

Package ‘RtsEva’

July 21, 2025

Type Package

Title Performs the Transformed-Stationary Extreme Values Analysis

Version 1.1.0

Maintainer Alois Tilloy <alois.tilloy@ec.europa.eu>

Description Adaptation of the 'Matlab' 'tsEVA' toolbox developed by Lorenzo Mentaschi available here:

<<https://github.com/menta78/tsEva>>. It contains an implementation of the Transformed-Stationary (TS) methodology for non-stationary extreme value Analysis (EVA) as described in Mentaschi et al. (2016) <[doi:10.5194/hess-20-3527-2016](https://doi.org/10.5194/hess-20-3527-2016)>.

In synthesis this approach consists in:

- (i) transforming a non-stationary time series into a stationary one to which the stationary extreme value theory can be applied; and
- (ii) reverse-transforming the result into a non-stationary extreme value distribution.

'RtsEva' offers several options for trend estimation (mean, extremes, seasonal) and contains multiple plotting functions displaying different aspects of the non-stationarity of extremes.

License GPL (>= 3)

URL <https://github.com/r-lib/devtools>,
<https://github.com/Alois/RtsEva>

Depends R (>= 2.10)

Imports dplyr, evd, ggplot2, lubridate, methods, moments, POT, pracma, scales, texmex, tsibble, xts, grDevices, rlang, changepoint

Suggests knitr, ncd4, rmarkdown, rnaturalearth, terra, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Config/testthat/edition 3

BugReports <https://github.com/Alois/RtsEva/issues>

NeedsCompilation no

Author Alois Tilloy [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-5881-0642>>)

Repository CRAN

Date/Publication 2025-06-10 10:10:02 UTC

Contents

ArdecheStMartin	3
check_timeseries	4
computeAnnualMaxima	4
computeMonthlyMaxima	5
DanubeVienna	6
declustpeaks	6
EbroZaragoza	7
empdis	8
empdisl	9
findMax	10
initPercentiles	10
max_daily_value	11
RhoneLyon	12
tsEasyParseNamedArgs	12
tsEstimateAverageSeasonality	13
tsEvaChangepts	14
tsEvaComputeReturnLevelsGEV	15
tsEvaComputeReturnLevelsGEVFromAnalysisObj	16
tsEvaComputeReturnLevelsGPD	18
tsEvaComputeReturnLevelsGPDFromAnalysisObj	19
tsEvaComputeReturnPeriodsGEV	20
tsEvaComputeReturnPeriodsGPD	21
tsEvaComputeRLsGEVGPD	23
tsEvaComputeTimeRP	24
tsEvaDetrendTimeSeries	25
tsEvaFillSeries	26
tsEvaFindTrendThreshold	26
tsEvaNanRunnigBlowTh	27
tsEvaNanRunningMean	28
tsEvaNanRunningPercentiles	29
tsEvaNanRunningStatistics	30
tsEvaNanRunningVariance	31
TsEvaNs	31
tsEvaPlotAllIRLevelsGEV	33
tsEvaPlotAllIRLevelsGPD	35
tsEvaPlotGEVImageSc	37
tsEvaPlotGEVImageScFromAnalysisObj	38
tsEvaPlotGPDImageSc	39
tsEvaPlotGPDImageScFromAnalysisObj	40

tsEvaPlotReturnLevelsGEV	42
tsEvaPlotReturnLevelsGEVFromAnalysisObj	43
tsEvaPlotReturnLevelsGPD	45
tsEvaPlotReturnLevelsGPDFromAnalysisObj	46
tsEvaPlotSeriesTrendStdDevFromAnalysisObj	48
tsEvaPlotTransfToStat	49
tsEvaPlotTransfToStatFromAnalysisObj	50
tsEvaRunningMeanTrend	51
tsEvaSampleData	52
tsEvaTransformSeriesToStationaryMMXTrend	53
tsEvaTransformSeriesToStationaryMultiplicativeSeasonality	54
tsEvaTransformSeriesToStationaryPeakTrend	56
tsEvaTransformSeriesToStationaryTrendAndChangepts	57
tsEvaTransformSeriesToStationaryTrendAndChangepts_ciPercentile	59
tsEvaTransformSeriesToStationaryTrendOnly	60
tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile	61
tsEvaTransformSeriesToStatSeasonal_ciPercentile	63
tsEVstatistics	64
tsGetNumberPerYear	66
tsGetPOT	67

Index**69**

ArdecheStMartin	<i>Simulated river discharge of the Ardeche river at Saint Martin d'Ardeche</i>
-----------------	---

Description

A time series of simulated river discharge of the Ardeche river close to its confluence with the Rhone (longitude = 4.658 \ latitude = 44.258) from 1951 to 2020. Time series extracted from the HERA dataset: <https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>. The Ardèche is Mediterranean river mostly known for tourism due to its scenic gorges, but floods and droughts can impact the local economy and environment.

Usage

```
data(ArdecheStMartin)
```

Format

Two column dataframe: #

- time (POSIXct timestamp with 6-hourly resolution)
- Q (6-hourly mean discharge in cubic meter per second)

Source

<https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>

check_timeseries	<i>Check if all years in a time series are present</i>
------------------	--

Description

This function checks if all years specified in a given time series are present.

Usage

```
check_timeseries(timeseries, yro)
```

Arguments

timeseries	A time series object.
yro	A vector specifying the start and end years.

Value

A logical value indicating whether all years in the time series are present.

Examples

```
ts_data <- seq(as.POSIXct("2000-01-01"), as.POSIXct("2004-12-31"), by = "year")
check_timeseries(ts_data, c(2000, 2004))
# Output: TRUE

check_timeseries(ts_data, c(2000, 2005))
# Output: FALSE
```

computeAnnualMaxima	<i>computeAnnualMaxima</i>
---------------------	----------------------------

Description

computeAnnualMaxima is a function that computes the annual maxima of a time series.

Usage

```
computeAnnualMaxima(timeAndSeries)
```

Arguments

timeAndSeries	A matrix or data frame containing the time stamps and series values. The first column should contain the time stamps and the second column should contain the series values.
---------------	--

Value

A list containing the annual maximum values, their corresponding dates, and their indices.

`annualMax` A numeric vector of annual maximum values.

`annualMaxDate` A vector of dates corresponding to the annual maximum values.

`annualMaxIndx` A vector of indices indicating the positions of the annual maximum values in the original time series.

Examples

```
timeAndSeries <- ArdecheStMartin
computeAnnualMaxima(timeAndSeries)
```

`computeMonthlyMaxima` *computeMonthlyMaxima*

Description

`computeMonthlyMaxima` is a function that computes the monthly maxima of a time series.

Usage

```
computeMonthlyMaxima(timeAndSeries)
```

Arguments

`timeAndSeries` A data frame containing the time stamps and series values. The first column should contain the time stamps, and the second column should contain the series values.

Value

A list containing the monthly maxima, corresponding dates, and indices.

`monthlyMax` A vector of the monthly maximum values.

`monthlyMaxDate` A vector of the dates corresponding to the monthly maximum values.

`monthlyMaxIndx` A vector of the indices of the monthly maximum values in the original series.

Examples

```
timeAndSeries <- ArdecheStMartin
computeMonthlyMaxima(timeAndSeries)
```

 DanubeVienna

Simulated river discharge of the Danube river at Vienna

Description

A time series of simulated river discharge of the Danube river in Vienna (longitude = 16.64 \ latitude = 48.13) from 1951 to 2020. Time series extracted from the HERA dataset: <https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>. The Danube is the longest river in the EU and is an important waterway for international trade, connecting several countries in Central and Eastern Europe.

Usage

```
data(DanubeVienna)
```

Format

Two column dataframe: #'

- time (POSIXct timestamp with 6-hourly resolution)
- Q (6-hourly mean discharge in cubic meter per second)

Source

<https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>

declustpeaks

declustpeaks

Description

declustpeaks is a function that takes in a data vector, minimum peak distance, minimum run distance, and a threshold value. It finds peaks in the data vector based on the minimum peak distance and threshold value. It then declusters the peaks based on the minimum run distance and threshold value. The function returns a data frame with information about the peaks, including the peak value, start and end indices, duration, and cluster information.

Usage

```
declustpeaks(data, minpeakdistance = 10, minrundistance = 7, qt)
```

Arguments

data	A numeric vector representing the data.
minpeakdistance	An integer specifying the minimum distance between peaks.
minrundistance	An integer specifying the minimum distance between runs.
qt	A numeric value representing the threshold for peak detection.

Value

A data frame with information about the peaks, including:

Q the peak value
max max time index
start start time index
end end time index
dur duration
cluster cluster information

Examples

```
data <- c(1, 2, 3, 4, 5, 4, 3, 2, 1)
declustpeaks(data, minpeakdistance = 2, minrundistance = 2, qt = 3)
```

EbroZaragoza

Simulated river discharge of the Ebro river at Zaragoza

Description

A time series of simulated river discharge of the Ebro river at Zaragoza (longitude = -0.825 \ latitude = 41.608) from 1951 to 2020. Time series extracted from the HERA dataset: <https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>. The Ebro is Spain's longest river, with low and high water levels alternating throughout the year, influenced by winter snowmelt and summer evaporation/human usage. The river is vital for agriculture.

Usage

```
data(EbroZaragoza)
```

Format

Two column dataframe: #'

- time (POSIXct timestamp with 6-hourly resolution)
- Q (6-hourly mean discharge in cubic meter per second)

Source

<https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>

`empdis`*empdis: Empirical Distribution Function*

Description

`empdis` is a function that calculates the empirical distribution function for a given dataset.

Usage

```
empdis(x, nyr)
```

Arguments

<code>x</code>	A numeric vector representing the dataset.
<code>nyr</code>	An integer representing the number of years in the dataset.

Value

A data frame containing:

`emp.RP` empirical return period

`haz.RP` Hazen return period

`cun.RP` Cunnane return period

`gumbel` Gumbel values

`emp.f` empirical cumulative density

`emp.hazen` Hazen cumulative density

`emp.cunnan` Cunnane cumulative density

`Q` original data

`timestamp` time component

Examples

```
x <- c(1, 2, 3, 4, 5)
nyr <- 5
empdis(x, nyr)
```

empdisl	<i>Empirical Distribution Function</i>
---------	--

Description

This function calculates the empirical distribution function for a given dataset, with a focus on low values

Usage

```
empdisl(x, nyr)
```

Arguments

x	A numeric vector or matrix representing the discharge values.
nyr	An integer representing the number of years.

Value

A data frame containing the following columns:

emp.RP	Empirical return period
haz.RP	Hazen return period
gumbel	Gumbel frequency
emp.f	Empirical frequency
emp.hazen	Empirical Hazen frequency
Q	Discharge values

Examples

```
x <- c(10, 20, 30, 40, 50)
nyr <- 5
empdisl(x, nyr)
```

findMax *findMax*

Description

findMax is a function that takes a subset of a vector and returns the index of the maximum value in that subset.

Usage

```
findMax(subIndxs, srs)
```

Arguments

subIndxs	A numeric vector representing the subset of indices to consider.
srs	A vector of numerical data

Value

The index of the maximum value in the subset.

Examples

```
srs <- c(10, 20, 30, 40, 50)
findMax(c(1, 3, 5),srs)
#result is 5.
```

initPercentiles *Initialize Percentiles*

Description

This function calculates percentiles for a given dataset

Usage

```
initPercentiles(subsrs, percentM, percent, percentP)
```

Arguments

subsrs	The input dataset.
percentM	The percentile for the lower bound.
percent	The percentile for the middle bound.
percentP	The percentile for the upper bound.

