Package 'ShiftHappens'

July 29, 2026

Type Package

Title Detecting, Visualizing and Estimating Shifts

Version 1.0.1

Description Detecting, visualizing and estimating shifts,
with a specific focus on stage-discharge rating shifts.
The main methods are described in Darienzo et al. (2021) <doi:10.1029/2020WR028607>
and Mansanarez et al. (2019) <doi:10.1029/2018WR023389>.
See also 'BayDERS' <https://github.com/MatteoDarienzo/BayDERS> for similar tools,
and 'RatingShiftHappens' <https://github.com/Felipemendezrios/RatingShiftHappens> for an older version of this package.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/benRenard/ShiftHappens

RoxygenNote 7.3.3

Imports lubridate, RBaM, ggplot2, scales, dplyr, tidyr, patchwork,
RColorBrewer

Suggests knitr, rmarkdown

Depends R (>= 3.5.0)

NeedsCompilation no

Author Felipe Mendez [aut] (ORCID: <https://orcid.org/0009-0004-0371-8699>),
Benjamin Renard [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-8447-5430>),
Matteo Darienzo [aut],
INRAE [fnd],
Ministère de la Transition Ecologique - SCHAPI [fnd]

Maintainer Benjamin Renard <benjamin.renard@inrae.fr>

Repository CRAN

Date/Publication 2026-07-29 16:40:45 UTC

Contents

`ArdecheRiverGaugings` *Gauging of the Ardèche River at Meyras, France, provided by UHPC Grand Delta*

Description

A data frame containing stages (H, in meters) and discharges measurements (Q, in cubic meters per second) for the Ardèche River at Meyras, France, along with the associated uncertainties expressed as standard deviations (uQ) from 2001 until 2018

Usage

```
ArdecheRiverGaugings
```

Format

Day Day

Month Month

Year Year

Hour Hour

Minute Minute

Second Second

Date Date

H Stage measurement

Q Discharge measurement

uQ Discharge uncertainty expressed as a standard deviation

Source

doi:10.1029/2018WR023389

ArdecheRiverStage *Stage record of the Ardèche River at Meyras, France, provided by UHPC Grand Delta*

Description

A data frame containing stages (H, in meters) for the Ardèche River at Meyras, France, from 07/11/2001 until 29/10/2018

Usage

```
ArdecheRiverStage
```

Format

Date Date, in POSIXct format

H Stage record

Source

https://theses.fr/2021GRALU006

extractedRecessions *extractedRecessions object constructor.*

Description

Creates a new instance of a 'extractedRecessions' object, used to store the results of a recession extraction algorithm.

Usage

```
extractedRecessions(
  date = as.Date(numeric(0)),
  time = numeric(0),
  H = numeric(0),
  uH = numeric(0),
  index = integer(0)
)
```

Arguments

date	Date vector, date.
time	numeric vector, within-recession time (in days), i.e. time is reset to zero at the beginning of each recession.
H	numeric vector, stage.
uH	numeric vector, stage uncertainty expressed as a standard deviation.
index	integer vector, recession index.

Value

An object of class `extractedRecessions()`: data frame containing the input vectors as columns.

Examples

```
rec <- extractedRecessions()
```

Extract_Recessions *Recessions extractor*

Description

Extract recessions from a stage record. NAs are not allowed.

Usage

```
Extract_Recessions(
  time,
  H,
  uH = 0 * H,
  deltahMax = stats::quantile(H, probs = 0.95, na.rm = TRUE),
  deltatMax = 20,
  dMin = 10,
  nMin = 10,
  burn = 0,
  deltatMin = 0,
  nSlim = 1
)
```

Arguments

time	POSIXct vector, time. The POSIXct format is mandatory.
H	real vector , stage (m).
uH	real vector, stage uncertainty, expressed as a standard deviation (m).
deltahMax	real value, maximum stage rise allowed within a recession (m). If the stage record rises by more than deltahMax m, the current recession is stopped and a new one is started.
deltatMax	real value, maximum time between two stage values allowed within a recession (in days). When two successive values are separated by more than deltatMax days, the current recession is stopped and a new one is started. Useful for dealing with periods of missing data.
dMin	real value, minimum duration (in days) for a recession to be retained in the output dataset.
nMin	integer value, minimum number of data points for a recession to be retained in the output dataset.
burn	numeric value >=0 and <1, burn factor. burn=0.4 means that the first 40 percents of the recession points are discarded.
deltatMin	real value, minimum time between two successive values (in days). When successive values are too close (less than deltatMin apart), the algorithm will jump to the next value for speed-up purposes (and hence ignore a stage value). Useful for quick preliminary runs, but deltatMin=0 is recommended for definitive runs.
nSlim	integer value, initial slimming of the stage record (for speed-up purposes). nSlim=10 means that only one stage value every 10 is kept before applying the extraction algorithm. Useful for quick preliminary runs, but nSlim=1 is recommended for definitive runs.

