Package ‘discharge’

October 13, 2022

Type Package

Title Fourier Analysis of Discharge Data

Version 1.0.0

Encoding UTF-8

Date 2019-03-06

Description Computes discrete fast Fourier transform of river discharge data and the derived metrics. The methods are described in J. L. Sabo, D. M. Post (2008) <doi:10.1890/06-1340.1> and J. L. Sabo, A. Ruhi, G. W. Holtgrieve, V. Elliott, M. E. Arias, P. B. Ngor, T. A. Räsänsen, S. Nam (2017) <doi:10.1126/science.aao1053>.

Imports lmom, ggplot2, CircStats, checkmate, boot

Depends R (>= 3.0.2)

License GPL-3

LazyData TRUE

RoxygenNote 6.1.1

NeedsCompilation no

Author Samarth Shah [aut, cre],
Albert Ruhi [aut],
Future H2O [cph]

Maintainer Samarth Shah <sbshah10@asu.edu>

Repository CRAN

Date/Publication 2019-03-08 15:42:48 UTC

R topics documented:

`allstats` *Calculate all characteristic stats for a site*

Description

Calculates all parameters at once for a single data file. The output gives numerical results from the functions `annualnoise`, `fourierAnalysis`, `lp3Events`,`sigmaHighFlows`, and `sigmaLowFlows`.

Usage

```
allstats(file.name, river.name, file.type="txt", date.col=3,
discharge.col=4, skipped.rows=28)
```

Arguments

`file.name`	Character string of the form "file.txt" or "file.csv".
`river.name`	Character string specifying river name.
`file.type`	Character string, "txt" or "csv". Defaults to "txt".
`date.col`	Numeric specifying column containing date in "MM-DD-YYYY" format. Defaults to 3.
`discharge.col`	Numeric specifying column containing discharge data. Defaults to 4.
`skipped.rows`	Numeric indicating number of rows to skip at beginning of file.

Value

A data frame with columns

`a.rms`	Root mean squared amplitude.
`n.rms`	Root mean squared noise.
`snr`	Signal-to-noise ratio.
`theta.d`	Daily noise color.
`theta.a`	Annual noise color.
`sigma.lf`	Sigma for low flow events.
`sigma.hf`	Sigma for high flow events.
`q2`	2-year return level (flood).
`q10`	10-year return level (flood).
`l2`	2-year return level (drought).
`l10`	10-year return level (drought).

Note

The arguments "date.col", "discharge.col", and "skipped.rows" are designed to give some flexibility in file input; however, tab-delimited text without extra columns will likely work best.

See Also

`parameters.list`

Examples

```
# allstats function works on files
# read R data into temporary file handle
data(sycamore)
f = tempfile(fileext="txt")
write.table(sycamore, file=f, sep="\t")

# print all statistics for this river
allstats(f,river.name="sycamore", date.col = 2, discharge.col = 3, skipped.rows = 1)
```

annualExtremes *Calculate annual extreme flows.*

Description

Calculates annual extreme events for all years in the series. By default, the function finds annual extreme discharge in `streamflow` object, but any matrix or `data.frame` may be used.

Usage

```
annualExtremes(x,data.col=NULL, year.col=NULL, moving.avg=FALSE)
```

Arguments

`x`	Object from which to extract extremes. Should be of class `streamflow` or `data.frame` or matrix.
`data.col`	Optional. If input is a matrix or `data.frame`, specifies which column contains the data.
`year.col`	Optional. If input is a matrix or `data.frame`, specifies which column contains the year.
`moving.avg`	Logical; defaults to FALSE. Can be specified TRUE to use 7-day moving average discharge when input is of class "`streamflow`".

Value

A `list` with items

`annual.max`	Matrix giving maximum flow for each year in series. Each row contains the maximum values and all corresponding variables from that observation.
`annual.min`	Matrix giving minimum flow for each year in series. Each row contains the minimum values and all corresponding variables from that observation.

Examples

```
data(sycamore)
sycamore.flows<-asStreamflow(sycamore,river.name="Sycamore Creek")
syc.extremes<-annualExtremes(sycamore.flows)
names(syc.extremes)
syc.extremes$annual.max[1:3,]
```

`annualnoise` *Annual noise color*

Description

The autocovariance function is estimated for the annual maxima in the series. An autoregressive model of the order of the highest significant lag is fit, using the Yule-Walker method to estimate the parameters. The function is transformed into the frequency domain, yielding an estimate `theta.a` of the annual noise color.

Usage

```
annualnoise(x)
```

Arguments

`x`	A numeric vector of annual extremes. A `streamflow` object may also be used. If input is `streamflow` the function uses annual maximum discharge.

Details

To determine the order of the AR model, the ACF is calculated at all lags less than or equal to the highest power of 2 less than the length of the series. The order of the AR model is the lag with the highest significantly non-zero autocorrelation.

