motivate reporting homogeneity separately from point QC.

Value

The range, duplicate, persistence, climatology, step, and spike tests return a logical zoo object matching `x`. tempQC_spatial returns a list with logical flags, numeric scores, numeric estimate, and selected neighbours. tempQC_internal returns a list of Tmin and Tmax flag objects. tempQC_breakpoint returns one data.frame row per station.

References

- Beele, E. et al. (2022). Quality control and correction method for air temperature data from a citizen science weather station network. *Earth System Science Data*, 14, 4681–4717.
- Boulanger, J.-P. et al. (2010). A procedure for automated quality control and homogenization of historical daily temperature and precipitation data. *Climatic Change*, 98, 471–491.
- Brunet, M. et al. (2020). *Best Practice Guidelines for Climate Data and Metadata Formatting, Quality Control and Submission*. Copernicus Climate Change Service.
- Cheng, A. R. et al. (2016). Quality control program for real-time hourly temperature observation in Taiwan. *Journal of Atmospheric and Oceanic Technology*, 33, 953–965.

Durre, I. et al. (2010). Comprehensive automated quality assurance of daily surface observations. *Journal of Applied Meteorology and Climatology*, 49, 1615–1633.

Estevez, J. et al. (2018). Spatial regression test for ensuring temperature data quality in southern Spain. *Theoretical and Applied Climatology*, 131, 309–318.

Feng, S., Hu, Q. and Qian, W. (2004). Quality control of daily meteorological data in China, 1951–2000. *International Journal of Climatology*, 24, 853–870.

Hubbard, K. G. et al. (2007). An improved QC process for temperature in the daily cooperative weather observations. *Journal of Atmospheric and Oceanic Technology*, 24, 206–213.

Xiong, X. et al. (2022). Research on quality control methods for surface temperature observations via spatial correlation analysis. *International Journal of Climatology*. doi:10.1002/joc.7897.

Xiong, X. et al. (2024). Research on quality control method of surface temperature observations for complex physical geography. *Journal of Atmospheric and Oceanic Technology*, 41, 803–815.

See Also

[tempQC_daily](#), [tempQC_subdaily](#)

Examples

```
dates <- as.Date("2020-01-01") + 0:9
x <- zoo::zoo(cbind(A=c(rep(10, 7), 11, 12, 13)), dates)
tempQC_persistence(x)
tempQC_range(x)
```

time2season	<i>Date/DateTime character -> Seasonal character</i>
-------------	---

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

When type=SONDJFM -) SONDJFM: September, October, November, December, January, February, March, with January, February and March belonging to the next calendar year. Leading January, February and March values before the first September season are returned as NA. Other months are returned as NA.

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") or "SONDJFM" when type="SONDJFM" -) months => c("DJF", "MAM", "JJA", "SON") or c("DJFM", "AM", "JJAS", "ON") or "SONDJFM"
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 -) SONDJFM => "SONDJFM"= Sep, Oct, Nov, Dec, Jan, Feb, Mar; other months are returned as NA

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")

