

# Package ‘hydroTSM’

January 23, 2014

**Type** Package

**Title** Time series management, analysis and interpolation for hydrological modelling

**Version** 0.4-2-1

**Date** 2014-01-22

**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 (>= 2.10.0), zoo (>= 1.7-2), xts (>= 0.8-2)

**Imports** e1071, gstat, automap, sp

**Suggests** rgdal, maptools, lattice

**URL** <http://www.rforge.net/hydroTSM/>,  
<http://cran.r-project.org/web/packages/hydroTSM/>

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

**BugReports** <https://www.rforge.net/bugzilla/describecomponents.cgi?product=hydroTSM>

**Keywords** hydrology, hydrological modelling, hydrologic modeling, time series

**LazyLoad** yes

**Author** Mauricio Zambrano-Bigiarini [aut, cre, cph]

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-01-23 00:18:53

**R topics documented:**

hydroTSM-package	3
(sub)daily2annual	4
(sub)daily2monthly	7
annualfunction	9
dip	11
diy	13
dm2seasonal	14
drawTimeAxis	16
dwdays	18
dwi	19
EbroCatchmentsCHE	21
EbroDEM1000m	22
EbroPPgis	23
EbroPPtsMonthly	23
extract	24
fdc	25
fdcu	29
gists2spt	33
hip	35
hydrokrige	36
hydropairs	45
hydroplot	46
hypsometric	50
infillxy	52
istdx	53
izoo2rzoo	54
KarameaAtGorgeQts	56
ma	57
matrixplot	58
mip	59
monthlyfunction	60
mspplot	63
OcaEnOnaQts	67
rm1stchar	68
SanMartinoPPts	69
seasonalfunction	69
sfreq	72
smry	73
sname2ts	75
stdx	76
subdaily2daily	77
time2season	79
vector2zoo	80
yip	81
zoo2RHtest	82

---

hydroTSM-package	<i>Management, analysis, interpolation 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: 0.4-2-1  
Date: 2014-01-22  
License: GPL >= 2  
LazyLoad: yes  
Packaged: Wed Jan 22 19:09:48 CLST 2014; MZB  
BuiltUnder: R version 3.0.2 (2013-09-25) – "Frisbee Sailing" ; x86\_64-pc-linux-gnu (64-bit)

## Author(s)

Mauricio Zambrano-Bigiarini

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

## See Also

<http://www.rforge.net/hydroGOF/>.

<http://www.rforge.net/hydroPSO/>.

[http://rwiki.sciviews.org/doku.php?id=guides:tutorials:hydrological\\_data\\_analysis](http://rwiki.sciviews.org/doku.php?id=guides:tutorials:hydrological_data_analysis)

## 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", dates=1, pfreq="ma")

#####
## Ex4) IDW interpolation and plot

## Loading the spatial data corresponding to 'EbroPPtsMonthly'
data(EbroPPgis)

## Loading the shapefile (polygon) with the subcatchments
data(EbroCatchmentsCHE)

## Selecting the first day of 'EbroPPtsMonthly' for all the stations
x.ts <- as.numeric(EbroPPtsMonthly[1, 2:ncol(EbroPPtsMonthly)])

## Setting the name of the gauging stations
names(x.ts) <- colnames(EbroPPtsMonthly[1,2:ncol(EbroPPtsMonthly)])

# Computing the interpolated values and plotting them
# Probably you will need to resize your window
## Not run:
x.idw <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                   X="EAST_ED50" , Y="NORTH_ED50" , sname="ID",
                   bname= "CHE_BASIN_NAME", elevation="ELEVATION",
                   type= "both",
                   subcatchments= EbroCatchmentsCHE,
                   cell.size= 1000)

## End(Not run)

#####
## 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)

```

**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, out.fmt = "%Y", ...)

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

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

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

**Arguments**

x	zoo, xts, 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 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 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).
na.rm	Logical. Should missing values be removed? -) TRUE : the monthly and annual values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA within a year, the corresponding annual value is NA.
out.fmt	Character indicating the date format for the output time series. See format in <a href="#">as.Date</a> . 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 for corresponding to each gauging station

	<p>If <code>dates</code> is a number (default), it indicates the index of the column in <code>x</code> that stores the dates</p> <p>If <code>dates</code> is a factor, it is converted into <code>Date</code> class, using the date format specified by <code>date.fmt</code></p> <p>If <code>dates</code> is already of <code>Date</code> class, the code verifies that the number of days on it be equal to the number of element in <code>x</code></p>
<code>date.fmt</code>	<p>character indicating the format in which the dates are stored in <code>dates</code>, e.g. <code>%Y-%m-%d</code>. See format in <a href="#">as.Date</a>.</p> <p>ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code>.</p>
<code>out.type</code>	<p>Character that defines the desired type of output. Valid values are:</p> <ul style="list-style-type: none"> <li>-) <code>data.frame</code>: a <code>data.frame</code>, with as many columns as stations are included in <code>x</code>, and row names indicating the Year</li> <li>-) <code>db</code>: a <code>data.frame</code>, with 3 columns will be produced.</li> </ul> <p>The first column (<code>StationID</code>) will store the ID of the station</p> <p>The second column (<code>Year</code>) will store the year,</p> <p>The third column (<code>Value</code>) will contain the annual value corresponding to the two previous columns.</p>
<code>verbose</code>	logical; if <code>TRUE</code> , progress messages are printed
<code>...</code>	further arguments passed to or from other methods.

**Value**

a zoo object with annual frequency

**Author(s)**

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

**See Also**

[daily2monthly](#), [monthly2annual](#), [hydroplot](#), [annualfunction](#), [vector2zoo](#), [as.Date](#)

**Examples**

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

## Daily to Annual
daily2annual(x, FUN=sum, na.rm=TRUE)

#####
## Ex2: Monthly to Annual (same result as )
m <- daily2monthly(x, FUN=sum, na.rm=TRUE)
monthly2annual(m, FUN=sum, na.rm=TRUE)

#####
```

```
## Ex3: 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)

#####
## Ex4: Loading the monthly time series of precipitation within the Ebro River basin
data(EbroPPtsMonthly)

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

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

---

(sub)daily2monthly      (sub)Daily -> Monthly

---

## Description

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

## Usage

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

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

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

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

## Arguments

x                      zoo, xts, data.frame or matrix object, with 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* 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 have to be applied for transforming from daily to annual 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 monthly and annual values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA within a year, the monthly values are 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 <i>x</i> that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by <i>date.fmt</i> 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 <i>x</i>
date.fmt	character indicating the format in which the dates are stored in <i>dates</i> , e.g. %Y-%m-%d. See format in <a href="#">as.Date</a> . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
out.type	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 <i>x</i> , 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.
out.fmt	OPTIONAL. Only used when <i>x</i> 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> .
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**

[daily2annual](#), [subdaily2daily](#), [monthlyfunction](#), [hydroplot](#), [vector2zoo](#), [izoo2rzoo](#), [as.Date](#)

**Examples**

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

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

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

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

---

annualfunction

*Annual Function*


---

**Description**

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

**Usage**

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

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

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

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

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

**Arguments**

x	zoo, xts, data.frame or matrix object, with daily/monthly/annual time series. Measurements at several gauging stations can be stored in a data.frame 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 dates, e.g. %Y-%m-%d. See format in <a href="#">as.Date</a> . 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="nubr" )

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

## End(Not run)

```

**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 <a href="#">as.Date</a> . ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	Character indicating the type of result that is given by this function. Valid values are: 1) seq : a vector of class Date with all the days between the two dates, OR 2) nmbr: a single numeric value with the amount of days between the two dates.

**Value**

Depending on the value of out.type, it returns:

- 1) a vector of class Date with all the days between from and to (both of them included), OR
- 2) the amount of days between the two dates

**Author(s)**

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

**See Also**

[mip](#), [yip](#), [hip](#), [diy](#)

**Examples**

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

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

---

diy	<i>Days in Year</i>
-----	---------------------

---

### Description

Given a single numeric value representing a year, it generates:

- 1) a vector of dates with all the days within the year, OR
- 2) the amount of days in the specified year

### Usage

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

### Arguments

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

---

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

---

## Description

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

## Usage

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

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

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

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

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

## Arguments

x	zoo, xts, data.frame or matrix object, with sub-daily, daily, weekly or monthly time series. Measurements at several gauging stations can be stored in a data.frame 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).
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). <b>The FUN value for the winter season (DJF or DJFM) is computed considering the consecutive months of December, January and February/March.</b> See 'Note' section.
na.rm	Logical. Should missing values be removed? -) TRUE : the seasonal values are computed considering only those values different from NA ( <b>very important when FUN=sum</b> ) -) 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 <a href="#">as.Date</a> . Possible values are: -) %Y : only the year will be used for the time. Default option. (e.g., "1961" "1962"...) -) %Y-%m-%d: a complete date format will be used for the time. (e.g., "1961-01-01" "1962-01-01"...)
dates	numeric, factor or Date object indicating how to obtain the dates. If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into Date class, by using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days on it be equal to the number of elements in x
date.fmt	Character indicating the format in which the dates are stored in dates, e.g. %Y-%m-%d. See format in <a href="#">as.Date</a> . ONLY required when class(dates)="factor" or class(dates)="numeric".
out.type	Character that defines the desired type of output. Valid values are: -) data.frame: a data.frame, with as many columns as stations are included in x, the year corresponding to each seasonal value are used as row names. -) db : a data.frame, with 4 columns will be produced. The first column (StationID) stores the ID of the station The second column (Year) stores the year, The third column (Season) stores the season, The fourth column (Value) contains the seasonal value corresponding to the values specified in the previous three columns
...	further arguments passed to or from other methods.