Value

An object of class `extractedRecessions()`, which is a data frame with the following columns:

1. date: date

2. time: within-recession time (in days), i.e. time is reset to zero at the beginning of each recession
3. H: stage
4. uH: stage uncertainty (expressed as a standard deviation)
5. index: index of the recession event

Examples

```
rec=Extract_Recessions(time=ArdecheRiverStage$Date,H=ArdecheRiverStage$H,
                       nSlim=10) # used to speed-up example, but nSlim=1 is recommended.
plot(ArdecheRiverStage$Date,ArdecheRiverStage$H,type='l',col='lightgray')
points(rec$date,rec$H,col=rec$index)
plot(rec$time,rec$H,col=rec$index)
```

`fittedModel` *fittedModel object constructor.*

Description

Creates a new instance of a 'fittedModel' object, used to store the results of a fitted model

Usage

```
fittedModel(
  time = numeric(0),
  x = numeric(0),
  y = numeric(0),
  ysim = numeric(0),
  res = y - ysim,
  uX = 0 * x,
  uY = 0 * y,
  uYsim = 0 * ysim,
  uRes = sqrt(uY^2 + uYsim^2),
  group = rep(1, length(time)),
  parameters = numeric(0),
  uParameters = NA * parameters
)
```

Arguments

`time`	numeric vector, time
`x`	numeric vector or matrix, observed inputs (predictors)
`y`	numeric vector, observed output (predictand)
`ysim`	numeric vector, simulated output
`res`	numeric vector, residual

uX	numeric vector or matrix, uncertainty in observed inputs (expressed as a standard deviation)
uY	numeric vector, uncertainty in observed outputs (expressed as a standard deviation)
uYsim	numeric vector, uncertainty in simulated outputs (expressed as a standard deviation)
uRes	numeric vector, residual uncertainty (expressed as a standard deviation)
group	integer vector, group
parameters	numeric vector, fitted parameters
uParameters	numeric vector, standard deviation or standard error of fitted parameters

Value

An object of class `fittedModel()`, containing the following fields:

1. data: data frame, column-binding all the arguments of the function except parameters
2. parameters: data frame, fitted parameters + standard error

Examples

```
f <- fittedModel()
```

`Fit_BaRatin` *BaRatin*

Description

Fit a rating curve with BaRatin

Usage

```
Fit_BaRatin(
  x,
  y,
  time = 1:NROW(y),
  uX = 0 * x,
  uY = 0 * y,
  flavor = c("BaRatin", "BaRatinBAC"),
  controlMatrix = matrix(1),
  priors = get3FlatPriors(x, y),
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  remnant = list(RBaM::remnantErrorModel()),
  temp.folder = file.path(tempdir(), "BaRatin")
)
```

Arguments

x	real vector, stage
y	real vector, discharge
time	vector (numeric or date), time
uX	real vector, stage uncertainty (as a standard deviation)
uY	real vector, discharge uncertainty (as a standard deviation)
flavor	string, BaRatin flavor
controlMatrix	square matrix, control matrix
priors	list, list of RBaM::parameter objects containing information for each parameter of the rating curve. Length 3*NROW(controlMatrix)
mcmc_options	mcmcOptions object, see ?RBaM::mcmcOptions
mcmc_cooking	mcmcCooking object, see ?RBaM::mcmcCooking
remnant	list of one remnantErrorModel object, see ?RBaM::remnantErrorModel
temp.folder	string, directory where config and result files are stored.

Value

An object of class `fittedModel()`, containing the following fields:

1. data: data frame, column-binding all the arguments of the function except parameters
2. parameters: data frame, fitted parameters + standard error

Examples

```
f=Fit_BaRatin(x=ArdecheRiverGaugings$H,y=ArdecheRiverGaugings$Q,
              uY=ArdecheRiverGaugings$uQ,time=ArdecheRiverGaugings$Date,
              # MCMC options are modified to speed-up example.
              # Using default MCMC options is safer and is recommended.
              mcmc_options=mcmcOptions(nAdapt=50,nCycles=20))
if(!is.null(f)){
  plot(f$data$x,f$data$y);points(f$data$x,f$data$ysim,col='red')
  plot(f$data$time,f$data$res)
}
```

Fit_LinearRegression *Linear regression*

Description

Fit a linear regression (with possibly several predictors but a single predictand)

Usage

```
Fit_LinearRegression(x, y, time = 1:NROW(y), uX = 0 * x, uY = 0 * y)
```

Arguments

x	real matrix or data frame, predictors
y	real vector, predictand
time	vector (numeric or date), time
uX	real matrix or data frame, uncertainty in predictors (ignored)
uY	real vector, uncertainty in predictand (ignored)

Value

An object of class fittedModel(), containing the following fields:

1. data: data frame, column-binding all the arguments of the function except parameters
2. parameters: data frame, fitted parameters + standard error

Examples

```
f=Fit_LinearRegression(x=RhoneRiverAMAX$H,y=RhoneRiverAMAX$Q,time=RhoneRiverAMAX$Date)
plot(f$data$x1,f$data$y);lines(f$data$x1,f$data$ysim,col='red')
plot(f$data$time,f$data$res)
```

Fit_Recessions *Fit recession model*

Description

Fit a model to a set of recession events.

Usage

```
Fit_Recessions(
  rec,
  equation = c("M6", "M1", "M2", "M3", "M4", "M5", "M7", "M8", "M9"),
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  remnant = list(getConstantRemnant(rec)),
  temp.folder = file.path(tempdir(), "Recessions")
)
```

Arguments

rec	extractedRecessions() object, resulting from a call to function Extract_Recessions()
equation	string, equation used to model recessions, from M1 to M9. For more details see Chapter 3 of the PhD thesis of Matteo Darienzo (2021, https://theses.hal.science/tel-03211343) or call getRecessionEquations()
mcmc_options	mcmcOptions object, see ?RBaM::mcmcOptions
mcmc_cooking	mcmcCooking object, see ?RBaM::mcmcCooking
remnant	list of one remnantErrorModel object, see ?RBaM::remnantErrorModel
temp.folder	string, directory where config and result files are stored.