Value

A list containing the calculated percentiles and probabilities.

See Also

[tsEvaNanRunningPercentiles\(\)](#)

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
initPercentiles(series, 89, 90, 91)
```

max_daily_value	<i>Max Daily Value Function</i>
-----------------	---------------------------------

Description

This function converts a 6-hourly time series to a daily time series and calculates the maximum value for each day.

Usage

```
max_daily_value(timeseries)
```

Arguments

timeseries A time series with a 6-hourly resolution.

Value

A data frame containing the daily maximum values.

Examples

```
# Example usage:
timeseries <- ArdecheStMartin
max_daily_value(timeseries)
```

 RhoneLyon

Simulated river discharge of the Rhone river at Lyon

Description

A time series of simulated river discharge of the Rhone river close to its confluence with the Saone (longitude = 4.891 \ latitude = 45.772) from 1951 to 2020. Time series extracted from the HERA dataset: <https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>. The Rhône is France's most powerful river, characterized by a significant seasonal variation in flow rates. The Rhône River is crucial for transportation, hydropower generation, and irrigation in the region.

Usage

```
data(RhoneLyon)
```

Format

Two column dataframe: #'

- time (POSIXct timestamp with 6-hourly resolution)
- Q (6-hourly mean discharge in cubic meter per second)

Source

<https://data.jrc.ec.europa.eu/dataset/a605a675-9444-4017-8b34-d66be5b18c95>

 tsEasyParseNamedArgs

Parse named arguments and assign values to a predefined argument structure.

Description

This function takes a list of named arguments and assigns their values to a predefined argument structure. The argument structure is a list with named elements representing the available arguments. If an argument is present in the list of named arguments, its value is assigned to the corresponding element in the argument structure. If an argument is not present, its value in the argument structure remains unchanged.

Usage

```
tsEasyParseNamedArgs(args, argStruct)
```

Arguments

args	A list of named arguments.
argStruct	A list representing the argument structure with named elements.

Value

A modified argument structure with values assigned from the list of named arguments.

Examples

```
args <- list(arg1 = 10, arg2 = "tanargue")
argStruct <- list(arg1 = 0, arg2 = "", arg3 = TRUE)
modifiedArgStruct <- tsEasyParseNamedArgs(args, argStruct)
modifiedArgStruct
```

tsEstimateAverageSeasonality
Estimate Average Seasonality

Description

This function estimates the average seasonality of a time series based on the given parameters.

Usage

```
tsEstimateAverageSeasonality(timeStamps, seasonalitySeries, timeWindow)
```

Arguments

`timeStamps` The time stamps of the time series.
`seasonalitySeries` The series representing the seasonality.
`timeWindow` The time window used for averaging the seasonality.

Value

A list containing the estimated regime and the seasonality series:

`regime` The estimated regime of the time series.

`Seasonality` A data frame containing the average and varying seasonality series.

`averageSeasonalitySeries` The average seasonality series.

`varyingSeasonalitySeries` The varying seasonality series.

Examples

```

timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
timeWindow <- 30*365 # 30 years
rs <- tsEvaDetrendTimeSeries(timeStamps, series, timeWindow)
nRunMn <- rs@nRunMn
cat("computing trend seasonality ... \n")
seasonalitySeries <- rs@detrendSeries
result <- tsEstimateAverageSeasonality(timeStamps, seasonalitySeries, timeWindow=rs@nRunMn)
#plot(result$regime, type = "l", xlab = "Day", ylab = "Regime", main = "Estimated Regime")
#plot(result$Seasonality$averageSeasonalitySeries, type = "l", xlab = "Day",
#ylab = "Seasonality", main = "Average Seasonality")
#plot(result$Seasonality$varyingSeasonalitySeries, type = "l", xlab = "Day",
#ylab = "Seasonality", main = "Varying Seasonality")

```

tsEvaChangepts

Change point detection in time series

Description

This function applies the PELT method for change point detection in a time series. It returns the mean and variance of the series segments between change points.

Usage

```
tsEvaChangepts(series, timeWindow, timeStamps)
```

Arguments

series	A numeric vector representing the time series.
timeWindow	An integer specifying the minimum length of segments.
timeStamps	A vector of timestamps corresponding to the series data points.

Value

A list containing:

trend	A numeric vector of the same length as series, with each segment between change points filled with its mean value
variance	A numeric vector of the same length as series, with each segment between change points filled with its variance
changepts	A vector of timestamps at which change points were detected

Examples

```

timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
timeWindow <- 30*365 # 30 years
result <- tsEvaChangepts(series, timeWindow, timeStamps)
plot(timeAndSeries, type = "l")
lines(timeStamps,result$trend,col=2)
points(timeStamps[result$changepts], result$trend[result$changepts], col = "red")

```

```
tsEvaComputeReturnLevelsGEV
```

```
tsEvaComputeReturnLevelsGEV
```

Description

tsEvaComputeReturnPeriodsGPD is a function that calculates the return levels for a Generalized Extreme Value (GEV) distribution given the GEV parameters and their standard errors. The return periods are specified in a time unit that corresponds to the size of the time segments for evaluating the maxima.

Usage

```

tsEvaComputeReturnLevelsGEV(
  epsilon,
  sigma,
  mu,
  epsilonStdErr,
  sigmaStdErr,
  muStdErr,
  returnPeriodsInDts
)

```

Arguments

epsilon	The shape parameter of the GEV distribution.
sigma	The scale parameter of the GEV distribution.
mu	The location parameter of the GEV distribution.
epsilonStdErr	The standard error of the shape parameter.
sigmaStdErr	The standard error of the scale parameter.
muStdErr	The standard error of the location parameter.
returnPeriodsInDts	The return periods expressed in the corresponding time unit. For example, while working on yearly

Value

A list containing the following components:

returnLevels A matrix of return levels corresponding to the specified return periods.

returnLevelsErr A matrix of standard errors for the return levels.

See Also

[empdis](#)

Examples

```
# Example usage with some sample data
epsilon <- c(0.1)
sigma <- c(2.3)
mu <- c(1.3)
epsilonStdErr <- c(0.01)
sigmaStdErr <- c(0.11)
muStdErr <- c(0.011)
returnPeriodsInDts <- c( 5, 10, 20, 50)
results <- tsEvaComputeReturnLevelsGEV(epsilon, sigma, mu, epsilonStdErr,
sigmaStdErr, muStdErr, returnPeriodsInDts)
head(results$returnLevels)
head(results$returnLevelsErr)
```

tsEvaComputeReturnLevelsGEVFromAnalysisObj

tsEvaComputeReturnLevelsGEVFromAnalysisObj

Description

tsEvaComputeReturnLevelsGEVFromAnalysisObj is a function that calculates the return levels for a Generalized Extreme Value (GEV) distribution using the parameters obtained from a non-stationary extreme value analysis. It supports non-stationary analysis by considering different parameters for each time index.

Usage

```
tsEvaComputeReturnLevelsGEVFromAnalysisObj(
  nonStationaryEvaParams,
  returnPeriodsInYears,
  timeIndex = -1
)
```

Arguments

- `nonStationaryEvaParams` The parameters obtained from a non-stationary extreme value analysis.
- `returnPeriodsInYears` The return periods expressed in years.
- `timeIndex` Temporal index corresponding to the time step on which compute the GEV RLs.

Value

A list containing the following components:

- `returnLevels` A matrix of return levels corresponding to the specified return periods.
- `returnLevelsErr` A matrix of standard errors for the return levels.
- `returnLevelsErrFit` A matrix of standard errors for the return levels obtained from fitting the non-stationary model.
- `returnLevelsErrTransf` A matrix of standard errors for the return levels obtained from the transformed data.

See Also

[tsEvaComputeReturnLevelsGEV](#)

Examples

```
# Example usage with some sample data
nonStationaryEvaParams <- list(list(
  parameters = list(
    epsilon = 0.1,
    sigma = c(2.1, 2.2, 2.3),
    mu = c(1.1, 1.2, 1.3),
    timeHorizonStart=as.POSIXct("1951-01-01"),
    timeHorizonEnd=as.POSIXct("2020-12-31"),
    nPeaks=90
  ),
  paramErr = list(
    epsilonErr = 0.01,
    sigmaErr = c(0.11, 0.12, 0.13),
    muErr = c(0.011, 0.012, 0.013)
  ),NA
)
returnPeriodsInYears <- c(1, 5, 10, 20, 50)
timeIndex=1
results <- tsEvaComputeReturnLevelsGEVFromAnalysisObj(nonStationaryEvaParams, returnPeriodsInYears)
head(results$returnLevels)
```

```
tsEvaComputeReturnLevelsGPD
    tsEvaComputeReturnLevelsGPD
```

Description

#' `tsEvaComputeReturnLevelsGPD` is a function that compute the return levels for a Generalized Pareto Distribution (GPD) using the parameters of the distribution and their standard errors.

Usage

```
tsEvaComputeReturnLevelsGPD(
  epsilon,
  sigma,
  threshold,
  epsilonStdErr,
  sigmaStdErr,
  thresholdStdErr,
  nPeaks,
  sampleTimeHorizon,
  returnPeriods
)
```

Arguments

<code>epsilon</code>	The shape parameter of the GPD.
<code>sigma</code>	The scale parameter of the GPD.
<code>threshold</code>	The threshold parameter of the GPD.
<code>epsilonStdErr</code>	The standard error of the shape parameter.
<code>sigmaStdErr</code>	The standard error of the scale parameter.
<code>thresholdStdErr</code>	The standard error of the threshold parameter.
<code>nPeaks</code>	The number of peaks used to estimate the parameters.
<code>sampleTimeHorizon</code>	The time horizon of the sample in the same units as the return periods (e.g., years).
<code>returnPeriods</code>	The return periods for which to compute the return levels.

Details

`sampleTimeHorizon` and `returnPeriods` must be in the same units, e.g. years

Value

A list containing the following components:

`returnLevels` A vector of return levels corresponding to the specified return periods.

`returnLevelsErr` A vector of standard errors for the return levels.

Examples

```
# Example usage with some sample data
epsilon <- c(0.1)
sigma <- c(2.3)
threshold <- c(1.3)
epsilonStdErr <- c(0.01)
sigmaStdErr <- c(0.11)
thresholdStdErr <- c(0.011)
returnPeriodsInDts <- c( 5, 10, 20, 50)
nPeaks=70
SampleTimeHorizon=70
results <- tsEvaComputeReturnLevelsGPD(epsilon, sigma, threshold, epsilonStdErr,
sigmaStdErr, thresholdStdErr, nPeaks, SampleTimeHorizon, returnPeriodsInDts)
head(results$returnLevels)
head(results$returnLevelsErr)
```

`tsEvaComputeReturnLevelsGPDFFromAnalysisObj`

tsEvaComputeReturnLevelsGPDFFromAnalysisObj

Description

`tsEvaComputeReturnLevelsGPDFFromAnalysisObj` is a function that calculates the return levels for a Generalized Pareto Distribution (GPD) using the parameters obtained from an analysis object. It takes into account non-stationarity by considering time-varying parameters and their associated standard errors.

Usage