**Value**

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

**Warning**

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

**Note**

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

**Author(s)**

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

**See Also**

, [hydroplot](#), [seasonalfunction](#), [time2season](#), [extract](#), [daily2monthly](#), [daily2annual](#), [monthly2annual](#)

**Examples**

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

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

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

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

---

drawTimeAxis

*Customized Time Axis*

---

**Description**

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

**Usage**

```
drawxaxis(x, tick.tstep = "auto", lab.tstep = "auto",
          lab.fmt=NULL, cex.axis=1, ...)
```

**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 <a href="#">as.Date</a> . 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 <a href="#">par</a> ).
...	further arguments passed to the axis function or from other methods.

**Note**

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

**Author(s)**

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

**Examples**

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

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

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

---

dwdays

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


---

### Description

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

### Usage

```
dwdays(x, ...)
```

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

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

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

### Arguments

x	zoo, data.frame or matrix object, usually with daily time series of precipitation. Measurements at several gauging stations can be stored in a data.frame 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 thr. Valid values are: wet, dry.
na.rm	Logical. Should missing values be removed before counting?
dates	numeric, factor or Date object indicating how to obtain the dates If dates is a number (default), it indicates the index of the column in x that stores the dates If dates is a factor, it is converted into Date class, using the date format specified by date.fmt If dates is already of Date class, the code verifies that the number of days in dates be equal to the number of element in x

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

verbose        logical; if TRUE, progress messages are printed

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

**Author(s)**

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

**Examples**

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

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

---

dwi

*Days with Information*


---

**Description**

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

**Usage**

```
dwi(x, ...)
```

## Default S3 method:

```
dwi(x, out.unit = "years", from = start(x), to = end(x),
    date.fmt = "%Y-%m-%d", timestep="days", ...)
```

## S3 method for class 'zoo'

```
dwi(x, out.unit = "years", from = start(x), to = end(x),
    date.fmt = "%Y-%m-%d", timestep="days", ...)
```

## S3 method for class 'data.frame'

```
dwi(x, out.unit = "years", from, to, date.fmt = "%Y-%m-%d", timestep="days",
    dates = 1, verbose = TRUE, ...)
```

## S3 method for class 'matrix'

```
dwi(x, out.unit = "years", from, to, date.fmt = "%Y-%m-%d", timestep="days",
    dates = 1, verbose = TRUE, ...)
```

**Arguments**

<code>x</code>	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 <code>x</code> represent the time series measured in each gauging station, and the column names of <code>x</code> have to correspond to the ID of each station (starting by a letter).
<code>out.unit</code>	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)
<code>from</code>	Character indicating the starting date for the computations. It has to be in the format indicated by <code>date.fmt</code> . When <code>x</code> is a data.frame and this value is not provided, the date corresponding to the first row of <code>x</code> is used
<code>to</code>	Character indicating the ending date for the computations. It has to be in the format indicated by <code>date.fmt</code> . When <code>x</code> is a data.frame and this value is not provided, the date corresponding to the last row of <code>x</code> is used
<code>date.fmt</code>	character indicating the format in which the dates are stored in <i>dates</i> , e.g. %Y-%m-%d. See format in <a href="#">as.Date</a> . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .
<code>tstep</code>	Time step used for storing the values in <code>x</code> . 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.
<code>dates</code>	numeric, factor or Date object indicating how to obtain the dates for each column of <code>x</code> If <code>dates</code> is a number, 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>verbose</code>	logical; if TRUE, progress messages are printed
<code>...</code>	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

## Days with information per year
dwi(x)

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

#####
## 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)

```

---

EbroCatchmentsCHE      *Subcatchments of the Ebro River basin (Spain)*

---

**Description**

Shapefile with 57 subcatchments of the Ebro River basin (Spain)

**Usage**

```
data(EbroCatchmentsCHE)
```

**Format**

[SpatialPolygonsDataFrame-class](#).

The fields stored in the @data slot of this object are:

- ) *CUECHE*: sequential counter
- ) *CUECHE\_ID*: ID of each subcatchment
- ) *NOMBRE*: name of each subcatchment
- ) *AREA\_KM2*: are of eac subcatchment, [km2]

**Details**

Projection: European Datum 50, Zone 30N.

**Note**

A small subcatchment of 0.3 km<sup>2</sup> was removed of the original shapefile, due to the fact that it did not have name and it was too small for the purposes of this dataset.

**Source**

Downloaded ('Divisorias Hidrograficas 1:50.000 - CHE') from the web site of the Confederacion Hidrografica del Ebro (CHE) <http://oph.chebro.es/ContenidoCartoHidrologia.htm>. Last accessed [March 2008].

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.

---

EbroDEM1000m

*Digital Elevation Model (DEM) of the Ebro River basin (Spain)*

---

**Description**

Digital Elevation Model of the Ebro River basin (Spain), with cells of 1000x1000 [m]

**Usage**

data(EbroDEM1000m)

**Format**

'SpatialGridDataFrame'

**Details**

Digital elevation model of the Ebro River basin, with 1000x1000m of spatial resolution.

**Source**

The original file (Modelo Digital MDT200 RASTER) was downloaded from the web site of the Confederacion Hidrografica del Ebro (CHE) <http://oph.chebro.es/ContenidoCartoApoyo.htm> (last accessed [March 2010]), and then resampled up to 1000x1000m of resolution.

This dataset is intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

EbroPPgis

*Spatial location of raingauges on the Ebro River basin***Description**

Spatial location of the 349 raingauges with daily precipitation on the Ebro River Basin (dataset 'EbroPPts'), Spain, with more than 70% of days with information (without missing values)

**Usage**

```
data(EbroPPgis)
```

**Format**

A data.frame with seven fields:

- \*) 'ID' : identifier of each gauging station.
- \*) 'STATION\_NAME' : name of the gauging station.
- \*) 'EAST\_ED50' : easting coordinate of the gauging station. European Datum 50, Zone 30 North.
- \*) 'NORTH\_ED50' : northing coordinate of the gauging station. European Datum 50, Zone 30 North.
- \*) 'ELEVATION' : elevation of the gauging station, [m a.s.l.].
- \*) 'CHE\_BASIN\_ID' : identifier of the subbasin in which the gauging station s located.
- \*) 'CHE\_BASIN\_NAME' : name of the subbasin in which the gauging station s located.

**Details**

Projection: European Datum 50, Zone 30N.

**Source**

Downloaded ('Red de Control Meteorologico') from the web site of the Confederacion Hidrografica del Ebro (CHE) <http://oph.chebro.es/ContenidoCartoClimatologia.htm>. Last accessed [March 2008], and then the name of 7 selected fields were translated into English language.

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.

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://oph.chebro.es/DOCUMENTACION/PrecipitacionMensualRelleno/PrecipitacionMensualRelleno.html>. 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 from 1 to 12: trgt is considered as a month (1=JAN, 2=FEB,....., 12=DEC), and all the values in x belonging to the month specified by trgt will be extracted. 2) integer > 12: trgt is considered as a year, and all the values in x belonging

to the year specified by `trgt` will be extracted  
3) character: `trgt` is considered as a weather season, and all the values in `x` belonging to the season specified by `trgt` will be extracted. Valid values are:

- ) DJF : December, January, February
- ) MAM : March, April, May
- ) JJA : June, July, August
- ) SON : September, October, November
- ) DJFM: December, January, February, March
- ) AM : April, May
- ) JJAS: June, July, August, September
- ) ON : October, November

... further arguments passed to or from other methods

### Value

a zoo object with the extracted values.

### Author(s)

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

### See Also

[time2season](#), [seasonalfunction](#), [daily2annual](#), [daily2monthly](#)

### Examples

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

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

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

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

### 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", 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",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", 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", 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.<br>Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of x represent the time series measured in each gauging station, and the column names of x have to correspond to the ID of each station (starting by a letter). When x is a matrix or data.frame, the flow duration curve is computed for each column. |
| lQ.thr | numeric, low-flow separation threshold. If this value is different from NA, a vertical line is drawn in this value, and all the values to the left of it are deemed low flows.  |

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 right of it are deemed high flows
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 <a href="#">plot</a> . An overall title for the plot: see <a href="#">title</a> .
xlab	A title for the x axis. See <a href="#">plot</a> .
ylab	A title for the y axis. See <a href="#">plot</a> .
ylim	The y limits of the plot. See <a href="#">plot.default</a> .
yat	Only used when <code>log="y"</code> . 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 <code>at</code> argument in <a href="#">Axis</a> .
xat	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 <a href="#">Axis</a> .
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 <a href="#">plot.default</a> .
pch	A vector of plotting characters or symbols: see <a href="#">points</a> . See <a href="#">plot.default</a> .
lwd	The line width, see <a href="#">par</a> . See <a href="#">plot.default</a> .
lty	The line type, see <a href="#">par</a> . See <a href="#">plot.default</a> .
cex	See <a href="#">plot.default</a> . 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
cex.axis	magnification of axis annotation relative to 'cex'.
cex.lab	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
leg.txt	vector with the names that have to be used for each column of x.
leg.cex	numeric, indicating the character expansion factor for the legend, *relative* to current <code>par("cex")</code> . Default value = 1
leg.pos	keyword to be used to position the legend. One of the list ""bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"". This places the legend on the inside of the plot frame at the given location. See <a href="#">Legend</a> .
verbose	logical; if TRUE, progress messages are printed (when x is a matrix or data.frame).
thr.shw	logical, indicating if the streamflow values corresponding to the user-defined thresholds lQ.thr and hQ.thr have to be shown in the plot.
new	logical, if TRUE (default), a new plotting window is created.
...	further arguments passed to or from other methods (to the plotting functions)

**Value**

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

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

**Author(s)**

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

**References**

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

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

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

**See Also**

[fdcu](#)

**Examples**

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

## Daily Flow Duration Curve
fdc(OcaEnOnaQts)

#####
# Comparing 2 FDCs:

x <- OcaEnOnaQts

# Random 2nd ts
y <- x + rnorm(length(x), mean=10)

# data.frame with the 2 time series
xx <- data.frame(x=x, y=y)

# FDC plot
fdc(xx, thr.shw=FALSE)
```