Value

An object of class `fittedModel()`, containing the following fields:

1. data: data frame, column-binding all the arguments of the function except parameters
2. parameters: data frame, fitted parameters + standard error

Source

https://theses.hal.science/tel-03211343

Examples

```
rec=Extract_Recessions(time=ArdecheRiverStage$Date,H=ArdecheRiverStage$H,
                       nSlim=10) # used to speed-up example, but nSlim=1 is recommended.
f=Fit_Recessions(rec=rec,equation='M7',
                 # MCMC options are modified to speed-up example.
                 # Using default MCMC options is safer and is recommended.
                 mcmc_options=mcmcOptions(nAdapt=20,nCycles=10))
if(!is.null(f)){
  plot(f)
  plot(f,type='ty')
}
```

---

getRecessionEquations *Recessions equations*

Description

Get all available recession equations. For details see Chapter 3 of the PhD thesis of Matteo Darienzo (2021, https://theses.hal.science/tel-03211343).

Usage

```
getRecessionEquations()
```

Value

A list containing all available equations.

Source

https://theses.hal.science/tel-03211343

Examples

```
getRecessionEquations()
```

getRecessionMin *Recessions minima*

Description

Get recessions lowest point from a `extractedRecessions()` object.

Usage

```
getRecessionMin(x)
```

Arguments

x	object of class extractedRecessions, resulting from a call to function `Extract_Recessions()`

Value

A data frame containing the minimum value of each recession, along with the corresponding dates and uncertainties.

Examples

```
rec=Extract_Recessions(time=ArdecheRiverStage$Date,H=ArdecheRiverStage$H,
                       nSlim=10) # used to speed-up example, but nSlim=1 is recommended.
lows=getRecessionMin(rec)
plot(lows$date,lows$H)
```

getShifts *Get Shifts*

Description

Extract shifts (MCMC samples) from a of class simpleSegmentation, multipleSegmentation or recursiveSegmentation

Usage

```
getShifts(x, castAsDate = TRUE)
```

Arguments

x	object, of class simpleSegmentation, multipleSegmentation or recursiveSegmentation.
castAsDate	boolean. The raw MCMC simulations use an internal numeric representation of time. If castAsDate is TRUE, they will be transformed into the time/date format that was used in the segmented dataset.

Value

a dataframe containing the MCMC samples of detected shift times

Examples

```
x <- Segmentation(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Date,u=RhoneRiverAMAX$uH)
shifts=getShifts(x)
summary(shifts)
```

`multipleSegmentation` *multipleSegmentation object constructor.*

Description

Creates a new instance of a 'multipleSegmentation' object, used to store the results of a segmentation with an unknown number of segment.

Usage

```
multipleSegmentation(results)
```

Arguments

`results`	list, vector of 'simpleSegmentation' objects, segmentation results for each tested number of segments

Value

An object of class `multipleSegmentation()`, containing the following fields:

1. nS: integer, optimal number of segments (minimum DIC)
2. DICs: real vector, DICs computed for each number of segment
3. data: data frame, all data with their respective periods after segmentation (for optimal nS)
4. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC (for optimal nS)
5. mcmc: data frame, MCMC simulations (for optimal nS)
6. DIC: real, DIC estimation (for optimal nS)
7. origin.date: positive real or date, date describing origin of the segmentation for a sample. Useful for recursive segmentation.
8. results: list, intermediate results for all tested number of segments see ?Segmentation_Engine

Examples

```
results=list()
results[[1]]=Segmentation_Engine(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,
                                 u=RhoneRiverAMAX$uH,nS=1)
results[[2]]=Segmentation_Engine(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,
                                 u=RhoneRiverAMAX$uH,nS=2)
sg <- multipleSegmentation(results)
```

---

numeric_to_time *Numeric to time format*

Description

Cast a numeric interpreted as the duration after an origin date into a time format.

Usage

```
numeric_to_time(d, origin.date)
```

Arguments

d	real vector, duration (in days) after origin.date
origin.date	value, origin date in character, POSIXct or Date format. If numeric, the untransformed vector d will be returned.

Value

the date d days after origin.date, in the same format as the latter

Examples

```
numeric_to_time(d=1,origin.date='1993-05-26')
numeric_to_time(d=366,origin.date='1993-05-26')
numeric_to_time(d=0.5,origin.date='1993-05-26')
```

---

plot.extractedRecessions *extractedRecessions plotter*

Description

Plot a `extractedRecessions()` object.

Usage

```
## S3 method for class 'extractedRecessions'
plot(x, type = c("dh", "th", "dhmin"), showAnnotation = FALSE, ...)
```

Arguments

x	object of class extractedRecessions, resulting from a call to function Extract_Recessions()
type	string, type of data plot: 'dh' for date (x) vs. stage (y), 'th' for time (x) vs. stage (y), 'dhmin' for date (x) vs. minimum stage of the recession (y).
showAnnotation	logical, show annotations on plot?
...	Optional arguments

Value

A ggplot.

Examples

```
rec=Extract_Recessions(time=ArdecheRiverStage$Date,H=ArdecheRiverStage$H,
                       nSlim=10) # used to speed-up example, but nSlim=1 is recommended.
plot(rec)
```

---

plot.fittedModel *fittedModel Plotter.*

Description

Plot a fittedModel() object.

Usage

```
## S3 method for class 'fittedModel'
plot(x, type = c("xy", "ty", "tres", "yysim"), ...)
```

Arguments

x	object of class fittedModel
type	string, type of plot: 'xy' for a x-y scatterplot, 'ty' for a time series plot of y and ysim, 'tres' for a time series plot of residuals, 'yysim' for a scatterplot y vs. ysim
...	Optional arguments

Value

a ggplot

Examples

```
f=Fit_LinearRegression(x=RhoneRiverAMAX$H,y=RhoneRiverAMAX$Q,time=RhoneRiverAMAX$Date)
plot(f)
plot(f,type='ty')
plot(f,type='tres')
plot(f,type='yysim')
```

plot.multipleSegmentation

multipleSegmentation plotter

Description

Plot a multipleSegmentation object.