```
tsEvaComputeReturnLevelsGPDFFromAnalysisObj(
  nonStationaryEvaParams,
  returnPeriodsInYears,
  timeIndex = -1
)
```

Arguments

`nonStationaryEvaParams`

The non-stationary parameters obtained from the analysis object.

`returnPeriodsInYears`

The return periods for which to compute the return levels, expressed in years.

`timeIndex`

Temporal index corresponding to the time step on which compute the GEV RLs.

Value

A list with the following components:

`returnLevels` A vector of return levels corresponding to the specified return periods.

`returnLevelsErrFit` A vector of standard errors for the return levels estimated based on the fit.

`returnLevelsErrTransf` A vector of standard errors for the return levels estimated based on the transformed parameters.

See Also

[tsEvaComputeReturnLevelsGPD](#)

Examples

```
# Example usage with some sample data
nonStationaryEvaParams <- list(NA, list(
  parameters = list(
    epsilon = 0.1,
    sigma = c(2.1, 2.2, 2.3),
    threshold = c(1.1, 1.2, 1.3),
    timeHorizonStart=as.POSIXct("1951-01-01"),
    timeHorizonEnd=as.POSIXct("2020-12-31"),
    nPeaks=90
  ),
  paramErr = list(
    epsilonErr = 0.01,
    sigmaErr = c(0.11, 0.12, 0.13),
    thresholdErr = c(0.011, 0.012, 0.013)
  )
)
returnPeriodsInYears <- c(1, 5, 10, 20, 50)
timeIndex=1
results <- tsEvaComputeReturnLevelsGPDFromAnalysisObj(nonStationaryEvaParams, returnPeriodsInYears)
head(results$returnLevels)
```

tsEvaComputeReturnPeriodsGEV

tsEvaComputeReturnPeriodsGEV

Description

`tsEvaComputeReturnPeriodsGEV` is a function that computes the return periods of a set of observations (can be Annual maxima or others) for a Generalized Extreme Value (GEV) distribution, given the GEV parameters and their standard error. The return levels represent the values of annual maxima with a certain probability, while the return periods indicate the average time between exceedances of those threshold values.

Usage

```
tsEvaComputeReturnPeriodsGEV(epsilon, sigma, mu, BlockMax)
```

Arguments

epsilon	The shape parameter of the GEV distribution.
sigma	The scale parameter of the GEV distribution.
mu	The location parameter of the GEV distribution.
BlockMax	A vector containing the block maxima data.

Value

A list containing the following components:

GevPseudo A matrix of pseudo observations obtained from the GEV distribution for each annual extreme at every time step.

returnPeriods A matrix of return periods corresponding to the pseudo observations.

PseudoObs The pseudo observation corresponding to the maximum value used in the computation.

See Also

[empdis](#)

Examples

```
# Example usage with some sample data
epsilon <- 0.1
sigma <- 2.2
mu <- 1.3
BlockMax <- c(10, 20, 30, 40, 50)

results <- tsEvaComputeReturnPeriodsGEV(epsilon, sigma, mu, BlockMax)
head(results$GevPseudo)
head(results$returnPeriods)
head(results$PseudoObs)
```

tsEvaComputeReturnPeriodsGPD

tsEvaComputeReturnPeriodsGPD

Description

tsEvaComputeReturnPeriodsGPD is a function that computes the return periods of a set of observations (peaks) for a Generalized Pareto Distribution (GPD), given the GPD parameters, threshold, peaks data, and sample time horizon.

Usage

```
tsEvaComputeReturnPeriodsGPD(
  epsilon,
  sigma,
  threshold,
  peaks,
  nPeaks,
  peaksID,
  sampleTimeHorizon
)
```

Arguments

epsilon	The shape parameter of the GPD.
sigma	The scale parameter of the GPD.
threshold	The threshold value for the GPD.
peaks	A vector containing the peak values.
nPeaks	The number of peak values.
peaksID	An identifier for each peak value.
sampleTimeHorizon	The time horizon of the sample.

Value

A list containing the following components:

GpdPseudo A matrix of pseudo observations obtained from the GPD for each peak value at every time step.

returnPeriods A matrix of return periods corresponding to the pseudo observations.

PseudoObs A data frame containing the pseudo observations and their corresponding identifiers.

See Also

[empdis](#)

Examples

```
# Example usage with some sample data
epsilon <- 0.1
sigma <- 2.2
threshold <- 1.3
peaks <- c(10, 20, 30, 40, 50)
nPeaks=5
peaksID=c(230,550,999,1540,3012)
SampleTimeHorizon = 70

results <- tsEvaComputeReturnPeriodsGPD(epsilon, sigma, threshold, peaks,
nPeaks, peaksID, SampleTimeHorizon)
```

```
head(results$GpdPseudo)
head(results$returnPeriods)
head(results$PseudoObs)
```

tsEvaComputeRLsGEVGPD *tsEvaComputeRLsGEVGPD*

Description

tsEvaComputeRLsGEVGPD is a function that calculates the return levels and their associated errors for a Generalized Extreme Value (GEV) and Generalized Pareto (GPD) distribution using the parameters obtained from a non-stationary extreme value analysis. It supports non-stationary analysis by considering different parameters for each time index.

Usage

```
tsEvaComputeRLsGEVGPD(nonStationaryEvaParams, RPgoal, timeIndex, trans = NA)
```

Arguments

nonStationaryEvaParams	The parameters obtained from a non-stationary extreme value analysis.
RPgoal	The target return period for which the return levels are computed.
timeIndex	The index at which the time-varying analysis should be estimated.
trans	A character string indicating the transformation to be applied to the data before fitting the EVD. default value is NA, corresponding to no transformation. Currently only the "rev" for reverse transformation is implemented.

Value

A list containing the following components:

Fit	A character string indicating whether the EVD could be fitted to the data ("No fit") or the EVD was successfully fitted to the data ("Fitted").
ReturnLevels	A data frame containing the target return period (ReturnPeriod), GEV return level (GEV), GPD return level (GPD), GEV return level error (errGEV), and GPD return level error (errGPD) for the specified time index.
Params	A list containing the GEV and GPD parameters for the specified time index, including their standard errors.

See Also

[tsEvaComputeReturnLevelsGEV](#), [tsEvaComputeReturnLevelsGPD](#)

tsEvaComputeTimeRP *tsEvaComputeTimeRP*

Description

tsEvaComputeTimeRP is a function that calculates the return period of a given event for GEV and GPD distributions at a given time index.

Usage

```
tsEvaComputeTimeRP(params, RPiGEV, RPiGPD)
```

Arguments

params	A data frame containing the following parameters: epsilonGEV Shape parameter for the Generalized Extreme Value (GEV) distribution. muGEV Location parameter for the GEV distribution. sigmaGEV Scale parameter for the GEV distribution. epsilonGPD Shape parameter for the Generalized Pareto (GPD) distribution. thresholdGPD Threshold parameter for the GPD distribution. sigmaGPD Scale parameter for the GPD distribution. nPeaks Number of peaks in the sample time horizon. SampleTimeHorizon Total number of years in the data sample.
RPiGEV	Value of RP for the GEV distribution.
RPiGPD	Value of RP for the GPD distribution.

Value

A vector with the calculated return period for GEV and GPD distributions.

Examples

```
#Parameter vector:
params <- t(data.frame(epsilonGEV = 0.2, muGEV = 3, sigmaGEV = 1,
                      epsilonGPD = 0.2, thresholdGPD = 3, sigmaGPD = 1,
                      nPeaks = 70, SampleTimeHorizon = 70))

tsEvaComputeTimeRP(params, RPiGEV = 10, RPiGPD = 10)
```

 tsEvaDetrendTimeSeries

Detrend a Time Series

Description

This function detrends a time series by subtracting the trend component from the original series.

Usage

```
tsEvaDetrendTimeSeries(
  timeStamps,
  series,
  timeWindow,
  percent = NA,
  fast = TRUE
)
```

Arguments

timeStamps	A vector of time stamps for the time series.
series	The original time series.
timeWindow	The size of the moving window used to calculate the trend.
percent	The percentile value used to calculate the trend for extreme values. Default is NA.
fast	A logical value indicating whether to print additional information. Default is FALSE.

Value

An object of class "tsTrend" with the following components:

originSeries The original time series

detrendSeries The detrended time series

trendSeries The trend component of the time series

nRunMn The number of data points in the moving window used to calculate the trend

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
timeWindow <- 365*30
detrended <- tsEvaDetrendTimeSeries(timeStamps, series, timeWindow)
```

tsEvaFillSeries	<i>Fill missing values in a time series using a moving average approach.</i>
-----------------	--

Description

This function takes a vector of timestamps and a corresponding series with missing values, and fills the missing values by taking the average of the surrounding values.

Usage

```
tsEvaFillSeries(timestamps, series)
```

Arguments

timestamps	A vector of timestamps.
series	A vector representing the time series with missing values.

Value

A vector with missing values filled using a moving average approach.

Examples

```
timestamps <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
series <- c(1, 2, NA, 4, 5, NA, 7, 8, NA, 10)
filledSeries <- tsEvaFillSeries(timestamps, series)
filledSeries
```

tsEvaFindTrendThreshold	<i>Find Trend Threshold</i>
-------------------------	-----------------------------

Description

This function calculates the optimal trend threshold for a given time series.

Usage

```
tsEvaFindTrendThreshold(series, timestamps, timeWindow)
```

Arguments

series	The time series data.
timestamps	The timestamps corresponding to the time series data.
timeWindow	The time window for detrending the time series.

Details

This function iterates over different percentiles and calculates the threshold based on each percentile. It then removes data points below the threshold and detrends the time series using the specified time window. The function calculates the correlation between the normalized trend and the time series and stores the correlation coefficient for each percentile. It performs a changepoint analysis to determine if there is a significant change in the correlation coefficients. If a change point is found, the function returns the percentile corresponding to the change point. If no change point is found, the function returns the percentile with the highest correlation coefficient. If there are negative values in the detrended time series, the function returns the percentile with the fewest negative values.

Value

The trend threshold value.

Examples

```
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 30 last years
yrs <- as.integer(format(timeAndSeries[,1], "%Y"))
tokeep <- which(yrs>=1990)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 10*365 # 10 years
timeStamps <- timeAndSeries[,1]
series <- timeAndSeries[,2]
tsEvaFindTrendThreshold(series, timeStamps, timeWindow)
```

tsEvaNanRunnigBlowTh *Calculate the return period of low flow based on a threshold and window size*

Description

This function calculates the return period of low flow for a given time series based on a threshold and window size. It uses a sliding window approach to count the number of values below the threshold within each window, and then calculates the return period based on the proportion of values below the threshold. Assumes that the input data has a 7 days timestep.

Usage

```
tsEvaNanRunnigBlowTh(series, threshold, windowSize)
```

Arguments

series	The time series data.
threshold	The threshold value for low flow.
windowSize	The size of the sliding window.

Value

A data frame with two columns: "time" representing the time points corresponding to the sliding windows, and "RP" representing the calculated return period of low flow.

Examples

```
series <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
threshold <- 5
windowSize <- 3
tsEvaNanRunnigBlowTh(series, threshold, windowSize)
```

tsEvaNanRunningMean *Calculate the running mean of a time series with NaN handling*

Description

This function calculates the running mean of a time series, taking into account NaN values. It uses a sliding window approach to calculate the mean, where the window size is specified by the user. If the number of non-NaN values within the window is greater than a threshold, the mean is calculated. Otherwise, NaN is returned.