fdcu

*Flow Duration Curve with uncertainty bounds.***Description**

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

**Usage**

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

```
## Default S3 method:
```

```
fdcu(x, lband, uband, sim, 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, 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, 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

<code>x</code>	<p>numeric, zoo, data.frame or matrix object with the observed streamflows for which the flow duration curve have to be computed.</p> <p>Measurements at several gauging stations can be stored in a data.frame of matrix object, and in that case, each column of <code>x</code> represent the time series measured in each gauging station, and the column names of <code>x</code> have to correspond to the ID of each station (starting by a letter). When <code>x</code> is a matrix or data.frame, the flow duration curve is computed for each column.</p>
<code>lband</code>	<p>numeric, zoo, data.frame or matrix object with the streamflows representing the the lower uncertainty bound of <code>x</code>, for which the flow duration curve have to be computed.</p> <p>Measurements at several gauging stations can be stored in a data.frame of matrix object. When <code>lband</code> is a matrix or data.frame, the flow duration curve is computed for each column.</p>
<code>uband</code>	<p>numeric, zoo, data.frame or matrix object with the streamflows representing the the upper uncertainty bound of <code>x</code>, for which the flow duration curve have to be computed.</p> <p>Measurements at several gauging stations can be stored in a data.frame of matrix object. When <code>uband</code> is a matrix or data.frame, the flow duration curve is computed for each column.</p>
<code>sim</code>	<p>OPTIONAL.</p> <p>numeric, zoo, data.frame or matrix object with the streamflows simulated for <code>x</code>, for which the flow duration curve have to be computed.</p> <p>Measurements at several gauging stations can be stored in a data.frame of matrix object. When <code>sim</code> is a matrix or data.frame, the flow duration curve is computed for each column.</p>
<code>lQ.thr</code>	<p>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.</p>
<code>hQ.thr</code>	<p>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</p>
<code>plot</code>	<p>logical. Indicates if the flow duration curve should be plotted or not.</p>
<code>log</code>	<p>character, indicates which axis has to be plotted with a logarithmic scale. Default value is <code>y</code>.</p>
<code>main</code>	<p>See <a href="#">plot</a>. An overall title for the plot: see <a href="#">title</a>.</p>
<code>xlab</code>	<p>See <a href="#">plot</a>. A title for the x axis: see <a href="#">title</a>.</p>
<code>ylab</code>	<p>See <a href="#">plot</a>. A title for the y axis: see <a href="#">title</a>.</p>
<code>ylim</code>	<p>See <a href="#">plot.default</a>. The y limits of the plot.</p>

yat	Only used when <code>log="y"</code> . 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 <code>at</code> argument in <a href="#">Axis</a> .
xat	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 <a href="#">Axis</a> .
col	See <a href="#">plot.default</a> . The colors for lines and points. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.
pch	See <a href="#">plot.default</a> . A vector of plotting characters or symbols: see <a href="#">points</a> .
lwd	See <a href="#">plot.default</a> . The line width, see <a href="#">par</a> .
lty	See <a href="#">plot.default</a> . The line type, see <a href="#">par</a> .
cex	See <a href="#">plot.default</a> . 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.
cex.axis	magnification of axis annotation relative to 'cex'.
cex.lab	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
leg.txt	vector with the names that have to be used for each column of x.
leg.cex	numeric, indicating the character expansion factor for the legend, *relative* to current <code>par("cex")</code> . Default value = 1
leg.pos	keyword to be used to position the legend. One of the list ""bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center". This places the legend on the inside of the plot frame at the given location. See <a href="#">legend</a> . 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
verbose	logical; if TRUE, progress messages are printed
thr.shw	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
border	See <a href="#">polygon</a> . The color to draw the border of the polygon with the uncertainty bounds. The default, 'NA', means to omit borders.
bands.col	See <a href="#">polygon</a> . 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.
bands.density	See <a href="#">polygon</a> . 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).

bands.angle See [polygon](#). The slope of shading lines for the polygon with the uncertainty bounds, given as an angle in degrees (counter-clockwise).

new logical, if TRUE, a new plotting window is created.

... further arguments passed to or from other methods (to the plotting functions)

### Note

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

### Author(s)

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

### References

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

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

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

### See Also

[fdc](#)

### Examples

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

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

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

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

## Description

Given a data.frame (`x.gis`) with the spatial coordinates of a set of measurement points (e.g., gauging stations) and the measurements in those stations (`x.ts`) this function merges the measurements in `x.ts` with the corresponding spatial location in `x.gis`, even if they are not in the same order.

If the spatial coordinates `X` and `Y` are given for `x.gis`, the resulting object will be a [SpatialPointsDataFrame-class](#) with coordinates given by the `X` and `Y` fields

If `p4s` is given, the resulting object will be projected according to the specification provided by `p4s`.

## Usage

```
gists2spt(x.gis, x.ts, sname, bname, X, Y, elevation,
          catchment.name="all", na.rm=TRUE, p4s)
```

## Arguments

<code>x.gis</code>	data.frame with the spatial information for all the measurement points in <code>x.gis</code> . The name of each station, stored in the field <code>sname</code> , have to be equal to the corresponding ID used in <code>x.ts</code> -) It MAY contain as many measurement points as you want, e.g., all the gauging stations in your database, but -) It HAVE TO contain -at least- the location of the measurement points of <code>x.ts</code> that will be used for the interpolations. The MINIMUM fields that HAVE TO be present in this file, and their corresponding column index are <code>X</code> , <code>Y</code> , <code>sname</code> .
<code>x.ts</code>	numeric or data.frame with the measured value at each station for a given time. Each value of <code>x.ts</code> has to have as name ( <code>names(x.ts)</code> ) the ID of the station. 1) It MAY contain as many stations as you want, e.g., all the gauging stations in the your database, but 2) It HAVE TO contain -at least- some stations of <code>x.gis</code>
<code>sname</code>	character, field name in <code>x.gis</code> that stores the name of the stations ( <b>the name of the stations have to start by a letter !!</b> )
<code>bname</code>	OPTIONAL. character, field name in <code>x.gis</code> that stores the name of the sub-catchment in <code>x.gis</code> that will be analysed. ONLY necessary when 'catchment.name' is not "all"
<code>X</code>	character, field name in <code>x.gis</code> that stores the easting coordinate of the measurement points. The expected name is 'x', but if the value provided by the user is different, a new 'x' field is created and is used as the easting coordinate of <code>x.gis</code>

Y	character, field name in <code>x.gis</code> that stores the northing coordinate of the measurement points. The expected name is 'y', but if the value provided by the user is different, a new 'y' field is created and is used as the northing coordinate of <code>x.gis</code>
elevation	OPTIONAL. character, field name <code>x.gis</code> that stores the elevation of the stations (m a.s.l.).
catchment.name	name of the catchment that will be analysed. Possible values are: -)all : ALL the stations in the <code>x.gis</code> will be used -)other character: ONLY those stations in <code>x.gis</code> with a <code>bname</code> field value == <code>catchment.name</code> will be used.
p4s	Character with information about the projection of the GIS files, usually created by the <code>CRS</code> function of the <code>sp</code> package
na.rm	a logical value indicating whether 'NA' values should be stripped before delivering the resulting object.

### Value

If `p4s` is given, the returning object will be a [SpatialPointsDataFrame-class](#), if not, it will be a `data.frame`

### Author(s)

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

### References

*Applied Spatial Data Analysis with R. Series: Use R.* Bivand, Roger S., Pebesma, Edzer J., Gomez-Rubio, Virgilio. 2008. ISBN: 978-0-387-78170-9

<http://r-spatial.sourceforge.net/gallery/>

### See Also

[krige](#), [spplot](#)

### Examples

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

## Loading the gis data
data(EbroPPgis)

## Putting the measurements of the first row of 'EbroPPtsMonthly' into their
## corresponding spatial location given by 'x.gis'
require(sp)
x.spt <- gists2spt(x.ts=EbroPPtsMonthly[1,], x.gis=EbroPPgis, X="EAST_ED50",
                  Y="NORTH_ED50", na.rm=FALSE, sname="ID")
```

```
## Plotting the measured values (only the first row of 'EbroPPtsMonthly') at their
## corresponding spatial location
splot(x.spt, zcol="value")
```

---

hip	<i>Hours in Period</i>
-----	------------------------

---

### 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")
```

### Arguments

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

---

hydrokrige

*Krige for Hydrological Time Series*

---

## Description

Automatic interpolation for hydrological ts, with optional plot (wrapper to some functions of the **gstat** and **automap** packages)

Originally it was thought as a way to make easier the computation of average precipitation over subcatchments (given as input in a shapefile map), based on values measured at several gauging stations, but nowadays it can also be used for interpolating any variable over a grid given by a raster map.

Available algorithms: inverse distance weighted (IDW), ordinary kriging (OK) and kriging with external drift (KED)

The (Block) Inverse Distance Weighted (IDW) interpolation is a wrapper to the `idw` function of the **gstat** package (so, it requires the **gstat** package).

The automatic kriging (OK or KED) is a wrapper to the `autoKrige` function of the **automap** package (so, it requires the **automap** and **gstat** packages), which automatically selects the best variogram model from four different ones: spherical, exponential, gaussian and Matern with M. Stein's parameterization (for more details, see `autoKrige`)

## Usage

```
hydrokrige(x.ts, x.gis, ...)
```

```
## Default S3 method:
hydrokrige(x.ts, x.gis, X= "x", Y= "y", sname, bname,
           elevation, predictors, catchment.name = "all", type="cells",
           formula, subcatchments, IDvar = NULL,
```

```

p4s=CRS(as.character(NA)), cell.size = 1000,
grid.type = "regular", nmin = 0, nmax = Inf, maxdist = Inf,
ColorRamp = "PCPAnomaly", plot = TRUE, col.nintv = 10,
col.at = "auto", main, stations.plot = FALSE, stations.offset,
arrow.plot = FALSE, arrow.offset, arrow.scale,
scalebar.plot = FALSE, sb.offset, sb.scale, verbose = TRUE,
allNA.action="error", ...)