Value

An object of S3 class `annualnoise` with the following attributes:

`auto.corr`	Sample autocorrelation.
`lm.fit`	`lm` object from regression of log power spectrum on log frequency.
`interval`	Upper and lower bounds of a 95% acceptance region when $\rho = 0$.
`log.log`	Matrix with log frequency and log power spectrum.
`reg.stats`	Slope and intercept of regression of log power spectrum on log frequency, where slope is the annual noise color (`theta.a`).
`order`	Indicates order of fitted AR model.
`fit.ar`	Object of class `ar` summarizing the fitted AR model.

Examples

```
data(sycamore)
sycamore.flows<-asStreamflow(sycamore,river.name="Sycamore Creek")
syc.ar<-annualnoise(sycamore.flows)
```

`assert.equal.length` *Check for equal length inputs*

Description

Ensure that the given arguments are equal length vectors

Usage

```
assert.equal.length(arg0, ...)
```

Arguments

`arg0`	At least one input argument to be checked needs to be given
`...`	Other inputs whose length equality needs to be checked

Value

Stops execution in case of failed check

`assert.for.year` *Check* `for.year` *argument*

Description

Ensure that the for.year argument is a numeric scalar value

Usage

```
assert.for.year(for.year)
```

Arguments

`for.year`	input argument to be checked

Value

Stops execution in case of failed check

`assert.numeric.vector` *Check for numeric vector*

Description

Ensure that the given argument is a numeric vector

Usage

```
assert.numeric.vector(arg)
```

Arguments

`arg`	input argument to be checked

Value

Stops execution in case of failed check

`asStreamflow` *Create streamflow object*

Description

This function converts a dataset to S3 object of class "streamflow". Data in this format can be provided as arguments to other functions to call default procedures.

Usage

```
asStreamflow(x,river.name=NULL, start.date=NULL, end.date=NULL,max.na=10)
## S3 method for class 'streamflow'
print(x, ...)
## S3 method for class 'streamflow'
summary(object, ...)
```

Arguments

`x,object`	A matrix with dates in the first column and discharge values in the second column. Dates should be of the format "YYYY-MM-DD".
`river.name`	A character vector listing the river name.
`start.date`	Optional character string giving date to start analysis, of the format "YYYY-MM-DD"
`end.date`	Optional date to start analysis, of the format "YYYY-MM-DD"
`max.na`	Optional number specifying maximum NA values to allow.
`...`	Other parameters.

Value

An object of class `streamflow` containing the following items:

- `data` Data frame with 8 columns
- `n` Number of observations in series.
- `n.nas` Number of NA observations in series.
- `start` Start date.
- `end` End date.
- `name` Name of river.
- `na.info` Matrix containing the index and start date of all blocks of more than one NA observation.

Examples

```
data(sycamore)
sycamore.flows<-asStreamflow(sycamore,start.date="1965-01-01",
                            end.date="2010-03-15",river.name="Sycamore Creek")
```

---

circ.s *Estimate directional statistics for one-sigma events*

Description

Uses the package "CircStats" to find the mean direction, rho, and kappa of the von Mises distribution summarizing the ordinal day of high- and low-flow events.

Usage

```
circ.s(x)
```

Arguments

- `x` Output from "compare.periods" function (of class "`compflows`").

Value

- `circstats` `data.frame` with rows corresponding to the high- and low-flows for both periods. The columns list n, mu, rho, and kappa as calculated using the CircStats package.

See Also

`compare.periods`

Examples

```
# load data
data("sycamore")

# compare for periods from 1960 to 1979 and 1980 to 1999
comp = compare.periods(c("1960-01-01", "1979-12-31"),
c("1980-01-01", "1999-12-31"), sycamore, plot=FALSE)

circ.s(comp)
```

---

compare.periods *Compare residual variability across time periods*

Description

The function finds the seasonal signal in the first of two time periods within the data series. This signal is used to calculate residual flows in the first time period and "historical residuals", the residuals when the second period is compared to the signal in the first period.

The output of the function gives event information for all residual events greater than σ as calculated from the functions `sigmaHighFlows` and `sigmaLowFlows`.

Usage

```
compare.periods(p1, p2, x, plot=T)
## S3 method for class 'compflows'
 plot(x, ...)
```

Arguments

p1	Character vector specifying start and end date of the first period. "YYYY-MM-DD" format.
p2	Character vector specifying start and end date of the second period. "YYYY-MM-DD" format.
x	Matrix with first column specifying dates and second column specifying raw discharge data.
plot	Logical; defaults to TRUE. If TRUE, the seasonal signal for both periods will be plotted.
...	Other parameters.

