Package ‘ctbi’

January 20, 2023

Type Package

Title A Procedure to Clean, Decompose and Aggregate Timeseries

Version 2.0.5

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., 27, 349–361, <doi:10.5194/hess-27-349-2023>, 2023.

License GPL-3

Encoding UTF-8

Depends R (>= 4.2.0)

RoxygenNote 7.2.2

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 2023-01-20 11:50:02 UTC

R topics documented:

ctbi 2
ctbi.cycle 5
ctbi.long.term 6
ctbi.outlier 7
ctbi.plot 8
ctbi.timeseries 9

hidd.check.bin.period 10
hidd.count.NA 10
hidd.count.noNA 11
hidd.mad 11
hidd.mean 12
hidd.median 12
hidd.rel.time 13
hidd.replace 13
hidd.sd 14
hidd.seq 14
hidd.sum 15

Index **16**

`ctbi` *ctbi*

Description

Please cite the following companion paper if you're using the `ctbi` package: Ritter, F.: Technical note: A procedure to clean, decompose, and aggregate time series, Hydrol. Earth Syst. Sci., 27, 349–361, https://doi.org/10.5194/hess-27-349-2023, 2023.

The goal of `ctbi` is to **clean**, **decompose**, **impute** and **aggregate** univariate time series. `ctbi` stands for *Cyclic/Trend decomposition using Bin Interpolation*: the time series is divided into a sequence of non-overlapping bins (inputs: `bin.side` or `bin.center`, and `bin.period`). Bins with enough data (input: `bin.max.f.NA`) are *accepted*, and otherwise *rejected* (their values are set to `NA`). The **long-term trend** is a linear interpolation of the mean values between successive bins. The **cyclic component** is the mean stack of detrended data within all accepted bins.

Outliers present in the residuals are flagged using an enhanced box plot rule (called **Logbox**, input: `coeff.outlier`) that is adapted to non-Gaussian data and keeps the type I error at $\frac{0.1}{\sqrt{n}}$ % (percentage of erroneously flagged outliers). **Logbox** replaces the original $\alpha = 1.5$ constant of the box plot rule with $\alpha = A \times \log(n) + B + \frac{C}{n}$. The variable $n \geq 9$ is the sample size, $C = 36$ corrects biases emerging in small samples, and A and B are automatically calculated on a predictor of the maximum tail weight (m_*).

The strength of the **cyclic pattern** within each bin is quantified by a new metric, the **Stacked Cycles Index** defined as $SCI = 1 - \frac{SS_{res}}{SS_{tot}} - \frac{1}{N_{bin}}$. The variable SS_{tot} is the sum of the squared detrended data, SS_{res} is the sum of the squared detrended & deseasonalized data, and N_{bin} is the number of accepted bins. A value of $SCI \leq 0$ is associated with no cyclicity, while $SCI = 1$ is associated with a perfectly cyclic signal. Data can be imputed if $SCI_{min} \leq SCI$ (input: `SCI.min`). Finally, data are aggregated in each bin (input: `bin.FUN`).

Important functions of the package: `ctbi`, `ctbi.outlier` (flag outliers in univariate datasets with the **Logbox** method) and `ctbi.plot` (plot the time series).

Usage

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

Arguments

`data.input`	two columns data.frame (or data.table) with the first column being the time component (POSIXct, Date or numeric) and the second column the value (numeric)
`bin.side`	one side of any bin (same class as the time component)
`bin.period`	time interval between two sides of a bin. If the time component x.t of data.input 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.min.accepted = 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. If `SCI.min` = `NA`, no values are imputed
`coeff.outlier`	one of `coeff.outlier` = 'auto' (default value), `coeff.outlier` = 'gaussian', `coeff.outlier` = c(A,B,C) or `coeff.outlier` = `NA`. If `coeff.outlier` = 'auto', C = 36 and the coefficients A and B are calculated on m_*. If `coeff.outlier` = 'gaussian', `coeff.outlier` = c(0.08,2,36), adapted to the Gaussian distribution. If `coeff.outlier` = `NA`, no outliers are flagged
`ylim`	numeric vector of length 2 that defines the range of possible values. Values strictly below ylim[1] or strictly above ylim[2] are set to NA. Values equal to ylim[1] or ylim[2] are discarded from the residuals

Value

A list that contains:

data0, the raw dataset (same class as `data.input`), with 9 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) residuals: residuals including the outliers; (vii) outliers: quarantined outliers; (viii) imputed: value of the imputed data points; (ix) 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)

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

summary.bin, a vector that contains bin.size (median number of points in non-empty bins), bin.size.min.accepted (minimum number of points for a bin to be accepted) and SCI

summary.outlier, a vector that contains A, B, C, m_*, the size of the residuals (n), and the lower and upper outlier threshold