## S3 method for class 'data.frame'
hydrokrige(x.ts, x.gis, X= "x", Y= "y", sname, bname,
elevation, predictors, catchment.name = "all", type = "block",
formula, subcatchments, IDvar= NULL,
p4s=CRS(as.character(NA)), cell.size = 1000,
grid.type = "regular", nmin = 0, nmax = Inf, maxdist = Inf,
ColorRamp = "PCPAnomaly", plot = FALSE, col.nintv = 10,
col.at = "auto", main, stations.plot = FALSE, stations.offset,
arrow.plot = FALSE, arrow.offset, arrow.scale,
scalebar.plot = FALSE, sb.offset, sb.scale,
verbose = TRUE, allNA.action="error",
dates=1, from, to, write2disk = TRUE,
out.fmt= "csv2",
fname = paste(ColorRamp, "by_Subcatch.csv", sep = ""), ...)

```

## Arguments

- x.ts**                    numeric or data.frame object, with measured values at several locations.
- ) x.ts CAN contain as many points as you want, e.g., all the gauging stations in the your database.  
2) x.ts HAVE to contain -at least- some points (e.g., gauging stations) that are also present in x.gis
- The names of each value in x.ts are used as the ID of each measurement point. When x.ts is a vector, this can be checked with `names(x)`, whereas when x.ts is a data.frame, it can be checked with `colnames(x.ts)`. The IDs of each measurement point have to be equal to the ID stored in the field `sname` of x.gis.
- When x.ts is a data.frame, its structure have to be as follow:
- ) 1st column : OPTIONAL. It MAY contain the dates, date-time or IDs that identify the measured values of each row. See `dates` argument.  
-) 2nd...Nth column: Measured values at each point (e.g., gauging station). The name of the columns is used as the ID of each station **-starting with a letter!!-**.
- x.gis**                    data.frame with the spatial information for each measurement point (e.g., gauging stations).
- ) x.gis MAY contain as many points as you want, e.g., all the stations in your database  
-) x.gis HAVE to contain -at least- the location of those stations that will be

used for the interpolations

The MINIMUM fields that have to be present in this file, and their corresponding column index are those described by: X, Y, sname.

The ID of each measurement point, given in the field sname, has to be equal to the corresponding ID used in x.ts

X	character, field name in x.gis that stores the easting coordinate of each measurement point.
Y	character, field name in x.gis that stores the northing coordinate of each measurement point.
sname	character, field name in x.gis that stores the ID of each measurement point. <b>the name of each measurement point HAS to start by a letter!!-</b>
bname	OPTIONAL. character, field name in x.gis that stores the name of the subcatchment in x.gis that will be analysed. ONLY necessary when catchment.name is different from all.
elevation	OPTIONAL. character, field name in x.gis that stores the elevation of the gauging stations (m a.s.l.).
predictors	OPTIONAL. <a href="#">SpatialGridDataFrame-class</a> object, with prediction/simulation locations (it is needed for KED). Usually, a digital elevation model (DEM) read with the <a href="#">readGDAL</a> function of the <b>rgdal</b> package. See the newdata argument in <a href="#">krige</a> . It should contain attribute columns with the independent variables (if present) and the coordinates with names as defined in x.gis If predictors is missing, the grid to be used as prediction/simulation locations is generated from sampling the polygon specified by the user in subcatchments, according to the arguments provided by cells.size and grid.type
catchment.name	name of the catchment that will be analysed. Possible values are: -)all : ALL the stations in the x.gis will be used -)other string: ONLY those stations in x.gis with a bname field value equal to catchment.name will be used .
type	Character, indicating the type of interpolation required by the user. When x.ts is a data.frame, the ONLY possible value is block. For all the other cases, possible values are: -) cells : the interpolated values are computed for each cell -) block : the interpolated values are computed for each subcatchment, where the value for each subcatchment is computed as the mean value over all the cells that belong to each subcatchment -) both : cells and block are computed.
formula	OPTIONAL. Formula to be used in case of ordinary kriging or kriging with external drift. Requires the <b>automap</b> package. All the variables to be used within formula has to be present both in x.gis and predictors. See the formula argument in <a href="#">krige</a> . formula defines the dependent variable as a linear model of independent variables. <b>Within this function, the dependent variable is always called value,</b>

therefore, for ordinary and simple kriging use the formula  $value \sim 1$ ; for simple kriging also define  $\beta$ ; for universal kriging, suppose  $value$  is linearly dependent on  $x$  and  $y$ , use the formula  $value \sim x + y$ .

subcatchments	<p>OPTIONAL. Only required when <code>predictors</code> is missing.</p> <p>Spatial polygon with all the subcatchments to be used as interpolation domain. The polygons are used to create the grid that will be used as prediction/simulation locations, from sampling it according to the arguments provided by <code>cells.size</code> and <code>grid.type</code>. Valid values are:</p> <ol style="list-style-type: none"> <li>1) Character, indicating the filename (with path) of the shapefile that contains all the subcatchments to be used as interpolation domain. It HAS TO BE of 'polygon' type</li> <li>2) <a href="#">SpatialPolygonsDataFrame-class</a> resulting from reading the shapefile (e.g., with the command <code>readShapePoly</code> of the <b>maptools</b> package) that contains all the subcatchments to be used as interpolation domain.</li> </ol>
IDvar	<p>See <code>readShapePoly</code>. a character string with the name of a field in the subcatchments shapefile containing the ID values used to identify each one of the subcatchments. When <code>type</code> is <code>block</code>, the values stored in this field will be used for labelling the computed values in each one of the subcatchments, therefore, if you don't provide this value, it could be difficult to identify which computed value corresponds to which subcatchment, because the ID is assigned by the <code>readShapePoly</code> function.</p>
p4s	<p>OPTIONAL. a character with information about the projection of the GIS files, usually created by the CRS function of the <b>sp</b> package.</p>
cell.size	<p>OPTIONAL. Only required when <code>predictors</code> is missing. Size of the cells [m] to be used for sampling the polygons specified by the user in subcatchments, to create a grid to be used as prediction/simulation locations .</p>
grid.type	<p>OPTIONAL. Only required when <code>predictors</code> is missing. See <code>spsample</code>. Character, indicating the type of grid to be computed over the area defined by subcatchments. Valid values are:</p> <ul style="list-style-type: none"> <li>-) <code>regular</code> : for regular (systematically aligned) sampling; Default option</li> <li>-) <code>random</code> : for completely spatial random;</li> <li>-) <code>stratified</code> : for stratified random (one single random location in each "cell")</li> <li>-) <code>nonaligned</code> : for non-aligned systematic sampling (nx random y coordinates, ny random x coordinates);</li> <li>-) <code>hexagonal</code> : for sampling on a hexagonal lattice;</li> <li>-) <code>clustered</code> : for clustered sampling</li> </ul>
nmin	<p>OPTIONAL. See <code>krige</code>. For local interpolation: if the number of nearest observations within distance <code>maxdist</code> is less than <code>nmin</code>, a missing value will be generated; see <code>maxdist</code>. By default <code>nmin=0</code>.</p>
nmax	<p>OPTIONAL. See <code>krige</code>. For local interpolation: the number of nearest observations that should be used for a kriging prediction, where nearest is defined in terms of the space of the spatial locations. By default, all observations are used.</p>
maxdist	<p>OPTIONAL. See <code>krige</code>. For local interpolation: only observations within a distance of <code>maxdist</code> from the prediction location are used for prediction or simulation; if combined with <code>nmax</code>, both criteria apply. By default, all observations are used.</p>

ColorRamp	Function defining the colour ramp to be used for plotting the maps OR character representing the colours to be used in the plot. In the latter case, valid values are in: <code>c('Precipitation', 'Temperature', 'PCPAnomaly', 'PCPAnomaly2', 'TEMPAnomaly', 'TEM</code>
plot	Logical, indicating if the interpolated values have to be plotted or not
col.nintv	integer, number of colours that have to be used for plotting the interpolated values
col.at	Specify at which interpolated values colours change. Valid values are: -) R : uses the default setting of <code>spplot</code> -) numeric: vector of reals giving the exact values in which the colors have to change. Useful when the user desires the same color for the same value when comparing to maps with different range of values -) auto : default option. <pre>at &lt;- seq(min, max, length.out=col.nintv) min &lt;- floor(min(idw["var1.pred"]@data, na.rm=TRUE)) max &lt;- ceiling(max(idw["var1.pred"]@data, na.rm=TRUE))</pre>
main	Character with the title to be used for the plot.
stations.plot	Logical, indicating if the gauging stations, defined by <code>x.gis</code> have to be plotted
stations.offset	2D list with the numeric coordinates in which the label with the amount of gauging stations have to be plotted. e.g., <code>stations.offset = c(450000, 4600000)</code> .
arrow.plot	Logical, indicating if a North Arrow have to be plotted
arrow.offset	2D list with the numeric coordinates in which the north arrow has to be plotted. e.g., <code>arrow.offset = c(690000, 4760000)</code>
arrow.scale	Scale (in the map units) to be used for plotting the north arrow, e.g., <code>scale = 20000</code>
scalebar.plot	Logical, indicating if a Scale Bar has to be plotted
sb.offset	2D list with the numeric coordinates in which the North Arrow has to be plotted. e.g., <code>sb.offset = c(400000, 4490000)</code>
sb.scale	Scale (in the map units) to be used for plotting the Scale Bar, e.g., <code>scale = 100000</code> , means that the scale bar will have a length of 100km
verbose	logical; if TRUE, progress messages are printed
allNA.action	Action to be executed when all the values in <code>x.ts</code> are NA. Valid values are: -) "error": it will produce an error message. Default option -) a single numeric value that will replace all the NA values in <code>x.ts</code> , giving place to a map with a constant value. <b>At your own risk !</b>
dates	numeric, factor, or character object indicating how to obtain the ID (usually dates) that will be used to identify the interpolation carried out for each row of <code>x.ts</code> . If dates is a single number (default), it indicates the index of the column in <code>x.ts</code> that stores the dates If dates is a factor or character vector, its values will be used as ID for the interpolations carried out in each row of <code>x.ts</code> .
from	Character indicating the starting date for the values stored in all the files that will be read.

to	Character indicating the ending date for the values stored in all the files that will be read.
write2disk	Logical. Indicates if we want to write the output into a CSV file. Default value is TRUE
out.fmt	OPTIONAL, only needed when write2disk==TRUE. Character indicating the type of csv file to be written with the results. Valid values are csv and csv2. For more information, see <a href="#">write.table</a>
fname	OPTIONAL. Character with the filename of the output file. Only needed when write2disk=TRUE
...	further arguments passed to or from other methods. In particular, these further arguments are passed to the function <code>idw</code> ( <b>gstat</b> package) OR <code>autoKrige</code> ( <b>automap</b> package), depending on the value passed to <code>formula</code> (see 'details' below): -) for IDW, the arguments are passed to: <code>idw(formula, locations, newdata, nmin, nmax, maxdist,</code> -) for OK, KED, the arguments are passed to: <code>autoKrige(formula, input_data, new_data, nmin, nm</code>

## Details

The type of interpolation (IDW, OK, KED) is obtained from the argument `formula`:

- ) When `formula` is missing, an IDW interpolation, by calling the `idw` function in the **gstat** package, with `formula = value~1`.
- ) When `formula = value~1`, an OK interpolation, by calling the `autoKrige` function, with `formula = value~1`.
- ) When `formula = value~pred1 + pred2 + ...`, a KED interpolation, by calling the `autoKrige` function, with the `formula` specified by the user.

When `type` is `block` or `both`, a block interpolation is carried out for each subcatchment defined by subcatchments, so far, computing the average value over all the cells belonging to each subcatchment.

The automatic kriging is carried out by using a variogram generated automatically with the `autofitVariogram` function of the **automap** package.

## Value

Cells	When <code>type</code> is <code>cells</code> , the output object is a <code>SpatialPixelsDataFrame-class</code> , which slot <code>'data'</code> has two variables: <code>'var1.pred'</code> and <code>'var1.var'</code> with the predictions and its variances, respectively
Block	When <code>type</code> is <code>block</code> , the output object is a <code>SpatialPolygonsDataFrame-class</code> , which slot <code>'data'</code> has four variables: <code>'x'</code> , <code>'y'</code> with the easting and northing coordinate of the centroid of the catchments specified by subcatchments, and <code>'var1.pred'</code> and <code>'var1.var'</code> with the predictions and its variances, respectively