Usage

```
## S3 method for class 'multipleSegmentation'
plot(x, type = c("both", "data", "shifts"), ...)
```

Arguments

x	object of class multipleSegmentation
type	string, type of plot
...	Optional arguments

Value

a patchwork object if type='both', a ggplot object if type='data' or type='shifts'.

Examples

```
x=Segmentation(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH)
plot(x)
```

plot.recursiveModeling

recursiveModeling plotter

Description

Plot a `recursiveModeling()` object.

Usage

```
## S3 method for class 'recursiveModeling'
plot(
  x,
  type = c("both", "data", "shifts", "fits"),
  dataPlotType = c("xy", "ty", "tx"),
  ...
)
```

Arguments

x	object of class recursiveModeling
type	string, type of plot: 'both' for a data panel on top and a shift panel on the bottom, 'data' for the data panel only, 'shifts' for the shift panel only, 'fits' for a plot of fitted models in each terminal mode.
dataPlotType	string, type of data plot: 'xy' for a x-y scatterplot, 'ty' for a time series plot of y, 'tx' for a time series plot of x
...	Optional arguments

Value

a patchwork object if type='both', a ggplot object if type='data' or type='shifts', a list of ggplot objects if type='fits'.

Examples

```
x=Segmentation_RecursiveModeling(x=ArdecheRiverGaugings$H,y=ArdecheRiverGaugings$Q)
plot(x)
```

---

plot.recursiveSegmentation

recursiveSegmentation plotter

Description

Plot a recursiveSegmentation object.

Usage

```
## S3 method for class 'recursiveSegmentation'
plot(x, type = c("both", "data", "shifts"), ...)
```

Arguments

x	object of class recursiveSegmentation
type	string, type of plot
...	Optional arguments

Value

a patchwork object if type='both', a ggplot object if type='data' or type='shifts'.

Examples

```
x=Segmentation_Recursive(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH)
plot(x)
```

plot.segmentationTree *segmentationTree Plotter*

Description

Plot a `segmentationTree()` object, resulting from a recursive segmentation procedure applied with function `Segmentation_Recursive()`.

Usage

```
## S3 method for class 'segmentationTree'
plot(x, ...)
```

Arguments

x	`segmentationTree()` object, tree resulting from a call to function `Segmentation_Recursive()`
...	Optional arguments

Value

a ggplot

Examples

```
res=Segmentation_Recursive(obs=RhoneRiverAMAX$H)
plot(res$tree)
```

plot.simpleSegmentation *simpleSegmentation plotter*

Description

Plot a simpleSegmentation object.

Usage

```
## S3 method for class 'simpleSegmentation'
plot(x, type = c("both", "data", "shifts"), ...)
```

Arguments

x	object of class simpleSegmentation
type	string, type of plot
...	Optional arguments

Value

a patchwork object if type='both', a ggplot object if type='data' or type='shifts'.

Examples

```
x=Segmentation_Engine(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH)
plot(x)
```

---

recursiveModeling *recursiveModeling object constructor.*

Description

Creates a new instance of a 'recursiveModeling' object, used to store the results of a recursive modeling segmentation

Usage

```
recursiveModeling(
  data = data.frame(),
  segmentation = recursiveSegmentation(),
  fit = list(fittedModel())
)
```

Arguments

data	data frame, initial dataset
segmentation	recursiveSegmentation() object, segmentation results
fit	list of fittedModel() objects, fitted model results

Value

An object of class recursiveModeling(), containing the following fields:

1. data: a data frame containing the original dataset used as input of the recursive modeling
2. segmentation: an object of class recursiveSegmentation(), containing the results of the segmentation of residuals
3. fit: a list of objects of class fittedModel(), containing the results of the model fit at each stage of the recursion

Examples

```
rec <- recursiveModeling()
```

`recursiveSegmentation` *recursiveSegmentation object constructor.*

Description

Creates a new instance of a 'recursiveSegmentation' object, used to store the results of a recursive segmentation with an unknown number of segment.

Usage

```
recursiveSegmentation(
  data = data.frame(),
  shifts = data.frame(),
  tree = segmentationTree(),
  results = list()
)
```

Arguments

`data`	data frame, data summary.
`shifts`	data frame, detected shifts (with columns tau, I95_lower and I95_upper)
`tree`	`segmentationTree()` object, recursion tree.
`results`	list, vector of 'multipleSegmentation' objects, segmentation results at each step of the recursion

Value

An object of class `recursiveSegmentation()`, containing the following fields:

1. nS: integer, final number of segments
2. data: data frame, all data with their respective periods after segmentation
3. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC
4. tree: `segmentationTree()` object, recursion tree.
5. results: list, intermediate results at each stage of the recursion.

Examples

```
sg <- recursiveSegmentation()
```

RhoneRiverAMAX *Floods of the Rhone River at Beaucaire, France*

Description

A data frame containing annual maximum stages (H, in meters) and discharges (Q, in cubic meters per second) for the Rhone River at Beaucaire, France, along with the associated uncertainties expressed as standard deviations (uH and uQ). Details on the reconstruction of these long series can be found in the article by Lucas et al. (2023) referenced below. Note that years 1968, 1969 and 1970 are missing and are not included in the data frame.

