Package 'ctbi'

March 9, 2022

Type Package

Title A Procedure to Clean, Decompose and Aggregate Timeseries

Version 1.0.1

Description

Clean, decompose and aggregate univariate time series following the procedure ``Cyclic/trend decomposition using bin interpolation'' and the Logbox method for flagging outliers, both detailed in Ritter, F.: Technical note: A procedure to clean, decompose and aggregate time series, Hydrol. Earth Syst. Sci. Discuss. [preprint], <doi:10.5194/hess-2021-609>, in review.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

RoxygenNote 7.1.2

URL https://github.com/fritte2/ctbi

Imports data.table (>= 1.14.2), stats (>= 4.1.0), utils (>= 4.1.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Francois Ritter [cre, aut] (<https://orcid.org/0000-0001-6123-2145>)

Maintainer Francois Ritter <ritter.francois@gmail.com>

Repository CRAN

Date/Publication 2022-03-09 20:40:06 UTC

R topics documented:

`ctbi` *ctbi*

Description

Clean, decompose and aggregate univariate time series following the procedure "Cyclic/trend decomposition using bin interpolation" and the Logbox method for flagging outliers, both detailed in Ritter, F.: Technical note: A procedure to clean, decompose and aggregate time series, Hydrol. Earth Syst. Sci. Discuss. [preprint], <https://doi.org/10.5194/hess-2021-609>, in review, 2021.

Usage

```
ctbi(
  data.input,
  bin.side = NULL,
  bin.period,
  bin.center = NULL,
  bin.FUN = "mean",
  bin.max.f.NA = 0.2,
  SCI.min = 0.6,
  k.outliers = 0.6,
  ylim = c(-Inf, +Inf)
)
```

Arguments

`data.input`	Two columns data.table (or data.frame) with the first column being the time component (POSIXct, Date or numeric) and the second column the value (numeric)
`bin.side`	one side of a bin (same class as the time component)
`bin.period`	time interval between two sides of a bin. If the time component x.t of data0 is numeric, bin.period is numeric. If x.t is POSIXct or Date, bin.period = 'k units', with k an integer and units = (seconds, minutes, hours, days, weeks, half-months, months, years, decades, centuries, millenaries)

bin.center	if bin.side is not specified, one center of a bin (same class as the time component)
bin.FUN	character ('mean', 'median' or 'sum') that defines the aggregating operator
bin.max.f.NA	numeric between 0 and 1 that specifies the maximum fraction of missing values for a bin to be accepted. The minimum number of non-NA points for a bin to be accepted is bin.size*(1-bin.max.f.NA) with bin.size the number of points per bin
SCI.min	numeric between 0 and 1 that is compared to the Stacked Cycles Index (SCI). If SCI > SCI.min, missing values are imputed in accepted bins with the sum of the long-term and cyclic components. SCI.min = Inf means that no values are imputed
k.outliers	positive numeric that defines the outlier level in the Logbox method used to flag outliers, with k.outliers = 0.16 corresponding to a Gaussian distribution and k.outliers = 0.8 to an Exponential distribution. The default value of k.outliers = 0.6 has been calculated based on a set of distributions with moderate skewness and kurtosis (the Pearson family). k.outliers = Inf means that no outliers are flagged
ylim	numeric vector of length 2 that defines the range of possible values. Values below ylim[1] or above ylim[2] are set to NA

Value

A list that contains:

data0, the raw dataset (same class as data.input), with 8 columns: (i) time; (ii) outlier-free and imputed data; (iii) index.bin: index of the bins associated with each data points (the index is negative if the bin is rejected); (iv) long.term: long-term trend; (v) cycle: cyclic component; (vi) outliers: quarantined outliers; (vii) imputed: value of the imputed data points; (viii) time.bin: relative position of the data points in their bins, between 0 and 1

data1, the aggregated dataset (same class as data.input), with 10 columns: (i) aggregated time (center of the bins); (ii) aggregated data; (iii) index.bin: index of the bin (negative value if the bin is rejected); (iv) bin.start: start of the bin; (v) bin.end: end of the bin; (vi) n.points: number of points per bin (including NA values); (vii) n.NA: number of NA values per bin, originally; (viii) n.outliers: number of outliers per bin; (ix) n.imputed: number of imputed points per bin; (x) variability associated with the aggregation (standard deviation for the mean, MAD for the median and nothing for the sum)