```
list(Cells, Block)
```

When type is both, the resulting object is a list, with the two elements previously described.

### Note

IMPORTANT: It is your responsibility to check the validity of the fitted variogram !!.

### Author(s)

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

### References

*N.A.C. Cressie, 1993, Statistics for Spatial Data, Wiley.*

*Applied Spatial Data Analysis with R. Series: Use R. Bivand, Roger S., Pebesma, Edzer J., Gomez-Rubio, Virgilio. 2008. ISBN: 978-0-387-78170-9*

*Pebesma, E.J., 2004. Multivariable geostatistics in S: the gstat package. Computers & Geosciences, 30: 683-691*

<http://www.gstat.org/>

<http://r-spatial.sourceforge.net/gallery/>

### See Also

[krige](#), [autoKrige](#), [readShapePoly](#), [spsample](#)

### Examples

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

## Loading the gis data
data(EbroPPgis)

## Loading the shapefile with the subcatchments
data(EbroCatchmentsCHE)

## Projection for the Subcatchments file
# European Datum 50, Zone 30N
require(sp)
p4s <- CRS("+proj=utm +zone=30 +ellps=intl +units=m +no_defs")

## Selecting the first day of 'EbroPPtsMonthly' for all the stations.
# The first column of 'EbroPPtsMonthly' has the dates
x.ts <- as.numeric(EbroPPtsMonthly[1, 2:ncol(EbroPPtsMonthly)])
```

```

## Setting the name of the gauging stations
names(x.ts) <- colnames(EbroPPtsMonthly[1,2:ncol(EbroPPtsMonthly)])

#####
## 1) IDW interpolation and plot
## Probably you will need to resize your window
## Not run:
x.idw <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                    X="EAST_ED50", Y="NORTH_ED50", sname="ID", bname="CHE_BASIN_NAME",
                    type= "both",
                    subcatchments= EbroCatchmentsCHE,
                    cell.size= 3000,
                    ColorRamp= "Precipitation",
                    main= "IDW Precipitation on the Ebro")

## End(Not run)

#####
## 2) Ordinary Kriging interpolation and plot, in catchments defined by a shapefile
## Probably you will need to resize your window
## Not run:

# Computing OK, over of 3000x3000m, sampled within the subcatchments defined by 'subcatchments'
x.ok <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                  X="EAST_ED50", Y="NORTH_ED50", sname="ID", bname="CHE_BASIN_NAME",
                  type= "both", formula=value~1,
                  subcatchments= EbroCatchmentsCHE,
                  p4s= p4s,
                  cell.size= 3000,
                  ColorRamp= "Precipitation",
                  main= "OK Precipitation on the Ebro", arrow.plot= TRUE,
                  arrow.offset= c(900000,4750000), arrow.scale= 20000,
                  scalebar.plot= TRUE,
                  sb.offset= c(400000,4480000), sb.scale= 100000)

## End(Not run)

#####
## 3) Ordinary Kriging interpolation and plot, in an area defined by a raster map.
## The raster map may be any \link[sp]{SpatialGridDataFrame-class} object, read with
## the \code{\link[rgdal]{readGDAL}} function of the \pkg{rgdal} package or similar.
## Probably you will need to resize your window

#Loading the DEM
data(EbroDEM1000m)

#Giving a meaningful name to the predictor
EbroDEM1000m$ELEVATION <- EbroDEM1000m$band1

# Saving memory
EbroDEM1000m$band1 <- NULL

```

```

# Computing OK, over the spatial grid defined by the DEM
## Not run:
x.ok <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                  X="EAST_ED50", Y="NORTH_ED50", sname="ID",
                  formula=value~1,
                  p4s= p4s,
                  predictors=EbroDEM1000m,
                  ColorRamp= "Precipitation",
                  main= "OK Precipitation on the Ebro",
                  arrow.plot= TRUE,
                  arrow.offset= c(900000,4750000), arrow.scale= 20000,
                  scalebar.plot= TRUE,
                  sb.offset= c(400000,4480000), sb.scale= 100000)

## End(Not run)

#####
## 4) Kriging with External Drift interpolation and plot
## Probably you will need to resize your window

#Loading the DEM
data(EbroDEM1000m)

#Giving a meaningful name to the predictor
EbroDEM1000m$ELEVATION <- EbroDEM1000m$band1

# Saving memory
EbroDEM1000m$band1 <- NULL

# Computing KED
## Not run:
x.ked <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                   X="EAST_ED50", Y="NORTH_ED50", sname="ID",
                   bname="CHE_BASIN_NAME", elevation="ELEVATION",
                   type= "cells",
                   formula=value~ELEVATION,
                   subcatchments= EbroCatchmentsCHE,
                   predictors=EbroDEM1000m,
                   cell.size= 3000,
                   ColorRamp= "Precipitation",
                   main= "KED Precipitation on the Ebro",
                   arrow.plot= TRUE,
                   arrow.offset= c(900000,4750000), arrow.scale= 20000,
                   scalebar.plot= TRUE,
                   sb.offset= c(400000,4480000), sb.scale= 100000)

## End(Not run)

#####
## 5) Block IDW interpolation and plot of 'EbroPPtsMonthly' for 3 months
## Not run:
x.idw <- hydrokrige(x.ts= EbroPPtsMonthly, x.gis=EbroPPgis,
                   X="EAST_ED50", Y="NORTH_ED50", sname="ID",

```

```

bname="CHE_BASIN_NAME",
type= "cells", #'both'
subcatchments= EbroCatchmentsCHE,
cell.size= 3000,
ColorRamp= "Precipitation",
arrow.plot= TRUE,
arrow.offset= c(900000,4750000), arrow.scale= 20000,
scalebar.plot= TRUE,
sb.offset= c(400000,4480000), sb.scale= 100000,
dates=1,
from="1942-01-01", to="1942-03-01")

## End(Not run)

```

---

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: <http://addictedtor.free.fr/graphiques/graphcode.php?graph=137>.