Usage

```
RhoneRiverAMAX
```

Format

Year Year

Date Date

Time Time

H Stage record

uH Uncertainty of the stage expressed as standard deviation

Q Discharge

uQ Uncertainty of the discharge expressed as standard deviation

Source

doi:10.1016/j.jhydrol.2023.129840

Segmentation *Segmentation*

Description

Segmentation procedure for an **unknown** number of segments

Usage

```
Segmentation(
  obs,
  time = 1:length(obs),
  u = 0 * obs,
  nSmax = 2,
  doQuickApprox = (nSmax < 3),
  nMin = ifelse(doQuickApprox, 3, 1),
  nSim = 500,
  varShift = FALSE,
  alpha = 0.1,
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  temp.folder = file.path(tempdir(), "BaM"),
  mu_prior = list()
)
```

Arguments

obs	real vector, observations
time	vector, time in POSIXct, string or numeric format
u	real vector, uncertainty in observations (as a standard deviation)
nSmax	integer, maximum number of segments to assess
doQuickApprox	logical, use quick approximation? see ?Segmentation_quickApprox
nMin	integer, minimum number of observations by segment
nSim	integer, number of Monte-Carlo simulated values for shift times. Only used when doQuickApprox=TRUE.
varShift	logical, allow for a shifting variance? Only used when doQuickApprox=TRUE.
alpha	real in (0;1), type-I error level of the underlying step-change test. Only used when doQuickApprox=TRUE.
mcmc_options	mcmcOptions object, options used to generate MCMC samples (e.g. size and adaption tunings), see ?RBaM::mcmcOptions. Only used when doQuickApprox=FALSE.
mcmc_cooking	mcmcCooking object, options used to post-process MCMC samples (e.g. burn and slice), see ?RBaM::mcmcCooking. Only used when doQuickApprox=FALSE.
temp.folder	directory, temporary directory to write BaM computations. Only used when doQuickApprox=FALSE.
mu_prior	list, object describing prior knowledge on the mean of residuals for each segment. Only used when doQuickApprox=FALSE.

Value

An object of class `multipleSegmentation()`, containing the following fields:

1. nS: integer, optimal number of segments (minimum DIC)

2. DICs: real vector, DICs computed for each number of segment
3. data: data frame, all data with their respective periods after segmentation (for optimal nS)
4. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC (for optimal nS)
5. mcmc: data frame, MCMC simulations (for optimal nS)
6. DIC: real, DIC estimation (for optimal nS)
7. origin.date: positive real or date, date describing origin of the segmentation for a sample. Useful for recursive segmentation.
8. results: list, intermediate results for all tested number of segments see ?Segmentation_Engine

Examples

```
res=Segmentation(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Date,u=RhoneRiverAMAX$uH)
res$shifts
res$DICs
hist(res$mcmc$tau)
# Transform tau values from numeric to date/time format
tau_date=numeric_to_time(res$mcmc$tau,origin.date=res$origin.date)
plot(tau_date)
# Segmentation using BaM, allowing more than 2 segments
## Not run:
# This line of code is wrapped in \dontrun{} since it relies
# on the installation of an executable with the package RBaM.
# See ?RBaM::downloadBaM for downloading and installing this executable.
res=Segmentation(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH,
     doQuickApprox=FALSE,nSmax=3)

## End(Not run)
```

---

segmentationTree *segmentationTree object constructor.*

Description

Creates a new instance of a 'segmentationTree' object, used to store the tree of a recursive segmentation.

Usage

```
segmentationTree(
  index = integer(0),
  level = integer(0),
  parent = integer(0),
  nS = integer(0)
)
```

Arguments

index	integer vector, node index.
level	integer vector, level of the recursive segmentation.
parent	integer vector, parent of the current node.
nS	integer vector, detected number of segments.

Value

An object of class segmentationTree(): data frame containing the input vectors as columns.

Examples

```
rec <- segmentationTree()
```

Segmentation_BaRatin *BaRatin-based segmentation*

Description

Segmentation based on gaugings and a recursive estimation of BaRatin rating curves. This function is just a wrapper around Segmentation_RecursiveModeling().

Usage

```
Segmentation_BaRatin(
  H,
  Q,
  time = 1:NROW(Q),
  uH = 0 * H,
  uQ = 0 * Q,
  flavor = c("BaRatin", "BaRatinBAC"),
  controlMatrix = matrix(1),
  priors = get3FlatPriors(H, Q),
  nSmax = 2,
  doQuickApprox = TRUE,
  nMin = ifelse(doQuickApprox, 3, 1),
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  remnant = list(RBaM::remnantErrorModel()),
  temp.folder = file.path(tempdir(), "BaRatin")
)
```

Arguments

H	real vector, stage
Q	real vector, discharge
time	vector (numeric or date), time
uH	real vector, stage uncertainty (as a standard deviation)
uQ	real vector, discharge uncertainty (as a standard deviation)
flavor	string, BaRatin flavor
controlMatrix	square matrix, control matrix
priors	list, list of RBaM::parameter objects containing information for each parameter of the rating curve. Length 3*NROW(controlMatrix)
nSmax	integer, maximum number of segments to assess
doQuickApprox	logical, use quick approximation? see ?Segmentation_quickApprox
nMin	integer, minimum number of observations by segment
mcmc_options	mcmcOptions object, options used to generate MCMC samples (e.g. size and adaption tunings), see ?RBaM::mcmcOptions. Only used when doQuickApprox=FALSE.
mcmc_cooking	mcmcCooking object, options used to post-process MCMC samples (e.g. burn and slice), see ?RBaM::mcmcCooking. Only used when doQuickApprox=FALSE.
remnant	list of one remnantErrorModel object, see ?RBaM::remnantErrorModel
temp.folder	directory, temporary directory to write BaM computations. Only used when doQuickApprox=FALSE.