SCI (Stacked Cycle Index), a numeric between 0 and 1 related to the strength of the cyclic pattern within each bin. SCI is defined as SCI = 1 - SS.res/SS.tot - 1/N.bin with SS.tot the sum of the squared detrended data, SS.res the sum of the squared detrended & deseasonalized data, and N.bin the number of accepted bins

mean.cycle, a dataset (same class as data.input) with bin.size rows and 4 columns: (i) generic.time.bin1: time of the first bin; (ii) mean: the mean stack of detrended data; (iii) sd: the standard deviation on the mean; (iv) time.bin: relative position of the data points in the bin, between 0 and 1

bin.size, the median number of points in non-empty bins

n.bin.min, the minimum number of points for a bin to be accepted

Examples

```
# example of contaminated sunspot data
example1 <- data.frame(year = 1700:1988,sunspot = as.numeric(sunspot.year))
example1[sample(1:289,30),'sunspot'] <- NA
example1[c(5,30,50),'sunspot'] <- c(-50,300,400)
example1 <- example1[-(70:100),]
bin.period <- 11 # aggregation performed every 11 years (the year is numeric here)
bin.side <- 1989 # to capture the last year, 1988, in a complete bin
bin.FUN <- 'mean'
bin.max.f.NA <- 0.2 # maximum of 20% of missing data per bin
ylim <- c(0,Inf) # negative values are impossible

list.main <- ctbi(example1,bin.period=bin.period,
                      bin.side=bin.side,bin.FUN=bin.FUN,
                      ylim=ylim,bin.max.f.NA=bin.max.f.NA)
data0.example1 <- list.main$data0 # cleaned raw dataset
data1.example1 <- list.main$data1 # aggregated dataset.
SCI.example1 <- list.main$SCI # this data set shows a moderate seasonality
mean.cycle.example1 <- list.main$mean.cycle # this data set shows a moderate seasonality
bin.size.example1 <- list.main$bin.size # 12 data points per bin on average (12 months per year)

plot(mean.cycle.example1[,'generic.time.bin1'],
     mean.cycle.example1[,'mean'],type='l',ylim=c(-80,80),
     ylab='sunspot cycle',
     xlab='11 years window')
lines(mean.cycle.example1[,'generic.time.bin1'],
      mean.cycle.example1[,'mean']+mean.cycle.example1[,'sd'],type='l',lty=2)
lines(mean.cycle.example1[,'generic.time.bin1'],
      mean.cycle.example1[,'mean']-mean.cycle.example1[,'sd'],type='l',lty=2)
title(paste0('mean cycle (weak cyclicity: SCI = ',SCI.example1,')'))
# the SCI is much higher on the raw dataset without contamination (SCI = 0.34)
ctbi.plot(list.main,show.n.bin=10)

# example of beaver data
temp.beaver <- beaver1[,'temp']
t.char <- as.character(beaver1[,'time'])
minutes <- substr(t.char,nchar(t.char)-1,nchar(t.char))
hours <- substr(t.char,nchar(t.char)-3,nchar(t.char)-2)
hours[hours==""] <- '0'
days <- c(rep(12,91),rep(13,23))
time.beaver <- as.POSIXct(paste0('2000-12-',days,' ',hours,':',minutes,':00'),tz='UTC')
example2 <- data.frame(time=time.beaver,temp=temp.beaver)

bin.period <- '1 hour' # aggregation performed every hour
bin.side <- as.POSIXct('2000-12-12 00:00:00',tz='UTC') # start of a bin
bin.FUN <- 'mean' # aggregation operator
bin.max.f.NA <- 0.2 # maximum of 20% of missing data per bin
ylim <- c(-Inf,Inf)
list.main <- ctbi(example2,bin.period=bin.period,
                  bin.side=bin.side,bin.FUN=bin.FUN,
                  ylim=ylim,bin.max.f.NA=bin.max.f.NA)
data0.example2 <- list.main$data0 # cleaned raw dataset
```

```
data1.example2 <- list.main$data1 # aggregated dataset. 1 outlier flagged.
SCI.example2 <- list.main$SCI # this data set shows no seasonality every hour
ctbi.plot(list.main,show.n.bin = 50)
```

---

ctbi.cycle *ctbi.cycle*

Description

Calculate the mean (or median) stack of the detrended data for all bins, and add the cyclic component column to data0.