Usage

```
tsEvaNanRunningMean(series, windowSize)
```

Arguments

series	The input time series
windowSize	The size of the sliding window

Value

A vector containing the running mean of the time series

Examples

```
series <- c(1,2,NaN,4,5,6,NaN,8,9,4,5,6,7,3,9,1,0,4,5,2)
windowSize <- 3
result <- tsEvaNanRunningMean(series, windowSize)
print(result)
```

```
tsEvaNanRunningPercentiles
      tsEvaNanRunningPercentiles
```

Description

Computes a running percentile for a given series using a window with a specified size.

Usage

```
tsEvaNanRunningPercentiles(timeStamps, series, windowSize, percent)
```

Arguments

timeStamps	The timestamps of the series.
series	The input series.
windowSize	The size of the window for the running percentile. Must be greater than or equal to 100.
percent	The percent level to which the percentile is computed.

Details

This function computes a running percentile for a given series using a window with a specified size. The running percentile is computed by interpolating the percentile value for the requested percentage based on the quitting values and incoming values in the window. The function also performs smoothing on the output and calculates the standard error.

The function uses the following label parameters:

percentDelta Delta for the computation of a percentile interval around the requested percentage. If the windowSize is greater than 2000, percentDelta is set to 1. If the windowSize is between 1000 and 2000, percentDelta is set to 2. If the windowSize is between 100 and 1000, percentDelta is set to 5.

nLowLimit Minimum number of non-NA elements for a window for percentile computation

Value

A list containing the approximated running percentile (nrprcnt) and the standard error (stdError).

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
windowSize <- 365
percent <- 90
result <- tsEvaNanRunningPercentiles(timeStamps, series, windowSize, percent)
print(result$nrprcnt)
```

```
print(result$stdError)
```

```
tsEvaNanRunningStatistics  
  tsEvaNanRunningStatistics
```

Description

Returns the moving statistical momentums to the forth.

Usage

```
tsEvaNanRunningStatistics(series, windowSize)
```

Arguments

series	The input time series data.
windowSize	The size of the moving window.

Details

This function calculates the running variance, running third statistical momentum, and running fourth statistical momentum for a given time series data using a moving window approach. The window size determines the number of observations used to calculate the statistics at each point.

Value

A data frame containing the following running statistics:

```
rnvar running variance  
rn3mom running third statistical momentum  
rn4mom running fourth statistical momentum
```

Examples

```
series <- c(1,2,NaN,4,5,6,NaN,8,9,4,5,6,7,3,9,1,0,4,5,2)  
windowSize <- 3  
tsEvaNanRunningStatistics(series, windowSize)
```

`tsEvaNanRunningVariance`*Calculate the running variance of a time series with NaN handling*

Description

This function calculates the running variance of a time series, taking into account NaN values. The series must be zero-averaged before passing it to this function.

Usage

```
tsEvaNanRunningVariance(series, windowSize)
```

Arguments

<code>series</code>	The time series data.
<code>windowSize</code>	The size of the window used for calculating the running variance.

Value

A vector containing the running variance values.

Examples

```
series <- c(1,2,NaN,4,5,6,NaN,8,9,4,5,6,7,3,9,1,0,4,5,2)
windowSize <- 3
tsEvaNanRunningVariance(series, windowSize)
```

`TsEvaNs`*TsEvaNs Function*

Description

This function performs non-stationary extreme value analysis (EVA) on a time series data.

Usage

```
TsEvaNs(  
  timeAndSeries,  
  timeWindow,  
  transfType = "trendPeaks",  
  minPeakDistanceInDays = 10,  
  seasonalityVar = NA,  
  minEventsPerYear = -1,  
  gevMaxima = "annual",
```

```

ciPercentile = 90,
gevType = "GEV",
evdType = c("GEV", "GPD"),
tail = "high",
epy = -1,
lowdt = 7,
trans = NULL,
TrendTh = NA
)

```

Arguments

<code>timeAndSeries</code>	A data frame containing the timestamp and corresponding series data.
<code>timeWindow</code>	The time window for analysis.
<code>transfType</code>	The transformation type for non-stationary EVA. It can be one of the following: <ul style="list-style-type: none"> <code>trend</code> Long-term variations of the timeseries <code>seasonal</code> Long-term and seasonal variations of extremes <code>trendCIPercentile</code> Long-term variations of extremes using a specified percentile <code>trendPeaks</code> Long-term variations of the peaks
<code>minPeakDistanceInDays</code>	The minimum peak distance in days.
<code>seasonalityVar</code>	A logical value indicating whether to consider seasonality in the analysis.
<code>minEventsPerYear</code>	The minimum number of events per year.
<code>gevMaxima</code>	The type of maxima for the GEV distribution (annual or monthly, default is annual).
<code>ciPercentile</code>	The percentile value for confidence intervals.
<code>gevType</code>	The type of GEV distribution (GEV or GPD).
<code>evdType</code>	The type of extreme value distribution (GEV or GPD).
<code>tail</code>	The mode of the analysis (e.g., high for flood peaks or low for drought peaks).
<code>epy</code>	The average number of events per year, can be specified by the user or automatically set according to the tail selected.
<code>lowdt</code>	The temporal resolution used for low values. default is 7 days.
<code>trans</code>	The transformation used to fit the EVD. Can be: <ul style="list-style-type: none"> <code>ori</code> use of original data <code>rev</code> Reversing the data (used for low extremes) <code>inv</code> inversing the data (used for low extreme, can lead to unstabilities) <code>lninv</code> log of inverse the data (used for low extreme, under development)
<code>TrendTh</code>	The threshold used to compute the trend on extreme events (only valid if <code>transfType==trendPeaks</code>). If not specified, the optimal threshold is identified within the function

Details

The function takes a time series data and performs non-stationary EVA using various transformation types and parameters depending on the input data provided. Results are returned as a list containing the non-stationary EVA parameters and the transformed data for stationary EVA and can be used as input for further analysis. In particular for the following function

Value

A list containing the results of the non-stationary EVA. Containing the following components:

`nonStationaryEvaParams` The estimated parameters for non-stationary EVA. Parameters include GEV and GPD parameters for each timestep, confidence intervals, and other statistical measures

`stationaryTransformData` The transformed data for stationary EVA. Includes the stationary series, trend, and standard deviation series

References

Mentaschi, L., Vousdoukas, M., Voukouvalas, E., Sartini, L., Feyen, L., Besio, G., and Alfieri, L. (2016). The transformed-stationary approach: a generic and simplified methodology for non-stationary extreme value analysis. *Hydrology and Earth System Sciences*, **20**(9), 3527-3547. doi:10.5194/hess-20-3527-2016.

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 30 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=1990)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 10*365 # 10 years
result <- TsEvaNs(timeAndSeries, timeWindow,
  transfType = 'trendPeaks',tail = 'high')
```

tsEvaPlotAllRLevelsGEV

tsEvaPlotAllRLevelsGEV

Description

`tsEvaPlotAllRLevelsGEV` is a function that generates a beam plot of return levels for a Generalized Extreme Value (GEV) distribution based on the provided parameters and data. The plot showcases the evolving relationship between return periods and return levels through time, allowing for visual analysis of extreme events and their probabilities.

Usage

```
tsEvaPlotAllRLevelsGEV(
  nonStationaryEvaParams,
  stationaryTransformData,
  rlvmax,
  timeIndex,
  timeStamps,
  tstamps,
  trans,
  ...
)
```

Arguments

nonStationaryEvaParams	A list of non-stationary evaluation parameters containing the GEV distribution parameters (epsilon, sigma, mu) and the time delta in years (dtSampleYears).
stationaryTransformData	The stationary transformed data used for the analysis.
rlvmax	The maximum return level data, including the return periods (haz.RP) and the actual return levels (QNS).
timeIndex	The index of the time step used for analysis.
timeStamps	The timestamps corresponding to the time steps in the analysis.
tstamps	The timestamps used for labeling the plot.
trans	The transformation used to fit the EVD, either "ori" (original) or "rev" (reverse).
...	Additional optional arguments for customizing the plot.

Value

A plot object showing the relationship between return periods and return levels for the GEV distribution at different timesteps.

See Also

[tsEvaComputeReturnLevelsGEV](#)

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 20 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=2000)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 5*365 # 5 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
```

```

transfType = 'trendPeaks', tail = 'high')
nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
amax <- nonStationaryEvaParams[[1]]$parameters$annualMax
amaxID <- nonStationaryEvaParams[[1]]$parameters$annualMaxIndx
timeStamps <- stationaryTransformData$timeStamps
trendPeaks <- stationaryTransformData$trendSeries[amaxID]
stdPeaks <- stationaryTransformData$stdDevSeries[amaxID]
amaxCor <- (amax - trendPeaks) / stdPeaks
nYears <- length(amaxCor)
rlvImax <- empdis(amaxCor, nYears)
rlvImax$QNS <- amax[order(amax)]
rlvImax$Idt <- stationaryTransformData$timeStamps[amaxID][order(amax)]
timeIndex <- 2
tstamps <- "Example Timestamps"
trans <- "ori"
# Call the function with the defined arguments
result <- tsEvaPlotAllRLevelsGEV(
  nonStationaryEvaParams, stationaryTransformData,
  rlvImax, timeIndex, timeStamps, tstamps,
  trans)
result

```

tsEvaPlotAllRLevelsGPD

tsEvaPlotAllRLevelsGPD

Description

tsEvaPlotAllRLevelsGPD is a function that generates a plot of return levels for a Generalized Pareto Distribution (GPD) based on the provided parameters and data. The plot showcases the evolving relationship between return periods and return levels, allowing for visual analysis of extreme events and their probabilities.

Usage

```

tsEvaPlotAllRLevelsGPD(
  nonStationaryEvaParams,
  stationaryTransformData,
  rlvImax,
  timeIndex,
  timeStamps,
  tstamps,
  trans,
  ...
)

```

Arguments

nonStationaryEvaParams	A list of non-stationary evaluation parameters containing the GPD distribution parameters (epsilon, sigma, threshold), time horizon start and end (thStart, thEnd), time horizon in years (timeHorizonInYears), and number of peaks (nPeaks).
stationaryTransformData	The stationary transformed data used for the analysis.
r1vmax	The maximum return level data, including the return periods (haz.RP) and the actual return levels (QNS).
timeIndex	The index of the time step used for analysis.
timeStamps	The timestamps corresponding to the time steps in the analysis.
tstamps	The timestamps used for labeling the plot.
trans	The transformation used to fit the EVD, either "ori" (original) or "rev" (reverse).
...	Additional optional arguments for customizing the plot.

Value

A plot object showing the relationship between return periods and return levels for the GPD distribution at different timest

See Also

[tsEvaComputeReturnLevelsGPD](#)

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 20 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=2000)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 5*365 # 5 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
                    transfType = 'trendPeaks',tail = 'high')
nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
peax <- nonStationaryEvaParams[[2]]$parameters$peaks
peaxID <- nonStationaryEvaParams[[2]]$parameters$peakID
timeStamps <- stationaryTransformData$timeStamps
trendPeaks <- stationaryTransformData$trendSeries[peaxID]
stdPeaks <- stationaryTransformData$stdDevSeries[peaxID]
peaksCor <- (peax - trendPeaks) / stdPeaks
nYears <- round(length(timeStamps) / 365.25 )
r1v1max <- empdis(peaksCor, nYears)
r1v1max$QNS <- peax[order(peax)]
r1v1max$Idt <- stationaryTransformData$timeStamps[peaxID][order(peax)]
```

```

timeIndex <- 2
tstamps <- "Example Timestamps"
trans <- "ori"
# Call the function with the defined arguments
result <- tsEvaPlotAllRLevelsGPD(
  nonStationaryEvaParams, stationaryTransformData,
  rlvmax, timeIndex, timeStamps, tstamps,
  trans)
# Plot the result
result

```

```
tsEvaPlotGEVImageSc  tsEvaPlotGEVImageSc
```

Description

tsEvaPlotGEVImageSc is a function that generates a plot of the Generalized Extreme Value (GEV) distribution with evolving parameters using the provided data.