Value

An object of class `fittedModel()`, containing the following fields:

1. data: data frame, column-binding all the arguments of the function except parameters
2. parameters: data frame, fitted parameters + standard error

Examples

```
sg=Segmentation_BaRatin(H=ArdecheRiverGaugings$H,Q=ArdecheRiverGaugings$Q,
                        uQ=ArdecheRiverGaugings$uQ,time=ArdecheRiverGaugings$Date,
                        # MCMC options are modified to speed-up example.
                        # Using default MCMC options is safer and is recommended.
                        mcmc_options=mcmcOptions(nAdapt=30,nCycles=20))
if(!is.null(sg)){
  plot(sg,dataPlotType='ty')
  patchwork::wrap_plots(plot(sg,type='fits'))
}
```

`Segmentation_Engine` *Segmentation engine*

Description

Segmentation procedure for a **known** given number of segments

Usage

```
Segmentation_Engine(
  obs,
  time = 1:length(obs),
  u = 0 * obs,
  nS = 2,
  doQuickApprox = (nS < 3),
  nMin = ifelse(doQuickApprox, 3, 1),
  nSim = 500,
  varShift = FALSE,
  alpha = 0.1,
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  temp.folder = file.path(tempdir(), "BaM"),
  mu_prior = list(NULL)
)
```

Arguments

`obs`	real vector, observations
`time`	vector, time in POSIXct, string or numeric format
`u`	real vector, uncertainty in observations (as a standard deviation)
`nS`	integer, number of segments
`doQuickApprox`	logical, use quick approximation? see ?Segmentation_quickApprox
`nMin`	integer, minimum number of observations by segment
`nSim`	integer, number of Monte-Carlo simulated values for shift times. Only used when doQuickApprox=TRUE.
`varShift`	logical, allow for a shifting variance? Only used when doQuickApprox=TRUE.
`alpha`	real in (0;1), type-I error level of the underlying step-change test. Only used when doQuickApprox=TRUE.
`mcmc_options`	mcmcOptions object, options used to generate MCMC samples (e.g. size and adaption tunings), see ?RBaM::mcmcOptions. Only used when doQuickApprox=FALSE.
`mcmc_cooking`	mcmcCooking object, options used to post-process MCMC samples (e.g. burn and slice), see ?RBaM::mcmcCooking. Only used when doQuickApprox=FALSE.

temp.folder	directory, temporary directory to write BaM computations. Only used when doQuickApprox=FALSE.
mu_prior	list, object describing prior knowledge on the mean of residuals for each segment. Only used when doQuickApprox=FALSE.

Value

An object of class simpleSegmentation(), containing the following fields:

1. data: data frame, all data with their respective periods after segmentation
2. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC
3. mcmc: data frame, MCMC simulations
4. DIC: real, DIC estimation
5. origin.date: positive real or date, date describing origin of the segmentation for a sample. Useful for recursive segmentation.

Source

https://theses.hal.science/tel-03211343

doi:10.1029/2020WR028607

Examples

```
# Default segmentation using quick approximation
res=Segmentation_Engine(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH)
res$shifts
hist(res$mcmc$tau)
# Segmentation using BaM, allowing more than 2 segments and the use of priors
## Not run:
# This line of code is wrapped in \dontrun{} since it relies
# on the installation of an executable with the package RBaM.
# See ?RBaM::downloadBaM for downloading and installing this executable.
prior=list(RBaM::parameter(name=paste0('mu1'),init=6,prior.dist='Gaussian',prior.par=c(6,5)),
       RBaM::parameter(name=paste0('mu2'),init=8,prior.dist='Gaussian',prior.par=c(8,2)))
res=Segmentation_Engine(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH,
     doQuickApprox=FALSE,nS=3,mu_prior=prior)

## End(Not run)
```

Segmentation_quickApprox

Segmentation engine - quick approximation algorithm

Description

A quick approximation to the segmentation procedure for either one or two segments, and no prior information. The approximation is based on a likelihood-ratio test for a single step change at an unknown position. DIC0 is hence replaced with the deviance of a no-shift model M0, while DIC1 is replaced by the maximum deviance of a single-shift model M1 + the critical value of the test. This way, a change will be detected when DIC1<DIC0, i.e. when deviance1+critical value<deviance0, i.e. when deviance0-deviance1>critical value, which precisely corresponds to the outcome of the likelihood ratio test.
Critical values are computed using the approximations suggested by Gombay and Horvath (1994, 1996a, 1996b, 1997), as detailled in Renard (2006, chapter 3, section I.1.5, p. 101, https://hal.science/tel-02588353).
The uncertainty in tau is still estimated by plugging-in point-estimates of mu1, mu2 and sigma for each possible value of tau, and computing the associated likelihood. This likelihood can then be normalized to estimate the posterior pdf of tau, which can then be numerically integrated to give the posterior cdf of tau. Samples of tau values can then be simulated from this cdf using uniform sampling then inverse-cdf transformation. The resulting algorithm is MCMC-free and independent of RBaM.

Usage

```
Segmentation_quickApprox(
  obs,
  time = 1:length(obs),
  u = 0 * obs,
  nS = 2,
  nMin = 3,
  nSim = 500,
  varShift = FALSE,
  alpha = 0.1
)
```

Arguments

obs	real vector, observations
time	vector, time in POSIXct, string or numeric format
u	real vector, uncertainty in observations (as a standard deviation)
nS	integer, number of segments, either 1 or 2
nMin	integer, minimum number of observations by segment
nSim	integer, number of simulated tau values
varShift	logical, allow for a shifting variance?
alpha	real in (0;1), type-I error level of the underlying step-change test.