Value

Object of S3 ckass compflows with the following items:

sigma.low	Estimate of $\sigma - lf$ from period 1
sigma.high	Estimate of $\sigma - hf$ from period 1
p1.levents	Matrix of low flow events for period 1
p1.hevents	Matrix of high flow events for period 1
p2.levents	Matrix of low flow events for period 2
p2.hevents	Matrix of high flow events for period 2

See Also

sigmaHighFlows sigmaLowFlows

Examples

```
# load data
data("sycamore")

# compare for periods from 1960 to 1979 and 1980 to 1999
compare.periods(c("1960-01-01", "1979-12-31"),
c("1980-01-01", "1999-12-31"), sycamore)
```

---

etowah *Etowah River Data*

Description

A data series from USGS station 02392000, Etowah River (Canton,GA).

Usage

```
data(etowah)
```

Format

The data are supplied as a matrix with years in the first column and average daily discharge in the second column. The series contains 18282 observations starting Jan 1, 1960 and ending Dec 31, 2009.

Source

- USGS National Water Information Sysem: http://waterdata.usgs.gov/usa/nwis/uv?site_no=02392000

`fftmetrics` *Summary of spectral anomolies for a specified year.*

Description

This function takes as input a streamFlow object and a year, and outputs the timing and magnitude of noteworthy spectral anomolies.

Usage

```
fftmetrics(x,year,candmin,candmax)
```

Arguments

`x`	streamflow object, as output from the asStreamflow function.
`year`	integer; data from this year will be analyzed. The plotted hydrograph will include data from August of the previous year to April of the following year.
`candmin`	numeric vector of possible ordinal days in which the predicted signal is lowest. This range need not be narrow, but a string of consecutive days should not include more than only local minimum. Used for calculating the high- and low-flow windows.
`candmax`	numeric vector of possible ordinal days in which the predicted signal is highest. This range need not be narrow, but a string of consecutive days should not include more than only local maximum.

Value

A list object with the following components;

`sam`	dataframe where row 1 corresponds to the largest low-residual event in the year, and row 2 corresponds to the largest high-residual. The dataframe contains each event's date, ordinal day, magnitude of residual and signal, index in the original data, and timing (in days) relative to the reference point.
`ref.point`	date used as a reference point for the timing of max and min events. If not given by the user, it is the first local maximum of the signal, within the year specified.
`events`	dataframe with rows corresponding to start dates of the longest low-flow and high-flow events. The data frame contains signal and residual data for the start date and the lenght of the run.
`auc`	dataframe with three values: 'net.auc', 'rel.auc.low', and rel.auc.high'. 'net.auc' is the sum of positive residuals in the high flow window divided by the sum of negative residuals in the low flow window. 'rel.auc.low' is postive residuals divided by the sum of negative residuals
`noise.color`	numeric, "theta.d" as calculated as in the fourierAnalysis function.

Examples

```
data(etowah)
etowah.flows=asStreamflow(etowah, river.name="Etowah")
# "candmax" chosen because preliminary analysis (e.g., with fourierAnalysis output)
# shows the signal is highest sometime between day 40 and day 125.
# "candmin" can be estimated analogously.
fftmetrics(etowah.flows,2002,candmin=c(190:330),candmax=c(40:125))
```

---

fft_metrics — *Discrete Fourier Transform Metrics*

Description

This is a wrapper function to calculate all the DFFT metrics for the given input signal

Usage

```
fft_metrics(data, candmin, candmax, river.name = "",
  baseline.signal = NULL)
```

Arguments

`data`	A matrix with dates in the first column and discharge values in the second column. Dates should be of the format "YYYY-MM-DD"
`candmin`	numeric vector of possible ordinal days in which the predicted signal is lowest. This range need not be narrow, but a string of consecutive days should not include more than only local minimum. Used for calculating the high- and low-flow windows
`candmax`	numeric vector of possible ordinal days in which the predicted signal is highest. This range need not be narrow, but a string of consecutive days should not include more than only local maximum.
`river.name`	A character vector listing the river name.
`baseline.signal`	If `NULL`, this function calculates baseline.signal using fourierAnalysis over the entire input series. The baseline signal can also be explicitly calculated and passed in as parameter. Check function `prepareBaseline()`

Value

A list containing 2 data frames:

`high.level.metrics`	Data frame containing NAA and FPExt values for each year in the given series
`naa.shape.components`	Data frame containing HSAM, LSAM, Transition time, HSAF, LSAF, timing of HSAM, timing of

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# fetch the DFFT metrics for this sample data
# "candmax" chosen because preliminary analysis (e.g. with fourierAnalysis
#           output) shows the signal is highest sometime between
#           day 190 and day 330
# "candmin" can be estimated analogously.
x.fftmetrics = fft_metrics(x, river.name = "Sycamore", candmin = c(40:125),
                           candmax = c(190:330), baseline.signal = x.bl)
```

`filterBaseline` *Filter the baseline signal for a given time window*

Description

Filter the baseline signal for a given time window

Usage