Usage

```

tsEvaPlotGEVImageSc(
  Y,
  timeStamps,
  serix,
  epsilon,
  sigma,
  mu,
  returnPeriodInDts,
  maxObs,
  trans,
  varargin
)

```

Arguments

Y	A vector of extreme values.
timeStamps	A vector of timestamps corresponding to the extreme values.
serix	The y-value at which to draw a horizontal line on the plot.
epsilon	A numeric value representing the shape parameter of the GEV distribution.
sigma	A vector of scale parameters corresponding to the timestamps.
mu	A vector of location parameters corresponding to the timestamps.
returnPeriodInDts	The return period in decimal time steps.
maxObs	A data frame containing the maximum observations.
trans	A character string indicating the transformation for the plot. Possible values are "rev" (reverse), "inv" (inverse), "linv" (log of inverse) and "ori"(original).
varargin	Additional arguments to customize the plot.

Value

A ggplot object representing the GEV plot with a raster image.

See Also

[tsEvaPlotGEVImageScFromAnalysisObj](#)

`tsEvaPlotGEVImageScFromAnalysisObj`

tsEvaPlotGEVImageScFromAnalysisObj

Description

`tsEvaPlotGEVImageScFromAnalysisObj` is a function that generates a GEV (Generalized Extreme Value) time-varying distribution through time as and show the evolution of exceedance probabilities.

Usage

```
tsEvaPlotGEVImageScFromAnalysisObj(  
  Y,  
  nonStationaryEvaParams,  
  stationaryTransformData,  
  trans,  
  ...  
)
```

Arguments

<code>Y</code>	The input data.
<code>nonStationaryEvaParams</code>	A list of non-stationary evaluation parameters.
<code>stationaryTransformData</code>	The stationary transform data.
<code>trans</code>	The transformation method.
<code>...</code>	Additional arguments.

Value

The GEV image scatter plot.

See Also

[tsEvaPlotGEVImageSc](#)

Examples

```

# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 20 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=2000)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 5*365 # 10 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
transfType = 'trendPeaks',tail = 'high')

# Define the required function arguments
stationaryTransformData <- TSEVA_data[[2]]
nonStationaryEvaParams <- TSEVA_data[[1]]
trans='ori'

ExRange= c(min(nonStationaryEvaParams$potObj$parameters$peaks),
max(nonStationaryEvaParams$potObj$parameters$peaks))
Y <- c(seq(min(ExRange),max(ExRange),length.out=700))

result = tsEvaPlotGEVImageScFromAnalysisObj(Y,nonStationaryEvaParams,
stationaryTransformData, trans)
result

```

```
tsEvaPlotGPDImageSc  tsEvaPlotGPDImageSc
```

Description

tsEvaPlotGPDImageSc is a function that generates a time series plot of the Generalized Pareto Distribution (GPD) with evolving parameters using the provided data.

Usage

```

tsEvaPlotGPDImageSc(
  Y,
  timeStamps,
  serix,
  epsilon,
  sigma,
  threshold,
  peakplot,
  trans,
  varargin
)

```

Arguments

Y	A vector of values.
timeStamps	A vector of timestamps corresponding to the values.
serix	A vector of series values.
epsilon	A numeric value representing the shape parameter of the GPD.
sigma	A vector of standard deviation values.
threshold	A vector of threshold values.
peakplot	A data frame containing peak values and their corresponding timestamps.
trans	A character string indicating the transformation to be applied to the data.
varargin	Additional optional arguments.

Value

A ggplot object representing the GPD plot.

See Also

[tsEvaPlotGPDImageScFromAnalysisObj](#)

tsEvaPlotGPDImageScFromAnalysisObj

tsEvaPlotGPDImageScFromAnalysisObj

Description

tsEvaPlotGPDImageScFromAnalysisObj is a function that plots the GPD (Generalized Pareto Distribution) time-varying distribution through time as and show the evolution of exceedance probabilities.

Usage

```
tsEvaPlotGPDImageScFromAnalysisObj(  
  Y,  
  nonStationaryEvaParams,  
  stationaryTransformData,  
  trans,  
  ...  
)
```

Arguments

Y	The input data.
nonStationaryEvaParams	A list containing non-stationary evaluation parameters.
stationaryTransformData	A data frame containing stationary transform data.
trans	The transformation method to be applied to the data.
...	Additional arguments to be passed to the tsEvaPlotGPDImageSc function.

Details

This function takes the input data Y, non-stationary evaluation parameters nonStationaryEvaParams, stationary transform data stationaryTransformData, transformation method trans, and additional arguments It then updates the arguments with the passed-in values, calculates the time stamps, and performs necessary transformations. Finally, it plots the GPD image score using the [tsEvaPlotGPDImageSc](#) function and returns the plot object.

Value

The plot object.

See Also

[tsEvaPlotGPDImageSc](#)

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 20 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=2000)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 5*365 # 5 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
  transType = 'trendPeaks',tail = 'high')
nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
trans='ori'
ExRange= c(min(nonStationaryEvaParams$potObj$parameters$peaks),
  max(nonStationaryEvaParams$potObj$parameters$peaks))
Y <- c(seq(min(ExRange),max(ExRange),length.out=700))
result = tsEvaPlotGEVImageScFromAnalysisObj(Y, nonStationaryEvaParams,
  stationaryTransformData, trans)
result
```

 tsEvaPlotReturnLevelsGEV

tsEvaPlotReturnLevelsGEV

Description

tsEvaPlotReturnLevelsGEV is a function that plots the return levels using the Generalized Extreme Value (GEV) distribution.

Usage

```
tsEvaPlotReturnLevelsGEV(
  epsilon,
  sigma,
  mu,
  epsilonStdErr,
  sigmaStdErr,
  muStdErr,
  rlvmax,
  tstamp,
  trans,
  ...
)
```

Arguments

epsilon	The shape parameter of the GEV distribution.
sigma	The scale parameter of the GEV distribution.
mu	The location parameter of the GEV distribution.
epsilonStdErr	The standard error of the shape parameter.
sigmaStdErr	The standard error of the scale parameter.
muStdErr	The standard error of the location parameter.
rlvmax	A data frame containing the return levels of annual maxima.
tstamp	The title for the plot.
trans	The transformation used to fit the EVD, either "ori" (original) or "rev" (reverse). "inv" and "linv" are also available but in development phase.
...	Additional arguments to be passed to the function.

Value

A ggplot object representing the plot of return levels.

See Also

[tsEvaComputeReturnLevelsGEV](#) [tsEvaPlotReturnLevelsGEVFromAnalysisObj](#)

Examples

```

# Define the required function arguments
epsilon <- 0.2
sigma <- 0.5
mu <- 10
epsilonStdErr <- 0.05
sigmaStdErr <- 0.05
muStdErr <- 0.1
rlvmax <- data.frame(
  haz.RP = c(2, 5, 10, 20, 50, 100, 200, 500, 1000),
  Idt = as.POSIXct(as.Date("2000-01-01") + round(runif(9, 0, 21 * 365.25))),
  origin = "1970-01-01"
),
  QNS = c(10, 12, 13, 13.2, 14, 15.7, 16, 16.2, 18)
)
tstamps <- "Example Timestamps"
trans <- "ori"
# Call the function with the defined arguments
result <- tsEvaPlotReturnLevelsGEV(
  epsilon, sigma, mu, epsilonStdErr, sigmaStdErr, muStdErr,
  rlvmax, tstamps, trans
)

# Plot the result
result

```

```

tsEvaPlotReturnLevelsGEVFromAnalysisObj
      tsEvaPlotReturnLevelsGEVFromAnalysisObj

```

Description

tsEvaPlotReturnLevelsGEVFromAnalysisObj is a function that plots the return levels for a Generalized Extreme Value (GEV) distribution using the parameters obtained from an analysis object. It considers non-stationarity by considering time-varying parameters and their associated standard errors.

Usage

```

tsEvaPlotReturnLevelsGEVFromAnalysisObj(
  nonStationaryEvaParams,
  stationaryTransformData,
  timeIndex,
  trans,
  ...
)

```

Arguments

nonStationaryEvaParams	The non-stationary parameters obtained from the analysis object.
stationaryTransformData	The stationary transformed data obtained from the analysis object.
timeIndex	The index at which the time-varying analysis should be estimated.
trans	The transformation used to fit the EVD. Can be "ori" for no transformation or "rev" for reverse transformation.
...	Additional arguments to be passed to the function.

Value

Plot 1 RLtstep: return level curve with confidence interval for the selected timeIndex

Plot 2 beam: beam of return level curve for all with highlighted curve for selected timeIndex

References

Mentaschi, L., Vousdoukas, M., Voukouvalas, E., Sartini, L., Feyen, L., Besio, G., and Alfieri, L. (2016). The transformed-stationary approach: a generic and simplified methodology for non-stationary extreme value analysis. *Hydrology and Earth System Sciences*, 20, 3527-3547. doi:10.5194/hess-20-3527-2016.

See Also

[tsEvaPlotReturnLevelsGEV\(\)](#) and [tsEvaPlotAllRLevelsGEV\(\)](#)

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 30 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=1990)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 10*365 # 10 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
  transType = 'trendPeaks', tail = 'high')
nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
timeIndex=2
trans='ori'
result = tsEvaPlotReturnLevelsGEVFromAnalysisObj(nonStationaryEvaParams, stationaryTransformData,
  timeIndex, trans)
result
```

```
tsEvaPlotReturnLevelsGPD
    tsEvaPlotReturnLevelsGPD
```

Description

tsEvaPlotReturnLevelsGPD is a function that plots the return levels using the Generalized Pareto Distribution (GPD).

Usage

```
tsEvaPlotReturnLevelsGPD(
  epsilon,
  sigma,
  threshold,
  epsilonStdErr,
  sigmaStdErr,
  thresholdStdErr,
  nPeaks,
  timeHorizonInYears,
  rlvmax,
  tstamps,
  trans,
  ...
)
```

Arguments

epsilon	The shape parameter of the GPD.
sigma	The scale parameter of the GPD.
threshold	The threshold parameter of the GPD.
epsilonStdErr	The standard error of the shape parameter.
sigmaStdErr	The standard error of the scale parameter.
thresholdStdErr	The standard error of the threshold parameter.
nPeaks	The number of peaks used in the GPD estimation.
timeHorizonInYears	The time horizon in years for the GPD estimation.
rlvmax	A data frame containing the return levels of annual maxima.
tstamps	The title for the plot.
trans	The transformation type for the return levels.
...	Additional arguments to be passed to the function.

Value

A ggplot object representing the plot of return levels.