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

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, to, stype="default",
season.names=c("Winter", "Spring", "Summer", "Autumn"),
h=NULL )

```

### Arguments

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

<code>xlab</code>	A title for the x axis. See <a href="#">plot</a> .
<code>ylab</code>	A title for the y axis. See <a href="#">plot</a> .
<code>win.len1</code>	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 <a href="#">as.Date</a> . 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 <a href="#">par</a> ).
<code>cex.main</code>	The magnification to be used for main titles relative to the current setting of <code>cex</code> (See <a href="#">par</a> ).
<code>cex.lab</code>	The magnification to be used for x and y labels relative to the current setting of <code>cex</code> (See <a href="#">par</a> ).
<code>cex.axis</code>	The magnification to be used for axis annotation relative to the current setting of <code>cex</code> (See <a href="#">par</a> ).
<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 <a href="#">plot.default</a> ).
<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 <a href="#">as.Date</a> . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="numeric"</code> .

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 <code>date.fmt</code> .
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 <code>date.fmt</code> .
stype	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
season.names	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
h	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).
...	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 `pptype`, 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(OcaEn0naQts, pfreq="o")

## only the year 1962 of the original time series
hydroplot(OcaEn0naQts, pfreq="o", from="1962-01-01", to="1962-12-31")

## seasonal plots
hydroplot(OcaEn0naQts, pfreq="seasonal", FUN=mean, stype="default")

## custom season names (let's assume to be in the Southern Hemisphere)
hydroplot(OcaEn0naQts, 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)

## 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")

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

```

---

hypsoMetric

*Hypsometric Curve*


---

## Description

Computes and plots the hypsoMetric curve corresponding to the data provided by a digital elevation model (DEM)

## Usage

```

hypsoMetric(x, band=1, main="Hypsometric Curve",
           xlab="Relative Area above Elevation, (a/A)",
           ylab="Relative Elevation, (h/H)", col="blue",...)

```

## Arguments

x	<a href="#">SpatialGridDataFrame-class</a> object with the elevations of the catchment. Possibly, a raster file already read with the <a href="#">readGDAL</a> function of the <a href="#">rgdal</a> package.
band	integer or character indicating the band in x that stores the elevation data.
main	See <a href="#">plot</a> . An overall title for the plot: see <a href="#">title</a> .
xlab	See <a href="#">plot</a> . A title for the x axis: see <a href="#">title</a> .
ylab	See <a href="#">plot</a> . A title for the y axis: see <a href="#">title</a> .
col	See <a href="#">plot.default</a> . The colors for lines and points.
...	further arguments passed to or from other methods

## Details

The hypsometric curve and the hypsometric integral are non-dimensional measures of the proportion of the catchment above a given elevation.

Strahler (1952, 1964) further asserted that different types of landform have different characteristic shape of their hypsometric curves, dividing landforms into 'young' and 'mature' with decreasing hypsometric integral -the area under the hypsometric curve- with age.

## Note

Based on <http://osgeo-org.1803224.n2.nabble.com/hypsometric-integral-from-ecdf-curve-td2231345.html>

## Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail>, with contribution of Jan Schwanbeck.

## References

Strahler, A. N. 1952. *Hypsometric (area-altitude) analysis of erosional topography*, *Geological Society of America Bulletin*, 63, 1117-1142

Strahler, A. N. 1964. *Quantitative geomorphology of drainage basins and channel networks*, in Chow, V. T. (Ed.) *Handbook of Applied Hydrology*, McGraw Hill, New York, 4-39-4-76

Luo, W. 1998. *Hypsometric analysis with a geographic information system*, *Computers & Geosciences* 24, pp. 815-821

Willgoose, G. and Hancock, G. 1998. *Revisiting the hypsometric curve as an indicator of form and process in transport-limited catchment*, *Earth Surface Processes and Landforms* 23, pp. 611-623

## Examples

```
## Not run:
# Loading the DEM
require(rgdal)
data(EbroDEM1000m)
dem <- EbroDEM1000m

# Plotting the DEM
require(sp)
splot(dem, scales=list(draw=TRUE, y=list(rot=90)))

# Computing and plotting the hypsometric curve
hypsometric(dem)

# If the raster file has more than 1 band, and the elevation data are in a
# band different from 1:
dem$ELEVATION <- EbroDEM1000m$band1 # dummy example
```

```
hypsothetic(dem, band= 2)
hypsothetic(dem, band= "ELEVATION") # same as before, but user-friendly

## 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)
```

**Description**

This function back transforms a standardized 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  $z = \text{re-scale}(x) = x * [ \text{xmax} - \text{xmin} ] + \text{xmin}$ .

**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 $x$ -) if $x$ is a vector, <code>xmin</code> has to be a real -) if $x$ is a matrix/data.frame, <code>xmin</code> has to be a vector, with the minimum values for each column of the original $x$ . In this case, the vector of minimums can be obtained as: <code>xmin &lt;- apply(x, 2, min, na.rm=TRUE)</code>
<code>xrange</code>	numeric with the range of value(s) in the original $x$ -) if $x$ is a vector, <code>xrange</code> has to be a real -) if $x$ is a matrix/data.frame, <code>xrange</code> has to be a vector, with the range of values for each column of the original $x$ . In this case, the vector of ranges can be obtained as: <code>xrange &lt;- apply(x, 2, range, na.rm=TRUE)</code> <code>xrange &lt;- 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))

## Standardized variable
s <- stdx(x)

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

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

#####
### Standardizing 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)

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

## Inverse of the standardized 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) 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= "%Y-%m-%d", tstep= "days", ...)

## S3 method for class 'zoo'

```

```
izoo2rzoo(x, from= start(x), to= end(x), date.fmt= "%Y-%m-%d", tstep= "days", ...)
```

### Arguments

x	irregular zoo object (vector or matrix) representing a time series (very likely read with some user-defined procedure, and with some missing values for particular days/months/years)
from	Character indicating the starting date for creating the regularly spaced zoo object. The default value corresponds to the date of the first element of x. It has to be in the format indicated by date.fmt.
to	Character indicating the ending date for creating the regularly spaced zoo object. The default value corresponds to the date of the last element of x. It has to be in the format indicated by date.fmt.
date.fmt	character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See 'Details' section in <a href="#">strptime</a>
tstep	character, indicating the time step used for creating the time sequence going from from to to that will be used as time(x). Valid values are (but not limited to) days, months, years, hours
...	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 do 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

[vector2zoo](#)

### 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 missing

## 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' (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
Y <- cbind(x,x)

# Adding a missing value (NA in this case) for Feb 29th in all the columns of Y
rY <- izoo2rzoo(Y, from="1964-02-25", to="1964-03-05")

##
## Example 4: Working with hourly data, from 01:00 to 10:00 on 12th December 2000
dates <- ISOdatetime(2000, 12, 12, 1:10, 0, 0)
values <- 1:10
x <- zoo(values, dates)

# removing values from 02:00 to 05:00 (not present at all in 'x', not even NA !)
x <- x[-c(2:5)]

# Adding missing values (NA in this case) from 02:00 to 05:00
y <- izoo2rzoo(x, date.fmt="%Y-%m-%d %H:%M:%S", tstep="hours")

```

---

KarameaAtGorgeQts

*Karamea at Gorge, time series of hourly streamflows*


---

## Description

Time series with hourly streamflows for the Karamea River(New Zeland) 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.

**Usage**

```
data(KarameaAtGorgeQts)
```

**Format**

zoo object.

**Source**

Provided by the National Institute of Water and Atmospheric Research <http://www.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")
```

---

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 = 70, main = "", ...)
```

**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>...</code>	further arguments passed to <code>levelplot</code> function ( <b>lattice</b> package) or from other methods

**Note**

Adapted from: [http://www2.warwick.ac.uk/fac/sci/moac/currentstudents/peter\\_cock/r/matrix\\_contour/](http://www2.warwick.ac.uk/fac/sci/moac/currentstudents/peter_cock/r/matrix_contour/)

**Author(s)**

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

**See Also**

[dwi](#)

**Examples**

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

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

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

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

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

---

mip

*Months in Period*

---

**Description**

Given any starting and ending dates, it generates:

- 1) a vector of class 'Date' with all the months between the two dates (both of them included), OR
- 2) the amount of months between the two dates

**Usage**

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

**Arguments**

from	Character indicating the starting date for creating the sequence. It has to be in the format indicated by date.fmt.
to	Character indicating the ending date for creating the sequence. It has to be in the format indicated by date.fmt.
date.fmt	Character indicating the format in which the dates are stored in from and to, e.g. %Y-%m-%d. See format in <a href="#">as.Date</a> .
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 -) nmb: 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 = "nmb")
```

---

monthlyfunction

*Monthly Function*

---

**Description**

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

**Usage**

```

monthlyfunction(x, ...)

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

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

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

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

```

**Arguments**

x	zoo, xts, data.frame or matrix object, with daily or monthly time series. Measurements at several gauging stations can be stored in a data.frame 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 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 <a href="#">as.Date</a> . ONLY required when class(dates)=="factor" or class(dates)=="numeric".
out.type	It is only used when x is a matrix or data.frame. Character defining the desired type of output. Valid values are: -) data.frame: a data.frame, with 12 columns representing the months, and as many rows as gauging stations are included in x

-) db : a data.frame, with 4 columns will be produced. Useful for a posterior boxplot  
 The first column ('StationID') will store the ID of the station,  
 The second column ('Year') will store the year,  
 The third column ('Month') will store month,  
 The fourth column ('Value') will contain the monthly value corresponding to the three previous columns.

verbose      Logical; if TRUE, progress messages are printed  
 ...          further arguments passed to or from other methods

### Value

When *x* is a zoo object, a numeric vector with 12 elements representing the computed monthly value for each month.

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

### Note

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

### Author(s)

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

### See Also

[annualfunction](#), [seasonalfunction](#), [dm2seasonal](#), [daily2monthly](#), [daily2annual](#)

### Examples

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

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

#####
## Boxplot of monthly values

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

## Median of the monthly values at the station
```

```

monthlyfunction(m, FUN=median, na.rm=TRUE)

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

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

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

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

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

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

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

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

m <- monthlyfunction(z, FUN=sum)

## Another way of computing the sum of the monthly values in each station of 'x'
## This way is useful 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)

```

## Description

Plots several spatial maps in the same plotting window, with options to easily add a scale bar and north arrow.

Originally, it was though to make easier the display of several maps (of the same spatial extent) obtained by different interpolation methods or for different time steps.