Usage

```
ctbi.cycle(data0, bin.size, outliers.checked)
```

Arguments

data0	data.table with the columns x (time series), y (values), time.bin (position of x between 0 and 1 with respect to the bin boundaries), cycle.index (index between 1 and bin.size attached to time.bin), and long.term (the long-term trend)
bin.size	median number of points within all non-empty bins
outliers.checked	boolean to indicate if the median (outliers.checked = FALSE) or the mean (outliers.checked = TRUE) should be used to calculate the stack

Value

A list that contains data0 (data0.l) and a data.table that contains the mean (or median) stack of all accepted bins (FUN.cycle.l)

Examples

```
library(data.table)
x <- seq(from=as.Date('2001-01-01'),to=as.Date('2010-12-01'),by='1 month')
y <- 3*cos(2*pi*(0:(length(x)-1))/12)+runif(length(x))
bin.size <- 12
outliers.checked <- TRUE
time.bin <- rep(((1:bin.size)/bin.size)-(1/(2*bin.size)),10)
cycle.index <- findInterval(time.bin,(1:(bin.size-1))/bin.size)+1
long.term <- rep(0,length(x))
data0 <- data.table(x=x,y=y,cycle.index=cycle.index,long.term=long.term,time.bin=time.bin)
list.cycle <- ctbi.cycle(data0,bin.size,outliers.checked)
data0.with.cyclic.component <- list.cycle$data0.l
mean.cycle <- list.cycle$FUN.cycle.l
```

ctbi.long.term *ctbi.long.term*

Description

Calculate the long-term trend with a linear interpolation between the mean (or median) of bins defined between two consecutive centers. Bins defined between two consecutive sides are calculated as well to complete for missing values if they have neighbors. Bins without sufficient data are discarded.

Usage

```
ctbi.long.term(data0, n.bin.min, seq.bin.side, outliers.checked)
```

Arguments

data0	data.table with the columns x (time series), y (values), side.index (index associated with each bin defined between two consecutive centers) and index.bin (index associated with each bin defined between two consecutive sides)
n.bin.min	minimum number of points for a bin to be accepted
seq.bin.side	sequence of the sides of the bins
outliers.checked	boolean to indicate if the median (outliers.checked = FALSE) or the mean (outliers.checked = TRUE) should be used to calculate the long-term trend

Value

data0 with the long-term trend added (column long.term)

Examples

```
library(data.table)
x <- seq(from=as.Date('2001-01-01'),to=as.Date('2010-12-01'),by='1 month')
y <- 3*cos(2*pi*(0:(length(x)-1))/12)+runif(length(x))
outliers.checked <- TRUE
seq.bin.side <- seq(from=as.Date('2001-01-01'),to=as.Date('2011-01-01'),by='1 year')
seq.bin.center <- seq(from=as.Date('2001-06-01'),to=as.Date('2010-06-01'),by='1 year')
index.bin <- findInterval(x,seq.bin.side)
side.index <- findInterval(x,seq.bin.center)+0.5
n.bin.min <- 10 # minimum of 10 months of data for a bin to be accepted
data0 <- data.table(x=x,y=y,index.bin=index.bin,side.index=side.index)
data0.with.long.term <- ctbi.long.term(data0,n.bin.min,seq.bin.side,outliers.checked)
```

ctbi.outliers *ctbi.outliers*

Description

Flag the outliers with the Logbox method, which replaces the original constant 1.5 of the Boxplot rule with k.outliers*log(n)+1, with n being the sample size of the residuals (detrended & deseasonalized).