See Also

[tsEvaComputeReturnLevelsGPD](#) [tsEvaPlotReturnLevelsGPDFFromAnalysisObj](#)

Examples

```
# Define the required function arguments
epsilon <- 0.2
sigma <- 0.5
threshold <- 10
epsilonStdErr <- 0.05
sigmaStdErr <- 0.05
thresholdStdErr <- 0.1
rlvmax <- data.frame(
  haz.RP = c(2, 5, 10, 20, 50, 100, 200, 500, 1000),
  Idt = as.POSIXct(as.Date("2000-01-01") + round(runif(9, 0, 21 * 365.25))),
  origin = "1970-01-01"
),
  QNS = c(10, 12, 13, 13.2, 14, 15.7, 16, 16.2, 18)
)
tstamps <- "Example Timestamps"
trans <- "ori"
nPeaks=70
SampleTimeHorizon=70
# Call the function with the defined arguments
result <- tsEvaPlotReturnLevelsGPD(
  epsilon, sigma, threshold, epsilonStdErr, sigmaStdErr, thresholdStdErr, nPeaks,
  SampleTimeHorizon, rlvmax, tstamps, trans
)
# Plot the result
result
```

[tsEvaPlotReturnLevelsGPDFFromAnalysisObj](#)

tsEvaPlotReturnLevelsGPDFFromAnalysisObj

Description

`tsEvaPlotReturnLevelsGPDFFromAnalysisObj` is a function that plots the return levels for a Generalized Pareto Distribution (GPD) using the parameters obtained from an analysis object. It considers non-stationarity by considering time-varying parameters and their associated standard errors.

Usage

```
tsEvaPlotReturnLevelsGPDFFromAnalysisObj(
  nonStationaryEvaParams,
  stationaryTransformData,
  timeIndex,
  trans,
  ...
)
```

Arguments

nonStationaryEvaParams	The non-stationary parameters obtained from the analysis object.
stationaryTransformData	The stationary transformed data obtained from the analysis object.
timeIndex	The index at which the time-varying analysis should be estimated.
trans	The transformation used to fit the EVD. Can be "ori" for no transformation or "rev" for reverse transformation.
...	Additional arguments to be passed to the function.

Value

Plot 1 RLstep: return level curve with confidence interval for the selected timeIndex

Plot 2 beam: beam of return level curve for all with highlighted curve for selected timeIndex

References

Mentaschi, L., Vousdoukas, M., Voukouvalas, E., Sartini, L., Feyen, L., Besio, G., and Alfieri, L. (2016). The transformed-stationary approach: a generic and simplified methodology for non-stationary extreme value analysis. *Hydrology and Earth System Sciences*, 20, 3527-3547. doi:10.5194/hess-20-3527-2016.

See Also

[tsEvaPlotReturnLevelsGPD\(\)](#) and [tsEvaPlotAllRLevelsGPD\(\)](#)

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 30 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=1990)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 10*365 # 10 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
  transType = 'trendPeaks', tail = 'high')
```

```

nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
timeIndex=2
trans='ori'
result = tsEvaPlotReturnLevelsGPDFFromAnalysisObj(nonStationaryEvaParams, stationaryTransformData,
timeIndex, trans)
result

```

```

tsEvaPlotSeriesTrendStdDevFromAnalysisObj
tsEvaPlotSeriesTrendStdDevFromAnalysisObj

```

Description

`tsEvaPlotTrendStdDevFromAnalysisObj` is a function that plots a time series along with its trend and standard deviation.

Usage

```

tsEvaPlotSeriesTrendStdDevFromAnalysisObj(
  nonStationaryEvaParams,
  stationaryTransformData,
  trans,
  ...
)

```

Arguments

<code>nonStationaryEvaParams</code>	The non-stationary evaluation parameters.
<code>stationaryTransformData</code>	The stationary transformed data.
<code>trans</code>	The transformation used to fit the EVD, either "ori" (original) or "rev" (reverse). "inv" and "linv" are also available
<code>...</code>	Additional arguments to customize the plot (optional).

Value

A ggplot object representing the plot.

Examples

```

# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 30 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))

```

```

tokeep <- which(yrs>=1990)
timeAndSeries <- timeAndSeries[tokeep,]
timeWindow <- 10*365 # 10 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
  transfType = 'trendPeaks', tail = 'high')
nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
trans='ori'
result = tsEvaPlotSeriesTrendStdDevFromAnalysisObj(nonStationaryEvaParams,
  stationaryTransformData, trans)
result

```

tsEvaPlotTransfToStat *tsEvaPlotTransfToStat*

Description

tsEvaPlotTransfToStat is a function that creates a line plot of time series data along with statistical measures.

Usage

```

tsEvaPlotTransfToStat(
  timeStamps,
  statSeries,
  srsmean,
  stdDev,
  st3mom,
  st4mom,
  varargin
)

```

Arguments

timeStamps	A vector of time stamps for the data points.
statSeries	A vector of the main time series data.
srsmean	A vector of the mean values for each time stamp.
stdDev	A vector of the standard deviation values for each time stamp.
st3mom	A vector of the third moment values for each time stamp.
st4mom	A vector of the fourth moment values for each time stamp.
varargin	Additional optional arguments to customize the plot.

Value

A ggplot object representing the line plot.

See Also

[tsEvaPlotTransfToStatFromAnalysisObj](#)

tsEvaPlotTransfToStatFromAnalysisObj

tsEvaPlotTransfToStatFromAnalysisObj

Description

tsEvaPlotTransfToStatFromAnalysisObj is a function that takes the parameters of a non-stationary time series evaluation, along with the transformed stationary data, and plots the converted stationary series.

Usage

```
tsEvaPlotTransfToStatFromAnalysisObj(
  nonStationaryEvaParams,
  stationaryTransformData,
  ...
)
```

Arguments

nonStationaryEvaParams
A list of parameters for non-stationary time series evaluation.

stationaryTransformData
A list containing the transformed stationary data.

...
Additional arguments to be passed to the [tsEvaPlotTransfToStat](#) function.

Value

The plot object representing the converted stationary series.

See Also

[tsEvaPlotTransfToStat](#)

Examples

```
# Example usage of TsEvaNs function
timeAndSeries <- ArdecheStMartin
#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
#keep only the 30 last years
yrs <- as.integer(format(timeAndSeries$date, "%Y"))
tokeep <- which(yrs>=1990)
timeAndSeries <- timeAndSeries[tokeep,]
```

```

timeWindow <- 10*365 # 10 years
TSEVA_data <- TsEvaNs(timeAndSeries, timeWindow,
  transfType = 'trendPeaks', tail = 'high')
# Define the required function arguments
nonStationaryEvaParams <- TSEVA_data[[1]]
stationaryTransformData <- TSEVA_data[[2]]
nonStationaryEvaParams <- TSEVA_data[[1]]
trans='ori'
ExRange= c(min(nonStationaryEvaParams$potObj$parameters$peaks),
  max(nonStationaryEvaParams$potObj$parameters$peaks))
Y <- c(seq(min(ExRange),max(ExRange),length.out=700))
result = tsEvaPlotTransfToStatFromAnalysisObj (nonStationaryEvaParams,
  stationaryTransformData)
result

```

tsEvaRunningMeanTrend *Calculate the running mean trend of a time series*

Description

This function calculates the running mean trend of a given time series using a specified time window.

Usage

```
tsEvaRunningMeanTrend(timeStamps, series, timeWindow)
```

Arguments

timeStamps	A vector of time stamps corresponding to the observations in the series.
series	A vector of numeric values representing the time series.
timeWindow	The length of the time window (in the same time units as the time stamps) used for calculating the running mean trend.

Value

A list containing the running mean trend series and the number of observations used for each running mean calculation.

Examples

```

timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
timeWindow <- 365*30
result <- tsEvaRunningMeanTrend(timeStamps, series, timeWindow)
result$trendSeries
result$nRunMn

```

tsEvaSampleData	<i>tsEvaSampleData Function</i>
-----------------	---------------------------------

Description

tsEvaSampleData is a function that calculates various statistics and data for time series evaluation.

Usage

```
tsEvaSampleData(
  ms,
  meanEventsPerYear,
  minEventsPerYear,
  minPeakDistanceInDays,
  tail = NA
)
```

Arguments

ms	A matrix containing the time series data.
meanEventsPerYear	The mean number of events per year.
minEventsPerYear	The minimum number of events per year.
minPeakDistanceInDays	The minimum peak distance in days.
tail	The tail to be studied for POT selection, either 'high' or 'low'.

Value

A list containing the following elements:

completeSeries	The complete time series data.
POT	The data for Peaks Over Threshold (POT) analysis.
years	The years in the time series data.
Percentiles	The desired percentiles and their corresponding values.
annualMax	The annual maximum values.
annualMaxDate	The dates corresponding to the annual maximum values.
annualMaxIndx	The indices of the annual maximum values.
monthlyMax	The monthly maximum values.
monthlyMaxDate	The dates corresponding to the monthly maximum values.
monthlyMaxIndx	The indices of the monthly maximum values.

See Also[tsGetPOT\(\)](#)**Examples**

```
# Generate sample data
data <- ArdecheStMartin
colnames(data) <- c("Date", "Value")
#select only the 5 latest years
yrs <- as.integer(format(data$Date, "%Y"))
tokeep <- which(yrs>=2015)
data <- data[tokeep,]
timeWindow <- 365 # 1 year
# Calculate statistics and data
result <- tsEvaSampleData(data, meanEventsPerYear=3, minEventsPerYear=0,
minPeakDistanceInDays=7, "high")
# View the result
print(result)
```

tsEvaTransformSeriesToStationaryMMXTrend

tsEvaTransformSeriesToStationaryMMXTrend

Description

tsEvaTransformSeriesToStationaryMMXTrend transforms a time series to a stationary one by focusing on the monthly maximum values. The trend and slowly varying amplitude are computed on the monthly maximum values.

Usage

```
tsEvaTransformSeriesToStationaryMMXTrend(timeStamps, series, timeWindow)
```

Arguments

timeStamps	A vector of time stamps corresponding to the observations in the series.
series	A vector of the time series data.
timeWindow	The size of the time window used for detrending.

Value

A list containing the following components:

runningStatsMultiplicity	The multiplicity of running statistics.
stationarySeries	The stationary series after removing the trend.
trendSeries	The trend component of the series.
trendSeriesNonSeasonal	NULL (not used).

trendError The error on the trend component.
 stdDevSeries The standard deviation series.
 stdDevSeriesNonSeasonal NULL (not used).
 stdDevError The error on the standard deviation series.
 timeStamps The time stamps.
 nonStatSeries The original non-stationary series.
 statSer3Mom The running mean of the third moment of the stationary series.
 statSer4Mom The running mean of the fourth moment of the stationary series.

See Also

[tsEvaDetrendTimeSeries\(\)](#), [tsEvaNanRunningVariance\(\)](#), [tsEvaNanRunningMean\(\)](#), [tsEvaNanRunningStatistics\(\)](#)

Examples

```

timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
# select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs >= 2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
result <- tsEvaTransformSeriesToStationaryMMXTrend(timeStamps, series, timeWindow)
plot(result$trendSeries)

```

tsEvaTransformSeriesToStationaryMultiplicativeSeasonality

tsEvaTransformSeriesToStationaryMultiplicativeSeasonality

Description

This function decomposes a time series into a season-dependent trend and a season-dependent standard deviation. It performs a transformation from non-stationary to stationary.

Usage