## Usage

```
mspplot(x, subcatchments, IDvar = NULL, p4s=CRS(as.character(NA)),
        plot = TRUE, ColorRamp="PCPAnomaly", col.nintv = 10, col.at = "auto",
        main, stations.plot = FALSE, stations.gis, X, Y,
        arrow.plot = FALSE, arrow.offset, arrow.scale,
        scalebar.plot = FALSE, sb.offset, sb.scale, verbose = TRUE)
```

## Arguments

x	<a href="#">SpatialPixelsDataFrame-class</a> object that stores (in the @data slot) all the maps that will be plotted.
subcatchments	<a href="#">SpatialPolygonsDataFrame-class</a> with all the subcatchments within the study area, <b>OR</b> character representing the filename (with path) of the shapefile ('polygon' type) that stores those subcatchments.
IDvar	See <a href="#">readShapePoly</a> . Character string with the name of a column in the subcatchments shapefile DBF containing the ID values of the catchments - the values will be converted to a character vector.
p4s	Character with information about the projection of the GIS files, usually created by the CRS function of the <b>sp</b> package
plot	Logical, indicating if the interpolated values have to be plotted or not
ColorRamp	Character or function defining a personalized color ramp for plotting the maps. Valid character values are in c("Precipitation", "Temperature", "PCPAnomaly", "PCPAnomaly2", "TEMPAnomaly", "TEMPAnomaly2", "TEMPAnomaly3").
col.nintv	integer, number of colors that have to be used for plotting the interpolated values
col.at	Specify at which interpolated values colours change. Valid values are: -) R : uses the default setting of 'splot' -) auto: default option. at <- seq(min, max,length.out=col.nintv) min <- floor( min(idw["var1.pred"]@data, na.rm=TRUE)) max <- ceiling( max(idw["var1.pred"]@data, na.rm=TRUE)) -) numeric: vector giving the exact values in which the colors have to change. Useful when the user desires the same color for the same value when comparing two or more maps with different range of values
main	Character with the title to be used for the plot
stations.plot	Logical, indicating if the gauging stations, defined by stations.gis have to be plotted
stations.gis	OPTIONAL. data.frame with the stations that will be added to the plot. ONLY required when stations.plot == TRUE.

X	OPTIONAL. character, field name in <code>x.gis</code> that stores the easting coordinate of the stations. ONLY required when <code>stations.plot == TRUE</code> .
Y	OPTIONAL. character, field name in <code>x.gis</code> that stores the northing coordinate of the stations. ONLY required when <code>stations.plot == TRUE</code> .
<code>arrow.plot</code>	Logical, indicating if a North Arrow have to be plotted
<code>arrow.offset</code>	OPTIONAL. 2D list with the numeric coordinates in which the North Arrow have to be plotted. e.g., <code>arrow.offset = c(690000,4760000)</code> . ONLY required when <code>arrow.plot=TRUE</code>
<code>arrow.scale</code>	OPTIONAL. Scale (in the map units) to be used for plotting the North Arrow, e.g., <code>scale = 20000</code> . ONLY required when <code>arrow.plot=TRUE</code> .
<code>scalebar.plot</code>	Logical, indicating if a Scale Bar have to be plotted
<code>sb.offset</code>	OPTIONAL. 2D list with the numeric coordinates in which the North Arrow have to be plotted. e.g., <code>sb.offset = c(400000,4490000)</code> . ONLY required when <code>scalebar.plot=TRUE</code>
<code>sb.scale</code>	OPTIONAL. Scale (in the map units) to be used for plotting the Scale Bar, e.g., <code>scale = 100000</code> , means that the scale bar will have a length of 100km. ONLY required when <code>scalebar.plot=TRUE</code> .
<code>verbose</code>	logical; if TRUE, progress messages are printed

**Author(s)**

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

**References**

*Applied Spatial Data Analysis with R. Series: Use R. Bivand, Roger S., Pebesma, Edzer J., Gomez-Rubio, Virgilio. 2008. ISBN: 978-0-387-78170-9*  
<http://r-spatial.sourceforge.net/gallery/>

**See Also**

[spplot](#), [krige](#), [autoKrige](#)

**Examples**

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

## Loading the gis data
data(EbroPPgis)

## Loading the shapefile with the subcatchments
data(EbroCatchmentsCHE)

## Projection for the Subcatchments file
require(sp)
p4s <- CRS("+proj=utm +zone=30 +ellps=intl +units=m +no_defs")
```

```

## Field name in 'x.gis' with the ID of the station
sname <- "ID"
## Field name in 'x.gis' with the name of the catchment to which each station belongs
bname <- "CHE_BASIN_NAME"
## Field name in 'x.gis' with the Easting spatial coordinate
X <- "EAST_ED50"
## Field name in 'x.gis' with the Northing spatial coordinate
Y <- "NORTH_ED50"
## Field name in 'x.gis' with the Elevation
elevation <- "ELEVATION"

#####
## Selecting Jan/1961 (first row) of 'EbroPPtsMonthly' in all the stations
x.ts <- as.numeric(EbroPPtsMonthly[1, 2:ncol(EbroPPtsMonthly)])

## Setting the name of the stations
names(x.ts) <- colnames(EbroPPtsMonthly[ , 2:ncol(EbroPPtsMonthly)])

#####
## IDW interpolation and plot (Jan/61)
x.idw <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                   X=X, Y=Y, sname=sname, bname=bname, elevation=elevation,
                   type= "cells", #'both'
                   subcatchments= EbroCatchmentsCHE, p4s= p4s,
                   cell.size= 3000, nmax= 50,
                   ColorRamp= "Precipitation",
                   main= "IDW Mean Annual Precipitation on the Ebro, Jan/1961")

## Storing the interpolated values
x.idw@data["Jan1961"] <- x.idw@data["var1.pred"]
x.idw@data["var1.pred"] <- NULL
x.idw@data["var1.var"] <- NULL

## Selecting the Jul/1961 of 'EbroPPtsMonthly' for all the stations
x.ts <- as.numeric(EbroPPtsMonthly[7, 2:ncol(EbroPPtsMonthly)])

## Setting the name of the stations
names(x.ts) <- colnames(EbroPPtsMonthly[ , 2:ncol(EbroPPtsMonthly)])

## Not run:
## IDW interpolation and plot (Jul/1961)
x.idw2 <- hydrokrige(x.ts= x.ts, x.gis=EbroPPgis,
                    X=X, Y=Y, sname=sname, bname=bname, elevation=elevation,
                    type= "cells", #'both'
                    subcatchments= EbroCatchmentsCHE, p4s= p4s,
                    cell.size= 3000, nmax= 50,
                    ColorRamp= "Precipitation",
                    main= "IDW Mean Annual Precipitation on the Ebro, Jul/1961")

```

```
# Adding the interpolated value for Jul/1961 to 'x.idw'  
x.idw@data["Jul1961"] <- x.idw2@data["var1.pred"]  
  
## Plotting in the same graph the 2 interpolated fields  
msplot(x=x.idw,  
        subcatchments=EbroCatchmentsCHE,  
        IDvar=NULL, p4s,  
        col.nintv=50,  
        main="IDW Monthly Precipitation on the Ebro River basin, [mm]",  
        stations.plot=FALSE,  
        arrow.plot=TRUE, arrow.offset=c(900000,4750000), arrow.scale=20000,  
        scalebar.plot=TRUE, sb.offset=c(400000,4480000), sb.scale=100000)  
  
## End(Not run)
```

---

OcaEnOnaQts

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

---

## Description

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

## Usage

```
data(OcaEnOnaQts)
```

## Format

zoo object.

## Source

Downloaded from the web site of the Confederacion Hidrografica del Ebro (CHE) <http://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.

---

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	<i>San Martino, ts of daily precipitation.</i>
----------------	--

---

**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 ( <b>very important when FUN=sum</b> ) -) 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 <a href="#">as.Date</a> . 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="nubr")

## 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	<i>Sampling Frequency</i>
-------	---------------------------

---

**Description**

This function identifies the sampling frequency of a zoo object. So far, it only works with daily/monthly/annual ts.

**Usage**

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

**Arguments**

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

**Value**

Character. Possible values are:

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

**Author(s)**

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

**See Also**

[frequency](#)

**Examples**

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

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

## Daily to Annual
a <- daily2annual(d, FUN=mean, na.rm=TRUE)

# Daily zoo object
```

```

sfreq(d)

# Monthly zoo object
sfreq(m)

# Annual zoo object
sfreq(a)

```

---

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 <b>e1071</b> package)
kurtosis	Kurtosis (using <b>e1071</b> 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 -&gt; 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
date.fmt	character indicating the format in which the dates are stored in dates, e.g. %Y-%m-%d. See format in <a href="#">as.Date</a> . ONLY required when class(dates)=="factor" or class(dates)=="numeric".
var.type	character representing the type of variable being plotted. Used for determining the function used for computing the monthly or/and annual values when FUN is missing. Valid values are: -) Precipitation => FUN=sum -) Temperature => FUN= mean -) Flow => FUN= mean
tstep.out	character that defines the time step of the desired output time series. Valid values are: -) daily : daily time series -) monthly: monthly time series -) annual : annual time series
FUN	ONLY required when var.type is missing and tstep is one of monthly or annual.

Function that have to be applied for transforming from daily to monthly or annual time step (e.g., for precipitation FUN=sum and for temperature and flow ts, FUN=mean)

`na.rm` a logical value indicating whether 'NA' values should be stripped before the computation proceeds.

`from` OPTIONAL, used for extracting a subset of values.  
Character indicating the starting date for the values to be extracted. It must be provided in the format specified by `date.fmt`.

`to` OPTIONAL, used for extracting a subset of values.  
Character indicating the ending date for the values to be extracted. It must be provided in the format specified by `date.fmt`.