## September to March selector
time2season(t, type="SONDJFM")
```

vector2zoo	<i>Vector -> Zoo</i>
------------	-------------------------

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 <i>dates</i> , 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).
---	---

<code>FUN</code>	Function that will be applied to ALL the values in <code>x</code> belonging to each one of the 12 months of the year (e.g., <code>FUN</code> can be some of <code>mean</code> , <code>sum</code> , <code>max</code> , <code>min</code> , <code>sd</code>).
<code>na.rm</code>	Logical. Should missing values be removed? -) <code>TRUE</code> : the monthly values and <code>FUN</code> are computed considering only those values in <code>x</code> different from <code>NA</code> -) <code>FALSE</code>: if there is AT LEAST one <code>NA</code> within a month, the corresponding monthly value will be <code>NA</code>
<code>na.rm.max</code>	Numeric in <code>[0, 1]</code> . 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 <code>na.rm.max</code> the corresponding weekly value will be <code>NA</code> .
<code>start</code>	character, indicating the starting time used for aggregating sub-daily time series into daily ones. It MUST be provided in the format specified by <code>start.fmt</code>. This value is used to define the time when a new day begins (e.g., for some rain gauge stations). -) All the values of <code>x</code> with a time attribute before <code>start</code> are considered as belonging to the day before the one indicated in the time attribute of those values. -) All the values of <code>x</code> with a time attribute equal to <code>start</code> are considered to be equal to "<code>00:00:00</code>" in the output zoo object. -) All the values of <code>x</code> with a time attribute after <code>start</code> 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 "<code>00:00:00</code>". Use with caution. See examples.
<code>start.fmt</code>	character indicating the format in which the time is provided in <code>start</code> , By default <code>date.fmt=%H:%M:%S</code> . See format in as.POSIXct .
<code>tz</code>	character, with the specification of the time zone used in both <code>x</code> and <code>start</code>. System-specific (see time zones), but <code>"</code> is the current time zone, and <code>"GMT"</code> is UTC (Universal Time, Coordinated). See Sys.timezone and as.POSIXct. If <code>tz</code> is missing (the default), it is automatically set to the time zone used in <code>time(x)</code>. This argument can be used to force using the local time zone or any other time zone instead of UTC as time zone.
<code>dates</code>	It is only used when <code>x</code> is not a zoo object. numeric, factor, Date indicating how to obtain the dates. If <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates If <code>dates</code> is a factor, it is converted into 'Date' class, using the date format specified by <code>date.fmt</code> If <code>dates</code> is already of Date class, the code verifies that the number of days in <code>dates</code> be equal to the number of elements in <code>x</code>
<code>date.fmt</code>	It is only used when <code>x</code> is not a zoo object. character indicating the format in which the dates are stored in <code>dates</code>, e.g. <code>%Y-%m-%d</code>. See format in as.Date. ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code>.

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

- * **datasets**
 - Cauquenes7336001, [23](#)
 - EbroPPtsMonthly, [47](#)
 - KarameaAtGorgeQts, [76](#)
 - MaquehueTemuco, [78](#)
 - OcaEn0naQts, [87](#)
 - SanMartinoPPts, [111](#)
- * **graphs**
 - baseflow, [17](#)
 - calendarHeatmap, [20](#)
 - fdc, [50](#)
 - fdcu, [53](#)
 - hydropairs, [59](#)
 - hydroplot, [60](#)
 - matrixplot, [79](#)
 - plot_pq, [87](#)
- * **hplot**
 - precipQC-class, [93](#)
- * **manip**
 - (sub)daily2annual, [6](#)
 - (sub)daily2monthly, [10](#)
 - annualfunction, [14](#)
 - baseflow, [17](#)
 - change_tz, [24](#)
 - climograph, [25](#)
 - cmv, [30](#)