Examples

```
# example of the contaminated sunspot data
example1 <- data.frame(year = 1700:1988,sunspot = as.numeric(sunspot.year))
example1[sample(1:289,30),'sunspot'] <- NA # contaminate data with missing values
example1[c(5,30,50),'sunspot'] <- c(-50,300,400) # contaminate data with outliers
example1 <- example1[-(70:100),] # create gap in the data
bin.period <- 11 # aggregation performed every 11 years (the year is numeric here)
bin.side <- 1989 # give one side of a 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.
mean.cycle.example1 <- list.main$mean.cycle # this data set shows a moderate seasonality
summary.bin.example1 <- list.main$summary.bin # confirmed with SCI = 0.50
summary.outlier.example1 <- list.main$summary.outlier

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 (moderate cyclicity: SCI = ',summary.bin.example1['SCI'],')'))
# plot tool:
```

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

# example of the 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') # give one side 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.
lower.threshold <- list.main$summary.outlier['lower.outlier.threshold']
upper.threshold <- list.main$summary.outlier['upper.outlier.threshold']
hist(data0.example2[,'residuals'],xlim=c(-0.5,0.5),30,main='beaver residuals')
abline(v=c(lower.threshold,upper.threshold),col='red',lwd=2) # show the histogram of the residuals
```

`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.outlier` *ctbi.outlier*

Description

Please cite the following companion paper if you're using the `ctbi` package: Ritter, F.: Technical note: A procedure to clean, decompose, and aggregate time series, Hydrol. Earth Syst. Sci., 27, 349–361, https://doi.org/10.5194/hess-27-349-2023, 2023.

Outliers in an univariate dataset `y` are flagged using an enhanced box plot rule (called **Logbox**, input: `coeff.outlier`) that is adapted to non-Gaussian data and keeps the type I error at $\frac{0.1}{\sqrt{n}}$ % (percentage of erroneously flagged outliers).

The box plot rule flags data points as outliers if they are below L or above U using the sample quantile q:

$L = q(0.25) - \alpha \times (q(0.75) - q(0.25))$

$U = q(0.75) + \alpha \times (q(0.75) - q(0.25))$

Logbox replaces the original $\alpha = 1.5$ constant of the box plot rule with $\alpha = A \times \log(n) + B + \frac{C}{n}$. The variable $n \geq 9$ is the sample size, $C = 36$ corrects biases emerging in small samples, and A and B are automatically calculated on a predictor of the maximum tail weight defined as $m_* = \max(m_-, m_+) - 0.6165$.

The two functions (m_-,m_+) are defined as:

$m_- = \frac{q(0.875) - q(0.625)}{q(0.75) - q(0.25)}$

$m_+ = \frac{q(0.375) - q(0.125)}{q(0.75) - q(0.25)}$

And finally, $A = f_A(m_*)$ and $B = f_B(m_*)$ with m_* restricted to [0,2]. The functions (f_A, f_B) are defined as:

$f_A(x) = 0.2294 \exp(2.9416x - 0.0512x^2 - 0.0684x^3)$

$f_B(x) = 1.0585 + 15.6960x - 17.3618x^2 + 28.3511x^3 - 11.4726x^4$

Both functions have been calibrated on the Generalized Extreme Value and Pearson families.

Usage

```
ctbi.outlier(y, coeff.outlier = "auto")
```

Arguments

`y`	univariate data (numeric vector)
`coeff.outlier`	one of `coeff.outlier` = 'auto' (default value), `coeff.outlier` = 'gaussian', `coeff.outlier` = c(A,B,C) or `coeff.outlier` = `NA`. If `coeff.outlier` = 'auto', C = 36 and the coefficients A and B are calculated on m_*. If `coeff.outlier` = 'gaussian', `coeff.outlier` = c(0.08,2,36), adapted to the Gaussian distribution. If `coeff.outlier` = `NA`, no outliers are flagged

Value

A list that contains:

xy, a two columns data frame that contains the clean data (first column) and the outliers (second column)

summary.outlier, a vector that contains A, B, C, m_*, the size of the residuals (n), and the lower and upper outlier threshold

Examples

```
x <- runif(30)
x[c(5,10,20)] <- c(-10,15,30)
example1 <- ctbi.outlier(x)
```

`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
`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, bin.center = NULL)
```

Arguments

`x.t`	original time series (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

ctbi, 2
ctbi.cycle, 5
ctbi.long.term, 6
ctbi.outlier, 7
ctbi.plot, 8
ctbi.timeseries, 9

hidd.check.bin.period, 10
hidd.count.NA, 10
hidd.count.noNA, 11
hidd.mad, 11
hidd.mean, 12
hidd.median, 12
hidd.rel.time, 13
hidd.replace, 13
hidd.sd, 14
hidd.seq, 14
hidd.sum, 15