Value

An object of class simpleSegmentation(), containing the following fields:

1. data: data frame, all data with their respective periods after segmentation

2. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC
3. mcmc: data frame, MCMC simulations
4. DIC: real, DIC estimation
5. origin.date: positive real or date, date describing origin of the segmentation for a sample. Useful for recursive segmentation.

Examples

```
# Segmentation into two segments for the RhoneRiverAMAX data set (details in ?RhoneRiverAMAX)
res=Segmentation_quickApprox(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH,nS=2)
res$shifts
hist(res$mcmc$tau)
```

`Segmentation_Recessions` *Recessions-based segmentation*

Description

Segmentation based on a segmentation of the lowest points reached by stage recessions. For more details see Chapter 3 of the PhD thesis of Matteo Darienzo (2021, https://theses.hal.science/tel-03211343).

Usage

```
Segmentation_Recessions(
  rec,
  equation = c("none", "M1", "M2", "M3", "M4", "M5", "M6", "M7", "M8", "M9"),
  nSmax = 2,
  doQuickApprox = TRUE,
  nMin = ifelse(doQuickApprox, 3, 1),
  nSim = 500,
  varShift = FALSE,
  alpha = 0.1,
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  mu_prior = list(),
  remnant = list(getConstantRemnant(rec)),
  temp.folder = file.path(tempdir(), "Recessions")
)
```

Arguments

rec	`extractedRecessions()` object, resulting from a call to function `Extract_Recessions()`

`equation`	string, equation used to model recessions. If 'none' (default), the minimum of each recession is extracted directly (hence no recession model is used) and seegmented. If a model between 'M1' and 'M9' is used, recessions are modeled and the 'asymptoti stage' parameter is segmented. For more details on the model equations, see Chapter 3 of the PhD thesis of Matteo Darienzo (2021, https://theses.hal.science/tel-03211343) or call getRecessionEquations().
`nSmax`	integer, maximum number of segments to assess
`doQuickApprox`	logical, use quick approximation? see ?Segmentation_quickApprox
`nMin`	integer, minimum number of observations by segment
`nSim`	integer, number of Monte-Carlo simulated values for shift times. Only used when doQuickApprox=TRUE.
`varShift`	logical, allow for a shifting variance? Only used when doQuickApprox=TRUE.
`alpha`	real in (0;1), type-I error level of the underlying step-change test. Only used when doQuickApprox=TRUE.
`mcmc_options`	mcmcOptions object, see ?RBaM::mcmcOptions
`mcmc_cooking`	mcmcCooking object, see ?RBaM::mcmcCooking
`mu_prior`	list, object describing prior knowledge on the mean of residuals for each segment. Only used when doQuickApprox=FALSE.
`remnant`	list of one remnantErrorModel object, see ?RBaM::remnantErrorModel
`temp.folder`	string, directory where config and result files are stored.

Value

An object of class `fittedModel()`, containing the following fields:

1. data: data frame, column-binding all the arguments of the function except parameters
2. parameters: data frame, fitted parameters + standard error

Source

https://theses.hal.science/tel-03211343

Examples

```
rec=Extract_Recessions(time=ArdecheRiverStage$Date,H=ArdecheRiverStage$H,
                       nSlim=10) # used to speed-up example, but nSlim=1 is recommended.
 sg=Segmentation_Recessions(rec)
 plot(sg)
```

`Segmentation_Recursive` *Recursive Segmentation*

Description

Recursive segmentation procedure for an **unknown** number of segments

Usage

```
Segmentation_Recursive(
  obs,
  time = 1:length(obs),
  u = 0 * obs,
  nSmax = 2,
  doQuickApprox = (nSmax < 3),
  nMin = ifelse(doQuickApprox, 3, 1),
  nSim = 500,
  varShift = FALSE,
  alpha = 0.1,
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  temp.folder = file.path(tempdir(), "BaM"),
  mu_prior = list()
)
```

Arguments

`obs`	real vector, observations
`time`	vector, time in POSIXct, string or numeric format
`u`	real vector, uncertainty in observations (as a standard deviation)
`nSmax`	integer, maximum number of segments to assess
`doQuickApprox`	logical, use quick approximation? see ?Segmentation_quickApprox
`nMin`	integer, minimum number of observations by segment
`nSim`	integer, number of Monte-Carlo simulated values for shift times. Only used when doQuickApprox=TRUE.
`varShift`	logical, allow for a shifting variance? Only used when doQuickApprox=TRUE.
`alpha`	real in (0;1), type-I error level of the underlying step-change test. Only used when doQuickApprox=TRUE.
`mcmc_options`	mcmcOptions object, options used to generate MCMC samples (e.g. size and adaption tunings), see ?RBaM::mcmcOptions. Only used when doQuickApprox=FALSE.
`mcmc_cooking`	mcmcCooking object, options used to post-process MCMC samples (e.g. burn and slice), see ?RBaM::mcmcCooking. Only used when doQuickApprox=FALSE.

`temp.folder`	directory, temporary directory to write BaM computations. Only used when doQuickApprox=FALSE.
`mu_prior`	list, object describing prior knowledge on the mean of residuals for each segment. Only used when doQuickApprox=FALSE.

Value

An object of class `recursiveSegmentation()`, containing the following fields:

1. nS: integer, final number of segments
2. data: data frame, all data with their respective periods after segmentation
3. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC
4. tree: `segmentationTree()` object, recursion tree.
5. results: list, intermediate results at each stage of the recursion.