```
filterBaseline(bl, filter.date.start, filter.date.end,
  date.format = "%Y-%m-%d")
```

Arguments

`bl`	baseline signal as returned from the function `prepareBaseline()`
`filter.date.start`	start date of the filtering window
`filter.date.end`	end date of the filtering window
`date.format`	format of date specified in `filter.date.start` and `filter.date.end`

Value

baseline signal filtered for the given date window

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# baseline for single run for all the years in input signal
x.bl = prepareBaseline(x.streamflow)

# filter the baseline signal between years 1993 and 2000
x.bl.filtered = filterBaseline(x.bl, filter.date.start = "1993-01-01",
                               filter.date.end = "2000-12-31")
```

`findMed` *Find median value*

Description

Find median value from the filtered part of the input vector. This function is used as a helper to the bootstrapping routine in prepare baseline

Usage

```
findMed(data, indices)
```

Arguments

`data`	complete input vector
`indices`	logical vector to filter indices from `data`

Value

median value of the filtered vector

`fourierAnalysis` *Extract seasonal signal from time series*

Description

The fast Fourier transform is used to extract the seasonal signal of a time series. The significant frequencies are found from among periods of length 2-, 3-, 4-, 6-, 12-, and 18-months.

The signal may be specified as stationary or non-stationary. If a non-stationary fit is allowed, simple linear regression estimates the long term linear trend. The seasonal signal is calculcated from the residuals.

Predicted flow (and corresponding residual) at each time point is calculated from seasonal signal and, if non-stationary, long term trend coefficient.

Usage

```
fourierAnalysis(x, stationary=F)
## S3 method for class 'ssignal'
 plot(x, plot.type="hydrograph", ...)
```

Arguments

`x`	An object of class `streamflow`
`stationary`	Logical; defaults to FALSE.
`plot.type`	Indicates the type of plot to create. The default "hydrograph" produces a plot of ordinary day and log normalized discharge, with the seasonal signal overlaid. "auto.corr" produces a plot of daily autocorrelation as calculated from the residual flows.
`...`	Other parameters.

Value

An object of class `ssignal` with items

`signal`	Data matrix augmented to included predicted and residual values.
`terms`	Matrix containing amplitude, phase, and frequency of seasonal signal.
`detrend.fit`	An `lm` object from regression of discharge on index of observation.
`logps.regression`	An `lm` object from regression of log power spectrum on log frequency (where log frequencies have seasonal signal removed.)
`rms`	`list` containing RMS amplitude for noise, RMS amplitude for signal, and signal-to-noise ratio.

Examples

```
data(sycamore)
sycamore.flows<-asStreamflow(sycamore,river.name="Sycamore Creek")
syc.seas<-fourierAnalysis(sycamore.flows)
summary(syc.seas)
```

`getFPExt` *Flood Pulse Extent (FPExt)*

Description

Calculate the Flood Pulse Extent (FPExt) from the given residual values.

Usage

```
getFPExt(resid, years, for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`for.year`	(optional) Calculate FPExt values only for the given year in this argument. If argument is omitted, NAA values for all years are calculated.

Value

Data frame containing two columns:

`year`	First column, represents year
`FPExt`	Second column, represents FPExt values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# FPExt
fpext = getFPExt(x.bl$resid.sig, x.streamflow$data$year)
```

getHSAF *High Spectral Anomaly Frequency (HSAF)*

Description

Compute High Spectral Anomaly Frequency (HSAF) from the given residual values.

Usage

```
getHSAF(resid, years, for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`for.year`	(optional) Calculate HSAF values only for the given year in this argument. If argument is omitted, HSAF values for all years are calculated.

Value

Data frame containing two Columns:

`year`	First column, represents year
`HSAF`	Second column, represents HSAF values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# HSAF
hsaf = getHSAF(x.bl$resid.sig, x.streamflow$data$year)
```

getHSAM *High Spectral Anomaly Mangitude (HSAM)*

Description

Compute High Spectral Anomaly Magnitude (HSAM) from the given residual values each year

Usage

```
getHSAM(resid, years, for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`for.year`	(optional) Calculate HSAM values only for the given year in this argument. If argument is omitted, HSAM values for all years are calculated.

Value

Data frame containing four columns:

`year`	First column, represents year
`HSAM`	Second column, represents HSAM values
`index.year`	Third column, representing index of HSAM value in that `year`
`index.all`	Fourth column, representing index of HSAM value in the input `resid`

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# HSAM
hsam = getHSAM(x.bl$resid.sig, x.streamflow$data$year)
```

`getIDI` *Inter-Draught Interval (IDI)*

Description

Compute Inter-Draught Interval (IDI) from the given residual values.