 - daily2weekly, [34](#)
 - dip, [38](#)
 - diy, [39](#)
 - dm2seasonal, [40](#)
 - dwdays, [44](#)
 - dwi, [45](#)
 - extract, [48](#)
 - fdc, [50](#)
 - fdcu, [53](#)
 - hip, [57](#)
 - hydroplot, [60](#)
 - infillxy, [65](#)
 - isComplete, [66](#)
 - istdx, [70](#)
 - izoo2rzoo, [72](#)
 - ma, [77](#)
 - mip, [81](#)
 - monthlyfunction, [82](#)
 - precipQC, [92](#)
 - precipQC_daily, [94](#)
 - precipQC_subdaily, [100](#)
 - precipQC_tests, [105](#)
 - rm1stchar, [110](#)
 - seasonalfunction, [112](#)
 - sfreq, [114](#)
 - shifteyears, [116](#)
 - si, [117](#)
 - smry, [119](#)
 - sname2ts, [121](#)
 - spi, [123](#)
 - stdx, [126](#)
 - subdaily2daily, [142](#)
 - subdaily2weekly, [145](#)
 - subhourly2hourly, [148](#)
 - subhourly2nhourly, [151](#)
 - subhourly2nminutes, [153](#)
 - time2season, [170](#)
 - vector2zoo, [172](#)
 - weeklyfunction, [173](#)
 - yip, [176](#)
 - zoo2RHtest, [177](#)
- * **math**
 - hydropairs, [59](#)
- * **package**
 - hydroTSM-package, [3](#)
- * **ts**
 - change_tz, [24](#)
 - precipQC, [92](#)
 - precipQC-class, [93](#)
 - precipQC_daily, [94](#)
 - precipQC_subdaily, [100](#)
 - precipQC_tests, [105](#)

- spi, 123
- (sub)daily2annual, 6
- (sub)daily2monthly, 10
- annualfunction, 4, 9, 14, 83, 113, 175
- as.Date, 8, 9, 12, 13, 15, 26, 32, 35, 36, 38, 41, 43, 45, 46, 58, 63, 81, 83, 113, 117, 122, 144, 147, 150, 152, 155, 174, 176
- as.POSIXct, 18, 25, 32, 58, 63, 67, 73, 74, 89, 144, 145, 147, 149, 150, 152–155, 174
- Axis, 51, 55, 56
- baseflow, 5, 17
- calendarHeatmap, 5, 20
- Cauquenes7336001, 4, 23
- change_tz, 24
- classIntervals, 79, 80
- climograph, 5, 25, 91, 116
- cmv, 5, 13, 30, 36, 116, 147
- colorRampPalette, 21, 22
- cor, 59, 60
- cor.test, 59
- daily2annual, 4, 13, 16, 33, 36, 42, 49, 83, 116, 147, 175
- daily2annual ((sub)daily2annual), 6
- daily2monthly, 4, 9, 33, 36, 42, 49, 83, 147, 175
- daily2monthly ((sub)daily2monthly), 10
- daily2weekly, 4, 34, 86
- dip, 5, 38, 40, 58, 81, 171, 172, 177
- diy, 5, 39, 39, 58, 81, 177
- dm2seasonal, 4, 40, 83, 113, 145, 150, 155, 171, 175
- drawTimeAxis, 43
- drawxaxis, 5
- drawxaxis (drawTimeAxis), 43
- dwdays, 5, 44
- dwi, 5, 33, 45, 80
- EbroPPtsMonthly, 4, 47
- extract, 5, 42, 48, 113, 171
- extractzoo (extract), 48
- fdc, 5, 19, 50, 57, 91
- fdcu, 5, 19, 53, 53, 91
- fivenum, 121
- frequency, 115
- hip, 5, 39, 40, 57, 81
- hydropairs, 5, 59
- hydroplot, 5, 9, 13, 19, 36, 42, 60, 91, 145, 147, 150, 153, 155
- hydroTSM (hydroTSM-package), 3
- hydroTSM-package, 3
- infillxy, 5, 65
- IQR, 121
- is.regular, 144
- isComplete, 66, 86
- istdx, 5, 70, 127
- izoo2rzoo, 5, 13, 33, 36, 72, 145, 147, 150, 153, 155, 172
- KarameaAtGorgeQts, 4, 76
- kurtosis, 121
- legend, 52, 56, 89
- levelplot, 22, 80
- ma, 5, 77
- MaquehueTemuco, 4, 78
- matrixplot, 5, 47, 79
- message, 125
- mip, 5, 39, 40, 58, 81, 171, 172, 177
- monthly2annual, 4, 9, 16, 42
- monthly2annual ((sub)daily2annual), 6
- monthlyfunction, 4, 13, 16, 29, 33, 82, 91, 113, 145, 150, 155
- na.approx, 18, 73, 88
- na.spline, 18, 73, 88
- ndays (nperiods), 85
- nhours (nperiods), 85
- nmonths (nperiods), 85
- nperiods, 85
- nweeks (nperiods), 85
- nyears (nperiods), 85
- OcaEnOnaQts, 4, 87
- OlsonNames, 24, 25
- optim, 125
- pairs, 59, 60
- par, 43, 51, 52, 56, 63
- periodicity, 114, 115
- plot, 51, 55, 62, 94

- `plot.default`, [18](#), [51](#), [52](#), [55](#), [56](#), [63](#), [89](#)
- `plot.precipQC`, [93](#), [99](#), [104](#), [110](#)
- `plot.precipQC (precipQC-class)`, [93](#)
- `plot.streamflowQC (streamflowQC-class)`, [129](#)
- `plot.tempQC`, [162](#), [166](#)
- `plot.tempQC (tempQC-class)`, [157](#)
- `plot_pq`, [5](#), [19](#), [87](#)
- `points`, [51](#), [56](#)
- `polygon`, [56](#)
- `precipQC`, [92](#), [93](#), [94](#), [99](#), [104](#), [128](#), [157](#)
- `precipQC-class`, [93](#)
- `precipQC_accumulation`, [104](#)
- `precipQC_accumulation (precipQC_tests)`, [105](#)
- `precipQC_breakpoint (precipQC_tests)`, [105](#)
- `precipQC_climatology (precipQC_tests)`, [105](#)