**Value**

zoo object

**Author(s)**

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

**See Also**

[sname2plot](#)

**Examples**

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

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

---

stdx

*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]))
```

---

subdaily2daily	<i>Sub-daily -&gt; Daily</i>
----------------	------------------------------

---

**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, ...)

## S3 method for class 'zoo'
subdaily2daily(x, FUN, na.rm = TRUE, ...)

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

**Arguments**

x	xts, 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 represent the time series measured in each gauging station, and the column names of x should correspond to the ID of each station (starting by a letter).
FUN	Function that have to be applied for transforming from sub-daily to daily 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 daily values are computed considering only those values different from NA -) FALSE: if there is AT LEAST one NA sub-daily value within a day, the corresponding daily value(s) will be NA as well
dates	numeric, factor, POSIXct or POSIXt object indicating how to obtain the dates and times for each 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 <a href="#">as.Date</a> . 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.

**Author(s)**

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

**See Also**

[subdaily2monthly](#), [subdaily2annual](#), [subdaily2seasonal](#), [as.POSIXct](#), [dm2seasonal](#), [monthlyfunction](#), [seasonalfunction](#), [hydroplot](#), [vector2zoo](#), [izoo2rzoo](#)

**Examples**

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

```
# Plotting the hourly streamflow values
plot(x)

## sub-daily to Daily
d <- subdaily2daily(x, FUN=sum, na.rm=TRUE)

# Plotting the daily streamflow values
plot(d)
```

---

```
time2season          Date/DateTime character -> Seasonal character
```

---

## Description

This function transforms a character vector of Dates or DateTimes into a character vector of seasons (summer, winter, autumn, spring), depending on the value of type:

When type=default -) winter = DJF: December, January, February  
 -) spring = MAM: March, April, May  
 -) summer = JJA: June, July, August  
 -) autumn = SON: September, October, November

When type=FrenchPolynesia -) winter = DJFM: December, January, February, March  
 -) spring = AM : April, May  
 -) summer = JJAS: June, July, August, September  
 -) autumn = ON : October, November

## Usage

```
time2season(x, out.fmt = "months", type="default")
```

## Arguments

x	vector with the dates that have to be transformed. class(x) must be Date
out.fmt	character, indicating the format of the output seasons. Possible values are: -) seasons => c("winter", "spring", "summer", "autumn") -) months => c("DJF", "MAM", "JJA", "SON") or c("DJFM", "AM", "JJAS", "ON")
type	character, indicating which weather seasons will be used for computing the output. Possible values are: -) default => "winter"= Dec, Jan, Feb; "spring"= Mar, Apr, May; "summer"=Jun, Jul, Aug; "autumn"= Sep, Oct, Nov -) FrenchPolynesia => "winter"= Dec, Jan, Feb, Mar; "spring"= Apr, May; "summer"=Jun, Jul, Aug, Sep; "autumn"= Oct, Nov

## Value

character vector with the weather season to which each date in x belongs

**Note**

Weather seasons corresponding to French Polynesia were defined following a comment from Lydie Sichoix

**Author(s)**

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

**See Also**

[dm2seasonal](#), [seasonalfunction](#), [extract](#), [dip](#), [mip](#)

**Examples**

```
## Sequence of daily dates between "1961-01-01" and "1961-12-31"
t <- dip("1961-01-01", "1961-12-31")
time2season(t)

## Sequence of monthly dates between "1961-01-01" and "1961-12-31"
t <- mip("1961-01-01", "1961-12-31")
time2season(t)
time2season(t, out.fmt="seasons")
```

---

vector2zoo

---

*Vector -> Zoo*


---

**Description**

Transform a numeric vector and its corresponding dates into a zoo object.

**Usage**

```
vector2zoo(x, dates, date.fmt = "%Y-%m-%d")
```

**Arguments**

x	numeric vector
dates	character, factor, Date or POSIXct object with the dates corresponding to each element of x. Valid object class for dates are: character, factor, Date, POSIXct
date.fmt	character indicating the format in which the dates are stored in <i>dates</i> , e.g. %Y-%m-%d. See ‘Details’ section in <a href="#">strptime</a> . ONLY required when <code>class(dates)=="factor"</code> or <code>class(dates)=="character"</code> .

**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)
```

---

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 <a href="#">as.Date</a> .
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. -) nmbrr => 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) nmbrr: 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 = "nmbrr")
```

---

zoo2RHtest

*Zoo -> RHTest*

---

**Description**

It creates the input file to the 'RHtest\_dlyPrp.r' script, used for testing the homogeneity of climatological time series (available at: <http://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**

<http://etccdi.pacificclimate.org/software.shtml> [http://etccdi.pacificclimate.org/RHtest/RHtestsV3\\_UserManual.doc](http://etccdi.pacificclimate.org/RHtest/RHtestsV3_UserManual.doc) <http://css.escwa.org.lb/sdpd/1802/d2-1.pdf>

**Examples**

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

#Getting the monthly ts
pcp.m <- daily2monthly(x, FUN=sum, na.rm=FALSE)

# 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")

## Not run:
# 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

## \*Topic **datasets**

EbroCatchmentsCHE, 21  
EbroDEM1000m, 22  
EbroPPgis, 23  
EbroPPtsMonthly, 23  
KarameaAtGorgeQts, 56  
OcaEn0naQts, 67  
SanMartinoPPts, 69

## \*Topic **graphs**

fdc, 25  
fdcu, 29  
hydropairs, 45  
hydroplot, 46  
hypsometric, 50  
matrixplot, 58  
mspplot, 63

## \*Topic **manip**

(sub)daily2annual, 4  
(sub)daily2monthly, 7  
annualfunction, 9  
dip, 11  
diy, 13  
dm2seasonal, 14  
dwdays, 18  
dwi, 19  
extract, 24  
fdc, 25  
fdcu, 29  
gists2spt, 33  
hip, 35  
hydroplot, 46  
hypsometric, 50  
infillxy, 52  
istdx, 53  
izoo2rzoo, 54  
ma, 57  
mip, 59  
monthlyfunction, 60  
rm1stchar, 68

seasonalfunction, 69  
sfreq, 72  
smry, 73  
sname2ts, 75  
stdx, 76  
subdaily2daily, 77  
time2season, 79  
vector2zoo, 80  
yip, 81  
zoo2RHtest, 82

## \*Topic **math**

hydrokrige, 36  
hydropairs, 45

## \*Topic **package**

hydroTSM-package, 3  
(sub)daily2annual, 4  
(sub)daily2monthly, 7

annualfunction, 6, 9, 62, 71  
as.Date, 5, 6, 8, 10, 12, 15, 17, 19, 20, 35, 48,  
60, 61, 70, 75, 78, 82  
as.POSIXct, 78  
autofitVariogram, 41  
autoKrige, 36, 41, 42, 65  
Axis, 27, 31

cor, 45, 46  
cor.test, 45

daily2annual, 8, 10, 16, 25, 62  
daily2annual ((sub)daily2annual), 4  
daily2monthly, 6, 16, 25, 62  
daily2monthly ((sub)daily2monthly), 7  
dip, 11, 13, 35, 60, 80–82  
diy, 12, 13, 35, 60, 82  
dm2seasonal, 14, 62, 71, 78, 80  
drawTimeAxis, 16  
drawxaxis (drawTimeAxis), 16  
dwdays, 18  
dwi, 19, 59

- EbroCatchmentsCHE, 21
- EbroDEM1000m, 22
- EbroPPgis, 23
- EbroPPtsMonthly, 23
- extract, 16, 24, 71, 80
- extractzoo (extract), 24
- fdc, 25, 32
- fdcu, 28, 29
- fivenum, 74
- frequency, 72
- gists2spt, 33
- hip, 12, 13, 35, 60
- hydrokrige, 36
- hydropairs, 45
- hydroplot, 6, 8, 16, 46, 78
- hydroTSM (hydroTSM-package), 3
- hydroTSM-package, 3
- hypsometric, 50
- infillxy, 52
- IQR, 74
- istdx, 53, 77
- izoo2rzoo, 8, 54, 78, 81
- KarameaAtGorgeQts, 56
- krige, 34, 38, 39, 42, 65
- kurtosis, 74
- legend, 27, 31
- levelplot, 58
- ma, 57
- matrixplot, 20, 58
- mip, 12, 13, 35, 59, 80–82
- monthly2annual, 6, 10, 16
- monthly2annual ((sub)daily2annual), 4
- monthlyfunction, 8, 10, 60, 71, 78
- mspplot, 63
- OcaEn0naQts, 67
- pairs, 45, 46
- par, 17, 27, 31, 48
- plot, 27, 30, 48, 50
- plot.default, 27, 30, 31, 48, 50
- points, 27, 31
- polygon, 31, 32
- readGDAL, 38, 50
- readShapePoly, 39, 42, 64
- rm1stchar, 68
- SanMartinoPPts, 69
- scale, 53, 77
- sd, 74
- seasonalfunction, 16, 25, 62, 69, 78, 80
- sfreq, 72
- skewness, 74
- smry, 73
- sname2plot, 76
- sname2plot (hydroplot), 46
- sname2ts, 49, 75
- SpatialGridDataFrame-class, 38, 50
- SpatialPixelsDataFrame-class, 41, 64
- SpatialPointsDataFrame-class, 33, 34
- SpatialPolygonsDataFrame-class, 21, 39, 41, 64
- spplot, 34, 40, 65
- spsample, 39, 42
- stdx, 53, 76
- strptime, 55, 80
- subdaily2annual, 78
- subdaily2annual ((sub)daily2annual), 4
- subdaily2daily, 8, 77
- subdaily2monthly, 78
- subdaily2monthly ((sub)daily2monthly), 7
- subdaily2seasonal, 78
- subdaily2seasonal (dm2seasonal), 14
- substr, 68
- summary, 74
- time2season, 16, 25, 71, 79
- timeBasedSeq, 35
- title, 27, 30, 50
- vector2zoo, 6, 8, 55, 78, 80
- write.table, 41
- yip, 10, 12, 13, 35, 60, 81, 81
- zoo, 81
- zoo2RHtest, 82