Usage

```
getIDI(resid, years, highflow.start, highflow.end, unique.years,
  for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`highflow.start`	A vector giving start index of high-flow window in each year
`highflow.end`	A vector giving end index of high-flow window in each year
`unique.years`	A vector or year values corresponding to the `highflow.start` and `highflow.end` values.
`for.year`	(optional) Calculate IDI values only for the given year in this argument. If argument is omitted, IDI values for all years are calculated.

Value

Data frame containing two columns:

`year`	First column, represents year
`IDI`	Second column, represents IDI values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# signal parts
x.sp = getSignalParts(x.bl$pred2, candmin = c(40:125), candmax = c(190:330),
                      years = x.streamflow$data$year,
                      months = x.streamflow$data$month,
                      jdays = x.streamflow$data$jday)

# IDI
idi = getIDI(x.bl$resid.sig, x.streamflow$data$year, x.sp$HF.window.start,
             x.sp$HF.window.end, x.sp$year)
```

---

getIFI *Inter-Flood Interval (IFI)*

Description

Compute Inter-Flood Interval (IFI) from the given residual values.

Usage

```
getIFI(resid, years, lowflow.start, lowflow.end, unique.years,
  for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`lowflow.start`	A vector giving start index of low-flow window in each year
`lowflow.end`	A vector giving end index of low-flow window in each year
`unique.years`	A vector or year values corresponding to the `highflow.start` and `highflow.end` values.
`for.year`	(optional) Calculate IFI values only for the given year in this argument. If argument is omitted, IFI values for all years are calculated.

Value

Data frame containing two columns:

`year`	First column, represents year
`IFI`	Second column, represents IFI values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# signal parts
x.sp = getSignalParts(x.bl$pred2, candmin = c(40:125), candmax = c(190:330),
                      years = x.streamflow$data$year,
                      months = x.streamflow$data$month,
                      jdays = x.streamflow$data$jday)

# IFI
ifi = getIFI(x.bl$resid.sig, x.streamflow$data$year, x.sp$LF.window.start,
             x.sp$LF.window.end, x.sp$year)
```

`getLSAF` *Low Spectral Anomaly Frequency (LSAF)*

Description

Compute Low Spectral Anomaly Frequency (LSAF) from the given residual values.

Usage

```
getLSAF(resid, years, for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`for.year`	(optional) Calculate LSAF values only for the given year in this argument. If argument is omitted, LSAF values for all years are calculated.

Value

Data frame containing two Columns:

`year`	First column, represents year
`LSAF`	Second column, represents LSAF values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# LSAF
lsaf = getLSAF(x.bl$resid.sig, x.streamflow$data$year)
```

---

getLSAM — *Low Spectral Anomaly Mangitude (LSAM)*

Description

Compute Low Spectral Anomaly Magnitude (LSAM) from the given residual values each year

Usage

```
getLSAM(resid, years, for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`for.year`	(optional) Calculate LSAM values only for the given year in this argument. If argument is omitted, LSAM values for all years are calculated.

Value

Data frame containing four columns:

`year`	First column, represents year
`LSAM`	Second column, represents LSAM values
`index.year`	Third column, representing index of LSAM value in that `year`
`index.all`	Fourth column, representing index of LSAM value in the input `resid`

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# LSAM
lsam = getLSAM(x.bl$resid.sig, x.streamflow$data$year)
```

`getNAA` *Net Annual Anomaly (NAA)*

Description

Calculate Net Annual Anomaly (NAA) from the given residual values.

Usage

```
getNAA(resid, years, for.year = NULL)
```

Arguments

`resid`	A vector of residual values generated with respect to the baseline signal
`years`	A vector of years corrosponding to the residual values
`for.year`	(optional) Calculate NAA values only for the given year in this argument. If argument is omitted, NAA values for all years are calculated.

Value

Data frame containing two columns:

year	First column, represents year
NAA	Second column, represents NAA values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# NAA
naa = getNAA(x.bl$resid.sig, x.streamflow$data$year)
```

---

getSignalParts *Signal parts*

Description

This function computes high flow and low flow window of seasonal signal, and the peak max and peak min values.

Usage

```
getSignalParts(seas.sig, candmin, candmax, years, months, jdays,
  for.year = NULL)
```

Arguments

seas.sig	Seasonal signal as generated from DFFT methods
candmin	numeric vector of possible ordinal days in which the predicted signal is lowest. This range need not be narrow, but a string of consecutive days should not include more than one local minimum. Used for calculating the high- and low-flow windows.
candmax	numeric vector of possible ordinal days in which the predicted signal is highest. This range need not be narrow, but a string of consecutive days should not include more than one local maximum.
years	A vector of years corrosponding to the seasonal signal values
months	A vector of months corrosponding to the seasonal signal values

`jdays`	A vector of julian days corrosponding to the seasonal signal values
`for.year`	(optional) Calculate signal parts only for the given year in this argument. If argument is omitted, all years are considered.

Value

Data frame containing following columns.

`year`	represents year
`max.peak.index.all`	represents index value within the entire vector
`max.peak.value`	represents value of max peak
`highwind.start.index.all`	start index of high flow window within the entire vector
`highwind.end.index.all`	end index of high flow window within the entire vector
`lowwind.start.index.all`	start index of low flow window within the entire vector
`lowwind.end.index.all`	end index of low flow window within the entire vector

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# signal parts
x.sp = getSignalParts(x.bl$pred2, candmin = c(40:125), candmax = c(190:330),
                      years = x.streamflow$data$year,
                      months = x.streamflow$data$month,
                      jdays = x.streamflow$data$jday)
```