```

tsEvaTransformSeriesToStationaryMultiplicativeSeasonality(
  timeStamps,
  series,
  timeWindow,
  seasonalityVar = TRUE
)

```

Arguments

timeStamps	A vector of timestamps for the time series data.
series	A vector of the time series data.
timeWindow	The size of the moving window used for trend estimation.
seasonalityVar	A logical value indicating whether to consider a time varying seasonality (30 years moving average) or a static seasonal cycle in the transformation. Default is TRUE.

Value

A list containing the transformed data and various statistics and errors.

runningStatsMulteplicity	The size of the moving window used for trend estimation
stationarySeries	The transformed stationary series
trendSeries	The trend component of the transformed series
trendSeriesNonSeasonal	The trend component of the original series without seasonality
stdDevSeries	The standard deviation component of the transformed series
stdDevSeriesNonSeasonal	The standard deviation component of the original series without seasonality
trendNonSeasonalError	The error on the non-seasonal trend component
stdDevNonSeasonalError	The error on the non-seasonal standard deviation component
trendSeasonalError	The error on the seasonal trend component
stdDevSeasonalError	The error on the seasonal standard deviation component
trendError	The overall error on the trend component
stdDevError	The overall error on the standard deviation component
Regime	The estimated regime of the trend seasonality
timeStamps	The input timestamps
nonStatSeries	The original non-stationary series
statSer3Mom	The third moment of the transformed stationary series
statSer4Mom	The fourth moment of the transformed stationary series

transformation non stationary -> stationary

```
transformation stationary -> non stationary y(t) = stdDev(t)*ssn_stdDev(t)*x(t) + trend(t) + ssn_trend(t)
trasfData.trendSeries = trend(t) + ssn_trend(t)
trasfData.stdDevSeries = stdDev(t)*ssn_stdDev(t)
```

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs>=2015)
```

```

timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
TrendTh <- NA
result <- tsEvaTransformSeriesToStationaryMultiplicativeSeasonality(timeStamps,
series, timeWindow, seasonalityVar=FALSE)
plot(result$trendSeries)

```

```

tsEvaTransformSeriesToStationaryPeakTrend
      tsEvaTransformSeriesToStationaryPeakTrend

```

Description

`tsEvaTransformSeriesToStationaryPeakTrend` transforms a time series to a stationary one by focusing on extremes. The trend and slowly varying amplitude are computed on values above a threshold defined by the user or automatically with the function `tsEvaFindTrendThreshold`.

Usage

```

tsEvaTransformSeriesToStationaryPeakTrend(
  timeStamps,
  series,
  timeWindow,
  TrendTh
)

```

Arguments

<code>timeStamps</code>	A vector of time stamps corresponding to the observations in the series.
<code>series</code>	A vector of the time series data.
<code>timeWindow</code>	The size of the time window used for detrending.
<code>TrendTh</code>	The threshold for fitting the trend on the means above a given quantile. If "NA", default to 0.5.

Value

A list containing the following components:

`runningStatsMulteplicity` The multiplicity of running statistics.
`stationarySeries` The stationary series after removing the trend.
`trendSeries` The trend component of the series.
`trendSeriesNonSeasonal` NULL (not used).
`trendError` The error on the trend component.
`stdDevSeries` The standard deviation series.

stdDevSeriesNonSeasonal NULL (not used).
 stdDevError The error on the standard deviation series.
 timeStamps The time stamps.
 nonStatSeries The original non-stationary series.
 statSer3Mom The running mean of the third moment of the stationary series.
 statSer4Mom The running mean of the fourth moment of the stationary series.

See Also

[tsEvaFindTrendThreshold\(\)](#)

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs>=2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
TrendTh <- NA
result <- tsEvaTransformSeriesToStationaryPeakTrend(timeStamps,
series, timeWindow, TrendTh)
plot(result$trendSeries)
```

tsEvaTransformSeriesToStationaryTrendAndChangepts

Transform Time Series to Stationary Trend and Change Points

Description

This function takes a time series and transforms it into a stationary trend series by removing the trend component and detecting change points. It computes the slowly varying standard deviation and normalizes the stationary series before detecting step changes. It also calculates the error on the trend and standard deviation.

Usage

```
tsEvaTransformSeriesToStationaryTrendAndChangepts(
  timeStamps,
  series,
  timeWindow
)
```

Arguments

<code>timeStamps</code>	A vector of time stamps corresponding to the data points in the series.
<code>series</code>	The original time series data.
<code>timeWindow</code>	The size of the time window used for detrending.

Value

A list containing the following elements:

<code>runningStatsMultiplicity</code>	The running statistics multiplicity.
<code>stationarySeries</code>	The transformed stationary series.
<code>trendSeries</code>	The trend series.
<code>trendonlySeries</code>	The trend series without the stationary component.
<code>ChpointsSeries2</code>	The trend component of the change points.
<code>changePoints</code>	The detected change points.
<code>trendSeriesNonSeasonal</code>	The trend series without the seasonal component.
<code>trendError</code>	The error on the trend.
<code>stdDevSeries</code>	The slowly varying standard deviation series.
<code>stdDevSeriesNonStep</code>	The slowly varying standard deviation series without step changes.
<code>stdDevError</code>	The error on the standard deviation.
<code>timeStamps</code>	The time stamps.
<code>nonStatSeries</code>	The original non-stationary series.
<code>statSer3Mom</code>	The running mean of the third moment of the stationary series.
<code>statSer4Mom</code>	The running mean of the fourth moment of the stationary series.

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs>=2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
percentile <- 90
result <- tsEvaTransformSeriesToStationaryTrendAndChangepts(timeStamps,
series, timeWindow)
plot(result$trendSeries)
```

tsEvaTransformSeriesToStationaryTrendAndChangepts_ciPercentile
*Transform Time Series to Stationary Trend and Change Points with
 Confidence Intervals*

Description

This function takes a time series and transforms it into a stationary trend series with change points and confidence intervals.

Usage

```
tsEvaTransformSeriesToStationaryTrendAndChangepts_ciPercentile(  
  timeStamps,  
  series,  
  timeWindow,  
  percentile  
)
```

Arguments

timeStamps	A vector of time stamps corresponding to the observations in the series.
series	The time series data.
timeWindow	The size of the sliding window used for detrending the series.
percentile	The percentile value used for computing the running percentile of the stationary series.

Value

A list containing the following elements:

runningStatsMultiplicity The running statistics multiplicity
 stationarySeries The transformed stationary series
 trendSeries The trend series
 trendonlySeries The trend series without the stationary component
 ChpointsSeries2 The trend series with change points
 changePoints The detected change points
 trendSeriesNonSeasonal The trend series without the seasonal component
 trendError The error on the trend
 stdDevSeries The standard deviation series
 stdDevSeriesNonStep The standard deviation series without the step change component
 stdDevError The error on the standard deviation
 timeStamps The time stamps

`nonStatSeries` The original non-stationary series

`statSer3Mom` The running mean of the third moment of the stationary series

`statSer4Mom` The running mean of the fourth moment of the stationary series

Examples

```
timeAndSeries <- ArdecheStMartin

#go from six-hourly values to daily max
timeAndSeries <- max_daily_value(timeAndSeries)
timeStamps <- timeAndSeries[,1]
series <- timeAndSeries[,2]

#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs>=2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
percentile <- 90
result <- tsEvaTransformSeriesToStationaryTrendAndChangepts_ciPercentile(timeStamps,
series, timeWindow, percentile)
plot(result$trendSeries)
```

```
tsEvaTransformSeriesToStationaryTrendOnly
      tsEvaTransformSeriesToStationaryTrendOnly
```

Description

`tsEvaTransformSeriesToStationaryTrendOnly` is the original detrending function implemented in Mentaschi et al.(2016). It takes a time series and transforms it into a stationary one. It computes the trend as a running average of the time series, the slowly varying amplitude as its standard deviation, and other statistical measures.

Usage

```
tsEvaTransformSeriesToStationaryTrendOnly(timeStamps, series, timeWindow)
```

Arguments

<code>timeStamps</code>	A vector of time stamps for the time series.
<code>series</code>	The original time series.
<code>timeWindow</code>	The size of the time window used for detrending.

Value

A list containing the following elements:

runningStatsMultiplicity The running statistics multiplicity.

stationarySeries The transformed stationary series.

trendSeries The trend series.

trendSeriesNonSeasonal The non-seasonal trend series.

trendError The error on the trend.

stdDevSeries The slowly varying standard deviation series.

stdDevSeriesNonSeasonal The non-seasonal slowly varying standard deviation series.

stdDevError The error on the standard deviation.

timeStamps The time stamps.

nonStatSeries The original non-stationary series.

statSer3Mom The third moment of the transformed stationary series.

statSer4Mom The fourth moment of the transformed stationary series.

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
timeWindow <- 30*365 # 30 years
#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs>=2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
result <- tsEvaTransformSeriesToStationaryTrendOnly(timeStamps, series, timeWindow)
plot(result$trendSeries)
```

tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile

tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile

Description

tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile transforms a time series to a stationary ones using a moving average as the trend and a running percentiles to represent the slowly varying amplitude of the distribution

Usage

```
tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile(
  timeStamps,
  series,
  timeWindow,
  percentile
)
```

Arguments

timeStamps	A vector of time stamps for the time series.
series	The original time series.
timeWindow	The size of the moving window used for detrending.
percentile	The percentile value used to compute the extreme trend of the stationary series.

Value

A list containing the following elements:

runningStatsMultiplicity The running statistics multiplicity
stationarySeries The transformed stationary trend only series
trendSeries The trend series
trendSeriesNonSeasonal The non-seasonal trend series
trendError The error on the trend
stdDevSeries The standard deviation series
stdDevSeriesNonSeasonal The non-seasonal standard deviation series
stdDevError The error on the standard deviation
timeStamps The time stamps
nonStatSeries The original non-stationary series
statSer3Mom The running mean of the third moment of the stationary series
statSer4Mom The running mean of the fourth moment of the stationary series

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs>=2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
percentile <- 90
result <- tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile(timeStamps,
series, timeWindow, percentile)
plot(result$trendSeries)
```

```
tsEvaTransformSeriesToStatSeasonal_ciPercentile
    tsEvaTransformSeriesToStatSeasonal_ciPercentile
```

Description

This function decomposes a time series into a season-dependent trend and a season-dependent standard deviation. The season-dependent amplitude is given by a seasonal factor multiplied by a slowly varying percentile.

Usage

```
tsEvaTransformSeriesToStatSeasonal_ciPercentile(
    timeStamps,
    series,
    timeWindow,
    percentile
)
```

Arguments

timeStamps	A vector of time stamps for the time series.
series	The original time series.
timeWindow	The length of the moving window used for trend estimation.
percentile	The percentile value used for computing the slowly varying percentile.