Usage

```
ctbi.outliers(data0, k.outliers)
```

Arguments

data0	data.table with the columns x (time series), y (values), long.term (the long-term trend) and cycle (the cyclic component)
k.outliers	positive numeric that defines the outlier level in the Logbox method used to flag outliers, with k.outliers = 0.16 corresponding to a Gaussian distribution and k.outliers = 0.8 to an Exponential distribution. The default value of k.outliers = 0.6 has been calculated based on a set of distributions with moderate skewness and kurtosis (the Pearson family). k.outliers = Inf means that no outliers are flagged

Value

data0 with the outliers flagged (column outliers), and the corresponding y values set to NA

Examples

```
library(data.table)
x <- seq(from=as.Date('2001-01-01'),to=as.Date('2010-12-01'),by='1 month')
y <- 3*cos(2*pi*(0:(length(x)-1))/12)+runif(length(x))
y[c(5,15,20)] <- c(-30,20,40) # add 3 outliers
long.term <- rep(0,length(x))
cycle <- 3*cos(2*pi*(0:(length(x)-1))/12)
k.outliers <- 0.6
data0 <- data.table(x=x,y=y,cycle=cycle,long.term=long.term)
data0.with.outliers.flagged <- ctbi.outliers(data0,k.outliers)
```

ctbi.plot *ctbi.plot*

Description

Plot the raw data with the bins, long-term trend and cyclic component shown.

Usage

```
ctbi.plot(list.main, show.outliers = TRUE, show.n.bin = 10)
```

Arguments

list.main	the list output from the function ctbi.main
show.outliers	boolean to show or hide flagged outliers
show.n.bin	number of bins shown within one graphic

Value

No return value

ctbi.timeseries *ctbi.timeseries*

Description

Calculate the sequence of bin sides that encompasses the original time series based on a bin period and a bin side (or a bin center). The sequence of bin centers is calculated as well.

Usage

```
ctbi.timeseries(x.t, bin.period, bin.side = NULL, bin.center = NULL)
```

Arguments

x.t	original timeseries (date, POSIXct or numeric)
bin.period	time interval between two sides of a bin. If x.t is numeric, bin.period is numeric. If x.t is POSIXct or Date, bin.period = 'k units', with k an integer and units = (seconds, minutes, hours, days, weeks, half-months, months, years, decades, centuries, millenaries)
bin.side	one side of a bin (same class as x.t)
bin.center	if bin.side is not specified, one center of a bin (same class as x.t)

Value

A list that contains:

seq.bin.side, the sequence of bin sides (same class as bin.side)

seq.bin.center, the sequence of bin centers (same class as bin.side)

time.step.median, the median time step (numeric)

Examples

```
x.t <- seq(from=as.Date('2001-01-01'),to=as.Date('2010-12-01'),by='1 month')
bin.side <- as.Date('2003-10-01')
bin.period <- '4 months'
list.ts <- ctbi.timeseries(x.t,bin.period,bin.side)
seq.bin.side.Date <- list.ts$seq.bin.side
seq.bin.center.Date <- list.ts$seq.bin.center

x.t <- seq(from=as.POSIXct('2001-01-01 12:45:23'),to=as.POSIXct('2001-01-01 13:34:21'),by='18 s')
bin.side <- as.POSIXct('2001-01-01 13:00:00')
bin.period <- '1 minute' # '60 s', '60 sec', '60 seconds', '1 min' are also possible
list.ts <- ctbi.timeseries(x.t,bin.period,bin.side)
seq.bin.side.POSIXct <- list.ts$seq.bin.side
seq.bin.center.POSIXct <- list.ts$seq.bin.center

x.t <- seq(from= - 50000,to= 2000 ,by=1000)
bin.side <- 0
bin.period <- 10000
list.ts <- ctbi.timeseries(x.t,bin.period,bin.side)
seq.bin.side.numeric <- list.ts$seq.bin.side
seq.bin.center.numeric <- list.ts$seq.bin.center
```