`getTimingHSAM` *Time of occurence of High Spectral Anomaly Magnitude (HSAM)*

Description

Compute the number of days separating HSAM and reference point for each year.

Usage

```
getTimingHSAM(index.hsam, index.ref, years, for.year = NULL)
```

Arguments

`index.hsam`	A scalar/vector of index of HSAM values in given year/years
`index.ref`	A scalar/vector of index of reference point in given year/years
`years`	A vector of years corresponding to HSAM and ref values. This argument can be NULL if the HSAM and ref values are scalars.
`for.year`	(optional) Calculate timing (HSAM) only for the given year in this argument. If argument is omitted, timing (HSAM) values for all years are calculated.

Value

Scalar timing HSAM value if the inputs are scalars, or a Data frame containing two Columns:

`year`	First column, represents year
`timing.hsam`	Second column, represents hsam timing values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# get signal parts
x.sp = getSignalParts(x.bl$pred2, candmin = c(40:125), candmax = c(190:330),
                      years = x.streamflow$data$year,
                      months = x.streamflow$data$month,
                      jdays = x.streamflow$data$jday)

# get HSAM values
hsam = getHSAM(x.bl$resid.sig, x.streamflow$data$year)

# timing HSAM
thsam = getTimingHSAM(hsam$Index.all, x.sp$peak.index, x.sp$year)
```

---

getTimingLSAM *Time of occurence of Low Spectral Anomaly Magnitude (LSAM)*

Description

Compute the number of days separating LSAM and reference point for each year.

Usage

```
getTimingLSAM(index.lsam, index.ref, years = NULL, for.year = NULL)
```

Arguments

index.lsam	A scalar/vector of index of LSAM values in given year/years
index.ref	A scalar/vector of index of reference point in given year/years
years	(optional) A vector of years corresponding to LSAM and ref values. This argument can be NULL if the LSAM and ref values are scalars.
for.year	(optional) Calculate timing (LSAM) only for the given year in this argument. If argument is omitted, timing (LSAM) values for all years are calculated.

Value

Scalar timing LSAM value if the inputs are scalars, or a Data frame containing two Columns:

year	First column, represents year
timing.lsam	Second column, represents lsam timing values

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# get signal parts
x.sp = getSignalParts(x.bl$pred2, candmin = c(40:125), candmax = c(190:330),
                      years = x.streamflow$data$year,
                      months = x.streamflow$data$month,
                      jdays = x.streamflow$data$jday)

# get LSAM values
lsam = getLSAM(x.bl$resid.sig, x.streamflow$data$year)

# timing LSAM
tlsam = getTimingLSAM(lsam$Index.all, x.sp$peak.index, x.sp$year)
```

---

getTransitionTime *Transition Time*

Description

Compute the number of days separating HSAM and LSAM for the given year/years.

Usage

```
getTransitionTime(index.hsam, index.lsam, years, for.year = NULL)
```

Arguments

`index.hsam`	A scalar/vector of index of HSAM values in given year/years
`index.lsam`	A scalar/vector of index of LSAM values in given year/years
`years`	A vector of years corresponding to HSAM and LSAM values. This argument can be NULL if the HSAM and LSAM values are scalars.
`for.year`	(optional) Calculate transition time only for the given year in this argument. If argument is omitted, transition times for all years are calculated.

Value

Scalar transition time if the inputs are scalars, or a Data frame containing two Columns:

`year`	First column, represents year
`transition.time`	Second column, represents transition times

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)

# prepare baseline signal
x.bl = prepareBaseline(x.streamflow)

# get HSAM and LSAM values
hsam = getHSAM(x.bl$resid.sig, x.streamflow$data$year)
lsam = getLSAM(x.bl$resid.sig, x.streamflow$data$year)

# transition time
tt = getTransitionTime(hsam$Index.all, lsam$Index.all, hsam$year)
```

`independentEvents` *Independent high- or low-flow events*

Description

Finds independent events greater than or less than a specified criterion. High (or low) -flow days occurring on consecutive days are considered part of one event. This function can be used to find events exceeding 2- or 10- year return levels (as calculated in `lp3Events` function, for example), or to find residual flows of a certain magnitude.

Usage