- `precipQC_daily`, [92](#), [93](#), [94](#), [94](#), [101](#), [102](#), [104](#), [105](#), [110](#)
- `precipQC_dryspell (precipQC_tests)`, [105](#)
- `precipQC_duplicate (precipQC_tests)`, [105](#)
- `precipQC_frequency (precipQC_tests)`, [105](#)
- `precipQC_gap (precipQC_tests)`, [105](#)
- `precipQC_persistence`, [104](#)
- `precipQC_persistence (precipQC_tests)`, [105](#)
- `precipQC_range`, [94](#), [99](#)
- `precipQC_range (precipQC_tests)`, [105](#)
- `precipQC_spatial`, [99](#), [104](#)
- `precipQC_spatial (precipQC_tests)`, [105](#)
- `precipQC_subdaily`, [92–94](#), [99](#), [100](#), [105](#), [110](#)
- `precipQC_tests`, [105](#)
- `precipQC_weekday`, [99](#), [103](#), [104](#)
- `precipQC_weekday (precipQC_tests)`, [105](#)
- `print.precipQC (precipQC-class)`, [93](#)
- `print.streamflowQC (streamflowQC-class)`, [129](#)
- `print.tempQC (tempQC-class)`, [157](#)
- `rm1stchar`, [5](#), [110](#)
- `SanMartinoPpts`, [4](#), [111](#)
- `scale`, [71](#), [127](#)
- `sd`, [121](#)
- `seasonalfunction`, [4](#), [42](#), [49](#), [83](#), [112](#), [145](#), [150](#), [155](#), [171](#), [175](#)
- `sfreq`, [5](#), [92](#), [93](#), [114](#), [126](#), [156](#)
- `shiftyears`, [116](#)
- `si`, [5](#), [117](#)
- `skewness`, [121](#)
- `smry`, [5](#), [119](#)
- `sname2plot`, [5](#), [122](#)
- `sname2plot (hydroplot)`, [60](#)
- `sname2ts`, [5](#), [64](#), [121](#)
- `spei`, [5](#)
- `spei (spi)`, [123](#)
- `spi`, [5](#), [123](#)
- `stdx`, [5](#), [71](#), [126](#)
- `streamflowQC`, [127](#), [129](#), [134](#), [138](#)
- `streamflowQC-class`, [129](#)
- `streamflowQC_breakpoint (streamflowQC_tests)`, [138](#)
- `streamflowQC_climatology (streamflowQC_tests)`, [138](#)
- `streamflowQC_daily`, [127–129](#), [130](#), [130](#), [138](#), [142](#)
- `streamflowQC_duplicate (streamflowQC_tests)`, [138](#)
- `streamflowQC_flatline (streamflowQC_tests)`, [138](#)
- `streamflowQC_fluctuation (streamflowQC_tests)`, [138](#)
- `streamflowQC_gap (streamflowQC_tests)`, [138](#)
- `streamflowQC_highflow (streamflowQC_tests)`, [138](#)
- `streamflowQC_range (streamflowQC_tests)`, [138](#)
- `streamflowQC_rate (streamflowQC_tests)`, [138](#)
- `streamflowQC_spatial (streamflowQC_tests)`, [138](#)
- `streamflowQC_spike (streamflowQC_tests)`, [138](#)
- `streamflowQC_subdaily`, [127–130](#), [134](#), [134](#), [138](#), [142](#)
- `streamflowQC_tests`, [128](#), [130](#), [134](#), [138](#), [138](#)
- `strptime`, [17](#), [21](#), [73](#), [89](#), [172](#)
- `subdaily2annual`, [4](#), [145](#), [150](#), [153](#), [155](#)
- `subdaily2annual ((sub)daily2annual)`, [6](#)
- `subdaily2daily`, [4](#), [13](#), [25](#), [33](#), [36](#), [118](#), [142](#), [147](#), [150](#), [153](#), [155](#)
- `subdaily2monthly`, [4](#), [145](#), [150](#), [153](#), [155](#)

subdaily2monthly ((sub)daily2monthly),
 10
 subdaily2seasonal, 4, 145, 150, 153, 155
 subdaily2seasonal (dm2seasonal), 40
 subdaily2weekly, 4, 145
 subhourly2hourly, 4, 9, 13, 25, 33, 36, 145,
 147, 148, 153, 155
 subhourly2nhourly, 4, 25, 104, 151
 subhourly2nminutes, 4, 150, 153, 153
 submonthly2monthly, 4
 submonthly2monthly
 ((sub)daily2monthly), 10
 substr, 111
 summary, 121
 Sys.timezone, 18, 32, 58, 63, 67, 73, 74, 89,
 144, 147, 149, 152, 154, 174

 tempQC, 128, 156, 157, 162, 166
 tempQC-class, 157
 tempQC_breakpoint (tempQC_tests), 166
 tempQC_climatology (tempQC_tests), 166
 tempQC_daily, 156, 157, 158, 158, 164–166,
 170
 tempQC_duplicate (tempQC_tests), 166
 tempQC_internal, 162
 tempQC_internal (tempQC_tests), 166
 tempQC_persistence (tempQC_tests), 166
 tempQC_range (tempQC_tests), 166
 tempQC_spatial (tempQC_tests), 166
 tempQC_spike (tempQC_tests), 166
 tempQC_step (tempQC_tests), 166
 tempQC_subdaily, 156–158, 162, 163, 166,
 170
 tempQC_tests, 157, 158, 162, 166, 166
 time2season, 5, 42, 49, 113, 170
 timeBasedSeq, 58
 title, 51, 55, 89

 vector2zoo, 5, 9, 13, 36, 74, 145, 147, 150,
 153, 155, 172

 weeklyfunction, 4, 36, 147, 173
 window, 68

 yip, 5, 16, 39, 40, 58, 81, 172, 176

 zoo, 25, 74, 85, 86, 99, 172
 zoo2RHtest, 5, 177