Value

A list containing the following components:

runningStatsMultiplicity	The size of each sample used to compute the average
stationarySeries	The transformed stationary series
trendSeries	The trend series
trendSeriesNonSeasonal	The non-seasonal trend series
stdDevSeries	The season-dependent standard deviation series
stdDevSeriesNonSeasonal	The non-seasonal standard deviation series
trendError	The error on the trend
stdDevError	The error on the standard deviation
statSer3Mom	The 3rd moment of the transformed stationary series
statSer4Mom	The 4th moment of the transformed stationary series
nonStatSeries	The original non-stationary series
Regime	The regime of the trend seasonality
timeStamps	The time stamps

trendNonSeasonalError The error on the non-seasonal trend

stdDevNonSeasonalError The error on the non-seasonal standard deviation

trendSeasonalError The error on the seasonal trend

stdDevSeasonalError The error on the seasonal standard deviation

this function decomposes the series into a season-dependent trend and a

season-dependent standard deviation. The season-dependent standard deviation is given by a seasonal factor multiplied by a slowly varying standard deviation. transformation non stationary -> stationary $x(t) = (y(t) - \text{trend}(t) - \text{ssn_trend}(t)) / (\text{stdDev}(t) * \text{ssn_stdDev}(t))$ transformation stationary -> non stationary $y(t) = \text{stdDev}(t) * \text{ssn_stdDev}(t) * x(t) + \text{trend}(t) + \text{ssn_trend}(t)$ `trasfData.trendSeries = trend(t) + ssn_trend(t)` `trasfData.stdDevSeries = stdDev(t) * ssn_stdDev(t)`

Examples

```
timeAndSeries <- ArdecheStMartin
timeStamps <- ArdecheStMartin[,1]
series <- ArdecheStMartin[,2]
#select only the 5 latest years
yrs <- as.integer(format(timeStamps, "%Y"))
tokeep <- which(yrs >= 2015)
timeStamps <- timeStamps[tokeep]
series <- series[tokeep]
timeWindow <- 365 # 1 year
percentile <- 90
result <- tsEvaTransformSeriesToStatSeasonal_ciPercentile(timeStamps,
series, timeWindow, percentile)
plot(result$trendSeries)
```

tsEVstatistics

tsEVstatistics

Description

tsEvstatistics is a function that calculates the Generalized Extreme Value (GEV) and Generalized Pareto Distribution (GPD) statistics and return levels for a given dataset of extreme values.

Usage

```
tsEVstatistics(
  pointData,
  alphaCI = 0.95,
  gevMaxima = "annual",
  gevType = "GEV",
  evdType = c("GEV", "GPD"),
  shape_bnd = c(-0.5, 1)
)
```

Arguments

pointData	A list containing the dataset of extreme values. It should include the following components: annualMax A vector of annual maximum values annualMaxDate A vector of dates corresponding to the annual maximum values monthlyMax A matrix of monthly maximum values
alphaCI	The confidence level for the confidence intervals of the parameter estimates. Default is 0.95.
gevMaxima	The type of maxima to use for GEV fitting. Can be either 'annual' or 'monthly'. Default is 'annual'.
gevType	The type of GEV distribution to use. Can be either 'GEV', 'Gumbel'. Default is 'GEV'.
evdType	The types of extreme value distributions to calculate. Can be a combination of 'GEV' and 'GPD'. Default is c('GEV', 'GPD').
shape_bnd	The lower and upper bounds for the shape parameter of the GEV distribution. Default is c(-0.5, 1).

Value

A list containing the following components:

EVmeta A list containing metadata about the analysis. It includes Tr, A vector of return periods for which return levels are calculated

EVdata A list containing the calculated statistics and return levels. It includes the following components:

 GEVstat A list containing the GEV statistics and return levels:

 method The method used for fitting the GEV distribution.

 values A vector of return levels calculated using the GEV distribution.

 parameters A vector of parameter estimates for the GEV distribution.

 paramCIs A matrix of confidence intervals for the parameter estimates.

 GPDstat list containing the GPD statistics and return levels:

 method The method used for fitting the GPD distribution

 values A vector of return levels calculated using the GPD distribution

 parameters A vector of parameter estimates for the GPD distribution

 paramCIs A matrix of confidence intervals for the parameter estimates

isValid A logical value indicating whether the analysis was performed or not.

Examples

```
# Create a sample dataset
data <- ArdecheStMartin
colnames(data) <- c("Date", "Value")
yrs <- as.integer(format(data$Date, "%Y"))
tokeep <- which(yrs >= 2015)
```

```

data <- data[tokeep,]
pointData <- tsEvaSampleData(data, meanEventsPerYear=3, minEventsPerYear=0,
minPeakDistanceInDays=7, "high")
result <- tsEVstatistics(pointData)
result$EVdata$GEVstat$values
result$EVdata$GPDstat$values

```

<code>tsGetNumberPerYear</code>	<i>tsGetNumberPerYear</i>
---------------------------------	---------------------------

Description

`tsGetNumberPerYear` is a function that calculates the number of events per year based on a given time series and a set of locations.

Usage

```
tsGetNumberPerYear(ms, locs)
```

Arguments

<code>ms</code>	A data frame representing the time series data, where the first column contains the dates of the events.
<code>locs</code>	A vector of indices representing the locations of interest in the time series.

Value

A data frame with two columns: "year" and "Freq". The "year" column contains the years, and the "Freq" column contains the number of events per year.

Examples

```

# Create a sample time series data frame
set.seed(123)
ms <- data.frame(date = seq(as.Date("2000-01-01"), as.Date("2022-12-31"), by = "day"),
                 values=rnorm(8401))
# Generate random events
events <- match(sample(ms$date, 100), ms$date)
# Get the number of events per year
tsGetNumberPerYear(ms, events)

```

tsGetPOT

*tsGetPOT Function***Description**

tsGetPOT is a function that calculates the Peaks Over Threshold (POT) for a given time series data.

Usage

```
tsGetPOT(
  ms,
  pcts,
  desiredEventsPerYear,
  minEventsPerYear,
  minPeakDistanceInDays,
  tail
)
```

Arguments

ms	A matrix containing the time series data with two columns: the first column represents the time and the second column represents the values.
pcts	A numeric vector specifying the percentiles to be used as thresholds for identifying peaks.
desiredEventsPerYear	The desired number of events per year.
minEventsPerYear	The minimum number of events per year.
minPeakDistanceInDays	The minimum distance between two peaks in days.
tail	The tail to be studied for POT selection, either 'high' or 'low'.

Value

A list containing the following fields:

threshold	The threshold value used for identifying peaks
thresholdError	The error associated with the threshold value
percentile	The percentile value used as the threshold.
peaks	The values of the identified peaks.
stpeaks	The start indices of the identified peaks.
endpeaks	The end indices of the identified peaks.
ipeaks	The indices of the identified peaks.
time	The time values corresponding to the identified peaks.
pars	The parameters of the Generalized Pareto Distribution (GPD) fitted to the peaks.

See Also

[tsEvaSampleData\(\)](#)

Examples

```
# Create a sample time series data
ms <- ArdecheStMartin

# Calculate the POT using the tsGetPOT function
pcts <- c(90, 95, 99)
desiredEventsPerYear <- 5
minEventsPerYear <- 2
minPeakDistanceInDays <- 10
tail <- "high"
POTdata <- tsGetPOT(ms, pcts, desiredEventsPerYear, minEventsPerYear, minPeakDistanceInDays, tail)
# Print the results
print(POTdata)
```

Index

* datasets

- ArdecheStMartin, [3](#)
 - DanubeVienna, [6](#)
 - EbroZaragoza, [7](#)
 - RhoneLyon, [12](#)
- ArdecheStMartin, [3](#)
- check_timeseries, [4](#)
- computeAnnualMaxima, [4](#)
- computeMonthlyMaxima, [5](#)
- DanubeVienna, [6](#)
- declustpeaks, [6](#)
- EbroZaragoza, [7](#)
- empdis, [8](#), [16](#), [21](#), [22](#)
- empdisl, [9](#)
- findMax, [10](#)
- initPercentiles, [10](#)
- max_daily_value, [11](#)
- RhoneLyon, [12](#)
- tsEasyParseNamedArgs, [12](#)
- tsEstimateAverageSeasonality, [13](#)
- tsEvaChangepts, [14](#)
- tsEvaComputeReturnLevelsGEV, [15](#), [17](#), [23](#),
[34](#), [42](#)
- tsEvaComputeReturnLevelsGEVFromAnalysisObj,
[16](#)
- tsEvaComputeReturnLevelsGPD, [18](#), [20](#), [23](#),
[36](#), [46](#)
- tsEvaComputeReturnLevelsGPDFFromAnalysisObj,
[19](#)
- tsEvaComputeReturnPeriodsGEV, [20](#)
- tsEvaComputeReturnPeriodsGPD, [21](#)
- tsEvaComputeRLsGEVGPD, [23](#)
- tsEvaComputeTimeRP, [24](#)
- tsEvaDetrendTimeSeries, [25](#)
- tsEvaDetrendTimeSeries(), [54](#)
- tsEvaFillSeries, [26](#)
- tsEvaFindTrendThreshold, [26](#)
- tsEvaFindTrendThreshold(), [57](#)
- tsEvaNanRunnigBlowTh, [27](#)
- tsEvaNanRunningMean, [28](#)
- tsEvaNanRunningMean(), [54](#)
- tsEvaNanRunningPercentiles, [29](#)
- tsEvaNanRunningPercentiles(), [11](#)
- tsEvaNanRunningStatistics, [30](#)
- tsEvaNanRunningStatistics(), [54](#)
- tsEvaNanRunningVariance, [31](#)
- tsEvaNanRunningVariance(), [54](#)
- TsEvaNs, [31](#)
- tsEvaPlotAllRLevelsGEV, [33](#)
- tsEvaPlotAllRLevelsGEV(), [44](#)
- tsEvaPlotAllRLevelsGPD, [35](#)
- tsEvaPlotAllRLevelsGPD(), [47](#)
- tsEvaPlotGEVImageSc, [37](#), [38](#)
- tsEvaPlotGEVImageScFromAnalysisObj, [38](#),
[38](#)
- tsEvaPlotGPDImageSc, [39](#), [41](#)
- tsEvaPlotGPDImageScFromAnalysisObj, [40](#),
[40](#)
- tsEvaPlotReturnLevelsGEV, [42](#)
- tsEvaPlotReturnLevelsGEV(), [44](#)
- tsEvaPlotReturnLevelsGEVFromAnalysisObj,
[42](#), [43](#)
- tsEvaPlotReturnLevelsGPD, [45](#)
- tsEvaPlotReturnLevelsGPD(), [47](#)
- tsEvaPlotReturnLevelsGPDFFromAnalysisObj,
[46](#), [46](#)
- tsEvaPlotSeriesTrendStdDevFromAnalysisObj,
[48](#)
- tsEvaPlotTransfToStat, [49](#), [50](#)
- tsEvaPlotTransfToStatFromAnalysisObj,
[50](#), [50](#)

tsEvaRunningMeanTrend, [51](#)
tsEvaSampleData, [52](#)
tsEvaSampleData(), [68](#)
tsEvaTransformSeriesToStationaryMMXTrend,
[53](#)
tsEvaTransformSeriesToStationaryMultiplicativeSeasonality,
[54](#)
tsEvaTransformSeriesToStationaryPeakTrend,
[56](#)
tsEvaTransformSeriesToStationaryTrendAndChangepts,
[57](#)
tsEvaTransformSeriesToStationaryTrendAndChangepts_ciPercentile,
[59](#)
tsEvaTransformSeriesToStationaryTrendOnly,
[60](#)
tsEvaTransformSeriesToStationaryTrendOnly_ciPercentile,
[61](#)
tsEvaTransformSeriesToStatSeasonal_ciPercentile,
[63](#)
tsEVstatistics, [64](#)
tsGetNumberPerYear, [66](#)
tsGetPOT, [67](#)
tsGetPOT(), [53](#)