```
independentEvents(cutoff.val, data, data.column, below.cutoff=FALSE)
```

Arguments

`cutoff.val`	Numeric specifying event criterion.
`data`	Data matrix or data frame with one column of streamflow data.
`data.column`	Numeric; specifies column in which to look for events.
`below.cutoff`	Logical. TRUE to find events less than the `cutoff.val` and FALSE to find events greater than the `cutoff.val`.

Value

A `data.frame` with columns

`events.starts`	Index of event start.
`events.ends`	Index of event end.
`events.duration`	Length (days) of event.
`extreme.this.events`	Maximum or minimum flow for this event.
`ind.extreme`	Index of maximum or minimum flow for this event. If extreme is not unique, the chronologically first index is given.
`...`	All columns of original data, corresponding to max or min flow. These columns will have the same column names as the original data.
`duplicates`	0 if the extreme is unique, 1 if it is not unique.

See Also

`lp3Events`

Examples

```
data(sycamore)
syc.sf<-asStreamflow(sycamore)
#find 10-year flood
q10<-lp3Events(syc.sf)$Q10

#find all events greater than 10-year flood
independentEvents(q10,syc.sf$data, data.col=8 , below.cutoff=FALSE)
```

`lp3Events` *Find 2- and 10-year return levels*

Description

Uses the method of moments to find the 2- and 10-year droughts and floods under the Log-III Pearson distribution.

Usage

```
lp3Events(x)
```

Arguments

`x`	Object of class "`streamflow`".

Details

Return levels are calculated using the method of moments through the package "lmom". High return levels (floods) are calculated using annual maxima of the raw (log normalized) data, while the low return levels (droughts) are calculated using annual mimima of the 7-day moving averages.

Value

A `list` with items

`Q2`	2-year high return level.
`Q10`	10-year high return level.
`L2`	2-year low return level.
`L10`	10-year low return level.

Examples

```
data(sycamore)
syc.sf<-asStreamflow(sycamore)
lp3Events(syc.sf)
```

parameters.list *Calculate all characteristic stats from a list of files*

Description

Function takes a vector of file names and returns seasonal signal-to-noise ratio, daily and annual noise color, 2- and 10-year return levels, and σ for high- and low-flow events.

All files in the vector should be of the same format.

Usage

```
parameters.list(x, names=NULL, file.type="txt", date.col=3, dis.col=4, skipped.rows=28)
```

Arguments

x	Character vector containing file names.
names	Optional character vector with names of sites.
file.type	Character string, "txt" or "csv". Defaults to "txt".
date.col	Numeric specifying column containing date in "MM-DD-YYYY" format. Defaults to 3.
dis.col	Numeric specifying column containing discharge data. Defaults to 4.
skipped.rows	Numeric indicating number of rows to skip at beginning of file.

Value

A data frame with one row for each file and the following columns:

a.rms	Root mean squared amplitude.
n.rms	Root mean squared noise.
snr	Signal-to-noise ratio.
theta.d	Daily noise color.
theta.a	Annual noise color.
sigma.lf	Sigma for low flow events.
sigma.hf	Sigma for high flow events.
q2	2-year return level (flood).
q10	10-year return level (flood).
l2	2-year return level (drought).
l10	10-year return level (drought).

Note

The arguments "date.col", "discharge.col", and "skipped.rows" are designed to give some flexibility in file input; however, tab-delimited text without extra columns will work best.

See Also

allstats

Examples

```
# this function works on list of files
# read R data into temporary file handle
data(sycamore)
f = tempfile(fileext="txt")
write.table(sycamore, file=f, sep="\t")

# print all statistics for the list of rivers
parameters.list(c(f), names=c("sycamore"), date.col=2,dis.col=3,skipped.rows = 1)
```

---

prepareBaseline — *Build baseline signal*

Description

Runs fourier analysis on the input signal, to build baseline signal.

Usage

```
prepareBaseline(x, year.start = NULL, year.end = NULL,
  window.20 = FALSE)
```

Arguments

`x`	streamflow object, as output from the `asStreamflow()` function
`year.start`	Start of the year for estimating baseline, or `NULL` to interpret this from input data
`year.end`	End of the year for estimating baseline, or `NULL` to interpret this from input data
`window.20`	If `TRUE`, baseline is constructed using windowing (20 year windows) and bootstrapping. If `FALSE`, baseline is constructed for a single run between start and end year.

Value

`ssignal` object containing the baseline signal

Examples

```
# load sample data
data("sycamore")
x = sycamore

# get streamflow object for the sample data
x.streamflow = asStreamflow(x)
```

```
# baseline for single run for all the years in input signal
bl.singlerun.all = prepareBaseline(x.streamflow)

# baseline for singlerun between the given start and end years
bl.singlerun.filtered = prepareBaseline(x.streamflow, year.start = 1961,
                                         year.end = 2000)