---

hidd.check.bin.period *hidd.check.bin.period*

Description

interpret the string character bin.period used in ctbi.timeseries or ctbi.main

Usage

```
hidd.check.bin.period(bin.period)
```

Arguments

bin.period	a character string or a numeric

Value

A list that contains:

number, a numeric that indicates the value of bin.period

units, a character that indicates the unit of bin.period

bin.period.value.seconds, a numeric that indicates the value in seconds of bin.period

bin.period.value.days, a numeric that indicates the value in days of bin.period

---

hidd.count.NA *hidd.count.NA*

Description

Calculate the number of NA values in a vector

Usage

```
hidd.count.NA(x)
```

Arguments

x	a numeric vector

Value

the number of NA values in a vector

---

hidd.count.noNA *hidd.count.noNA*

Description

Calculate the number of non-NA values in a vector

Usage

```
hidd.count.noNA(x)
```

Arguments

x	a numeric vector

Value

the number of non-NA values in a vector

`hidd.mad` *hidd.mad*

Description

Calculate the mean absolute deviation (MAD) of a vector if the number of its non-NA values is above a threshold. Otherwise, return NA.

Usage

```
hidd.mad(x, N.min.NA)
```

Arguments

`x`	a numeric vector
`N.min.NA`	a numeric threshold

Value

a numeric (either NA or the MAD of x)

`hidd.mean` *hidd.mean*

Description

Calculate the mean of a vector if the number of its non-NA values is above a threshold. Otherwise, return NA.

Usage

```
hidd.mean(x, N.min.NA)
```

Arguments

`x`	a numeric vector
`N.min.NA`	a numeric threshold

Value

y a numeric (either NA or the mean of x)

`hidd.median` *hidd.median*

Description

Calculate the median of a vector if the number of its non-NA values is above a threshold. Otherwise, return NA.

Usage

```
hidd.median(x, N.min.NA)
```

Arguments

`x`	a numeric vector
`N.min.NA`	a numeric threshold

Value

a numeric (either NA or the median of x)

`hidd.rel.time` *hidd.rel.time*

Description

calculate the relative position of each timestep of a vector with respect to the bin boundaries.

Usage

```
hidd.rel.time(x, seq.bin.side)
```

Arguments

`x`	a vector (numeric, POSIXct or Date)
`seq.bin.side`	the sequence of bin sides

Value

the relative position of each value of x with respect to the bins in seq.bin.side (between 0 and 1)

`hidd.replace` *hidd.replace*

Description

Replace all values within a vector with NA values if the sum of its non-NA values is below a threshold.

Usage

```
hidd.replace(x, N.min.NA)
```

Arguments

`x`	a numeric vector
`N.min.NA`	a numeric threshold

Value

the vector x

`hidd.sd` *hidd.sd*

Description

Calculate the standard deviation of a vector if the number of its non-NA values is above a threshold. Otherwise, return NA.

Usage

```
hidd.sd(x, N.min.NA)
```

Arguments

`x`	a numeric vector
`N.min.NA`	a numeric threshold

Value

a numeric (either NA or the standard deviation of x)

hidd.seq *hidd.seq*

Description

similar to seq, except that when 'from' starts the 29, 30 or 31 of a month, seq2 adds 5 days to 'from', run seq, and then subtracts 5 days to the output. This is done because the function seq is not consistent when time series start at the end of the months.

Usage

```
hidd.seq(
  from = 1,
  to = 1,
  by = ((to - from)/(length.out - 1)),
  length.out = NULL
)
```

Arguments

from, to	the starting and (maximal) end values of the sequence. Of length 1 unless just from is supplied as an unnamed argument.
by	number: increment of the sequence.
length.out	desired length of the sequence. A non-negative number, which for seq and seq.int will be rounded up if fractional.

Value

a vector of same class than from

hidd.sum *hidd.sum*

Description

Calculate the sum of a vector if the number of its non-NA values is above a threshold. Otherwise, return NA.

Usage

```
hidd.sum(x, N.min.NA)
```

Arguments

x	a numeric vector
N.min.NA	a numeric threshold

Value

a numeric (either NA or the sum of x)

Index