Examples

```
res=Segmentation_Recursive(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Date)
res$shifts
res$tree
plot(res$data$obs,col=res$data$period)
# Create an artificial series with more changes to see recursion in action
obs=c(RhoneRiverAMAX$H,RhoneRiverAMAX$H)
res=Segmentation_Recursive(obs=obs)
res$shifts
res$tree
plot(res$data$obs,col=res$data$period)
```

Segmentation_RecursiveModeling

Recursive Modeling Segmentation

Description

Recursive Modeling Segmentation procedure for an **unknown** number of segments, aimed at detecting changes in the relation between X and Y.

Usage

```
Segmentation_RecursiveModeling(
  x,
  y,
  time = 1:NROW(y),
  uX = 0 * x,
  uY = 0 * y,
  nSmax = 2,
```

```
  doQuickApprox = (nSmax < 3),
  nMin = ifelse(doQuickApprox, 3, 1),
  Fit_funk = Fit_LinearRegression,
  ...,
  nSim = 500,
  varShift = FALSE,
  alpha = 0.1,
  mcmc_options = RBaM::mcmcOptions(),
  mcmc_cooking = RBaM::mcmcCooking(),
  temp.folder = file.path(tempdir(), "BaM"),
  mu_prior = list()
)
```

Arguments

x	real matrix or data frame, predictors
y	real vector, predictand
time	vector (numeric or date), time
uX	real matrix or data frame, uncertainty in predictors x (as a standard deviation). May be ignored if the fitting function 'Fit_funk' does not handle input uncertainty (which is often the case).
uY	real vector, uncertainty in predictand y (as a standard deviation). May be ignored if the fitting function 'Fit_funk' does not handle output uncertainty (which is often the case).
nSmax	integer, maximum number of segments to assess
doQuickApprox	logical, use quick approximation? see ?Segmentation_quickApprox
nMin	integer, minimum number of observations by segment
Fit_funk	function, function used to fit the model
...	optional arguments passed to function Fit_funk
nSim	integer, number of Monte-Carlo simulated values for shift times. Only used when doQuickApprox=TRUE.
varShift	logical, allow for a shifting variance? Only used when doQuickApprox=TRUE.
alpha	real in (0;1), type-I error level of the underlying step-change test. Only used when doQuickApprox=TRUE.
mcmc_options	mcmcOptions object, options used to generate MCMC samples (e.g. size and adaption tunings), see ?RBaM::mcmcOptions. Only used when doQuickApprox=FALSE.
mcmc_cooking	mcmcCooking object, options used to post-process MCMC samples (e.g. burn and slice), see ?RBaM::mcmcCooking. Only used when doQuickApprox=FALSE.
temp.folder	directory, temporary directory to write BaM computations. Only used when doQuickApprox=FALSE.
mu_prior	list, object describing prior knowledge on the mean of residuals for each segment. Only used when doQuickApprox=FALSE.

Value

an object of class recursiveModeling()

Examples

```
res=Segmentation_RecursiveModeling(x=ArdecheRiverGaugings$H,
                                   y=ArdecheRiverGaugings$Q,
                                   uY=ArdecheRiverGaugings$uQ,
                                   time=ArdecheRiverGaugings$Date)
res$segmentation$shifts
res$segmentation$tree
plot(res$data$x1,res$data$y,col=res$data$period)
```

---

simpleSegmentation *simpleSegmentation object constructor.*

Description

Creates a new instance of a 'simpleSegmentation' object, used to store the results of a segmentation with an known number of segment

Usage

```
simpleSegmentation(
  obs,
  time = 1:length(obs),
  u = 0 * obs,
  data = NULL,
 shifts = data.frame(tau = numeric(0), I95_lower = numeric(0), I95_upper = numeric(0)),
  mcmc = data.frame(),
  DIC = NA,
  origin.date = min(time)
)
```

Arguments

obs	real vector, observations
time	real vector, time
u	real vector, uncertainty in observations (as a standard deviation)
data	data frame, data summary, should contain the following columns: time,obs,u,I95_lower,I95_upper,period. If NULL, will be automatically intialised from obs,time,u. If provided, will supersede obs, time, u.
shifts	data frame, detected shifts (with columns tau, I95_lower and I95_upper)
mcmc	data frame, MCMC simulations
DIC	real, DIC estimation
origin.date	dates origin (useful for date conversions)

Value

An object of class simpleSegmentation(), containing the following fields:

1. data: data frame, all data with their respective periods after segmentation
2. shifts: data frame, all detected shift times and their uncertainty in numeric or POSIXct format in UTC
3. mcmc: data frame, MCMC simulations
4. DIC: real, DIC estimation
5. origin.date: positive real or date, date describing origin of the segmentation for a sample. Useful for recursive segmentation.

Examples

```
sg <- simpleSegmentation(obs=RhoneRiverAMAX$H,time=RhoneRiverAMAX$Year,u=RhoneRiverAMAX$uH)
```

---

time_to_numeric *Time to numeric format*

Description

Cast a time or date into a numeric interpreted as the duration(in dayss) after an origin date.

Usage

```
time_to_numeric(date)
```

Arguments

date	vector, time in POSIXct, character or Date format

Value

List with the following components :

1. origin.date: time in POSIXct, character or Date format, corresponding to the oldest date from the input d.
2. d: real vector, duration (in days) after origin.date.

Examples

```
time_to_numeric(c(as.Date('2024/02/19'),as.Date('2024/02/10')))
time_to_numeric(c(as.POSIXct('2024-04-19 12:30:00',tz='UTC'),
                as.POSIXct('2024-03-19 18:30:00',tz='UTC')))
time_to_numeric(c(as.character('20240419'),as.character('20240119')))
```

Index