# baseline with windowinng and bootstrapping on all years in the input signal
bl.windowed.all = prepareBaseline(x.streamflow, window.20 = TRUE)

# baseline with windowing and bootstrapping on given start year
#  with end year inferred from singal
bl.windowed.filtered = prepareBaseline(x.streamflow, year.start = 1961,
                                        window.20 = TRUE)
```

---

residplot.extreme *Plot annual extreme residuals*

Description

Creates a plot with the maximum annual low- and high residuals for each year in the series.

Usage

```
residplot.extreme(x, text=FALSE, data=FALSE)
```

Arguments

x	Object of class `streamflow`.
text	Logical. If true, points corresponding to flows greater than 2σ are labeled on the plot.
data	Logical. If true, the extreme residuals are returned in the output.

Value

Plot with year on the x-axis and the maximum residual magnitude for that year on the y-axis.

If `data=TRUE`, output includes a `list` with the following components:

annual.max	Matrix with data corresponding to the maximum residual flow for each year in series.
annual.min	Matrix with data corresponding to the minimum residual flow for each year in series.

See Also

`sigmaHighFlows` `sigmaLowFlows`

Examples

```
# load data
data(sycamore)

# plot
residplot.extreme(asStreamflow(sycamore))
```

`sigmaeventsplot` *Plot events by day of the year*

Description

Creates a . This is similar to the plots produced in the "compare.periods" function, but only displays the data for a single time period.

Usage

```
sigmaeventsplot(x)
```

Arguments

`x`	Object of class "streamflow".

Value

A "ggplot2" plot depicting frequency of events greater than sigma, organized circularly by ordinal day of the year.

See Also

`sigmaHighFlows` `sigmaLowFlows` `compare.periods`

sigmaHighFlows *Estimate catastrophic flow variability*

Description

Calculates catastrophic variability for high flow events. Positive residuals from the seasonal signal are used to calculate $\sigma.hf$, the standard deviation of high-flow events.

Usage

```
sigmaHighFlows(x, resid.column)
```

Arguments

x	An object of class data.frame or streamflow. If a data.frame is used, one column should contain residuals.
resid.column	Optional numeric specifiying which column contains residuals. Required if x is a data frame.

Value

An object of class list with items

n.floods	Number of independent events with positive residuals.
sigma.hfa	Estimated sigma using the y-intercept.
sigma.hfb	Estimated sigma using the slope ($\sigma.hf$).
flood.line	Matrix containing fitted, observed, and residual values from regression of log counts on bin midpoints.
onesigma.events	matrix containing information for all events below $\sigma.hf$ (as calculated using the slope). Columns will contain the same data as the output from the independentEvents function.
twosigma.events	matrix containing information for all events below $2\sigma.hf$. Columns will contain the same data as the output from the independentEvents function.

See Also

independentEvents sigmaLowFlows

Examples

```
# load data
data(sycamore)

# get streamflow object
sf = asStreamflow(sycamore)

# estimate catastrophic high flow variability
sigmaHighFlows(sf)
```

`sigmaLowFlows` *Estimate catastrophic flow variability*

Description

Calculates catastrophic variability for low flow events. Negative residuals from the seasonal signal are used to calculate $\sigma.lf$, the standard deviation of low-flow events.

Usage

```
sigmaLowFlows(x, resid.column)
```

Arguments

`x`	An object of class `data.frame` or `streamflow`. If a `data.frame` is used, one column should contain residuals.
`resid.column`	Optional numeric specifiying which column contains residuals. Required if `x` is a data frame.

Value

An object of class `list` with items

`n.droughts`	Number of independent events with negative residuals.
`sigma.lfa`	Estimated sigma using the y-intercept.
`sigma.lfb`	Estimated sigma using the slope ($\sigma.lf$).
`drought.line`	Matrix containing fitted, observed, and residual values from regression of log counts on bin midpoints.
`onesigma.events`	matrix containing information for all events below $\sigma.lf$ (as calculated using the slope). Columns will contain the same data as the output from the `independentEvents` function.
`twosigma.events`	matrix containing information for all events below $2\sigma.lf$. Columns will contain the same data as the output from the `independentEvents` function.

See Also

independentEvents sigmaHighFlows

Examples

```
# load data
data(sycamore)

# get streamflow object
sf = asStreamflow(sycamore)

# estimate catastrophic low flow variability
sigmaLowFlows(sf)
```

---

sycamore — *Sycamore Creek Data*

Description

A data series from USGS station 09510200, Sycamore Creek (Fort McDowell, AZ).

Usage

```
data(sycamore)
```

Format

The data are supplied as a matrix with years in the first column and average daily discharge in the second column. The series contains 18263 observations starting Jan 1, 1961 and ending Jan 1, 2011.

Source

- USGS National Water Information Sysem: http://waterdata.usgs.gov/usa/nwis/uv?09510200

Index