Package 'harbinger'

November 11, 2023

Title An Unified Time Series Event Detection Framework

Version 1.0.737

Description By analyzing time series, it is possible to observe significant changes in the behavior of observations that frequently characterize events. Events present themselves as anomalies, change points, or motifs. In the literature, there are several methods for detecting events. However, searching for a suitable time series method is a complex task, especially considering that the nature of events is often unknown. This work presents Harbinger, a framework for integrating and analyzing event detection methods. Harbinger contains several state-of-the-art methods described in Salles et al. (2020) <doi:10.5753/sbbd.2020.13626>.

License MIT + file LICENSE

URL https://github.com/cefet-rj-dal/harbinger, https://cefet-rj-dal.github.io/harbinger/

Encoding UTF-8

RoxygenNote 7.2.3

Imports stats, daltoolbox, TSPred, tsmp, dtwclust, rugarch, forecast, ggplot2, changepoint, strucchange, stringr, dplyr, reticulate

Config/reticulate list(packages = list(list(package = ``scipy''), list(package = ``torch''), list(package = ``pandas''), list(package = ``numpy''), list(package = ``matplotlib''), list(package = ``scikit-learn'')))

NeedsCompilation no

Author Rebecca Salles [aut],
Janio Lima [aut],
Lais Baroni [aut],
Antonio Castro [aut],
Leonardo Carvalho [aut],
Heraldo Borges [aut],
Diego Carvalho [aut],
Rafaelli Coutinho [aut],
Eduardo Bezerra [aut],
Esther Pacitti [aut],
Fabio Porto [aut],

Eduardo Ogasawara [aut, ths, cre]
(<https://orcid.org/0000-0002-0466-0626>),
Federal Center for Technological Education of Rio de Janeiro (CEFET/RJ)
[cph]

Maintainer Eduardo Ogasawara <eogasawara@ieee.org>

Repository CRAN

Date/Publication 2023-11-11 10:53:24 UTC

R topics documented:

detect *Detect events in time series*

Description

Event detection using a fitted Harbinger model

Usage

```
detect(obj, ...)
```

Arguments

obj	harbinger object
...	optional arguments.

Value

a data frame with the index of observations and if they are identified or not as an event, and their type

Examples

```
# See ?hanc_ml for an example of anomaly detection using machine learning classification
# See ?hanr_arima for an example of anomaly detection using ARIMA
# See ?hanr_fbiad for an example of anomaly detection using FBIAD
# See ?hanr_garch for an example of anomaly detection using GARCH
# See ?hanr_kmeans for an example of anomaly detection using kmeans clustering
# See ?hanr_ml for an example of anomaly detection using machine learning regression
# See ?hanr_cf_arima for an example of change point detection using ARIMA
# See ?hanr_cf_ets for an example of change point detection using ETS
# See ?hanr_cf_lr for an example of change point detection using linear regression
# See ?hanr_cf_garch for an example of change point detection using GARCH
# See ?hanr_cf_scp for an example of change point detection using the seminal algorithm
# See ?hmo_sax for an example of motif discovery using SAX
# See ?hmu_pca for an example of anomaly detection in multivariate time series using PCA
```

hanct_dtw *Anomaly detector using DTW*

Description

Anomaly detection using DTW The DTW is applied to the time series. When seq equals one, observations distant from the closest centroids are labeled as anomalies. When seq is grater than one, sequences distant from the closest centroids are labeled as discords. It wraps the tsclust presented in the dtwclust library.

Usage

```
hanct_dtw(seq = 1, centers = NA)
```

Arguments

seq	sequence size
centers	number of centroids

Value

hanct_dtw object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 1
dataset <- har_examples$example1
head(dataset)

# setting up time series regression model
model <- hanct_dtw()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hanct_kmeans *Anomaly detector using kmeans*

Description

Anomaly detection using kmeans The kmeans is applied to the time series. When seq equals one, observations distant from the closest centroids are labeled as anomalies. When seq is grater than one, sequences distant from the closest centroids are labeled as discords. It wraps the kmeans presented in the stats library.

Usage

```
hanct_kmeans(seq = 1, centers = NA)
```

Arguments

seq	sequence size
centers	number of centroids

Value

hanct_kmeans object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 1
dataset <- har_examples$example1
head(dataset)

# setting up time series regression model
model <- hanct_kmeans()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hanc_ml *Anomaly detector based on machine learning classification*

Description

Anomaly detection using daltoolbox classification. A training and test set should be used. The training set must contain labeled events. A set of preconfigured of classification methods are described in https://cefet-rj-dal.github.io/daltoolbox/. They include: cla_majority, cla_dtree, cla_knn, cla_mlp, cla_nb, cla_rf, cla_svm

Usage

```
hanc_ml(model)
```

Arguments

model	DALToolbox classification model

Value

hanc_ml object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 17
dataset <- har_examples$example17
dataset$event <- factor(dataset$event, labels=c("FALSE", "TRUE"))
slevels <- levels(dataset$event)

# separating into training and test
train <- dataset[1:80,]
test <- dataset[-(1:80),]

# normalizing the data
norm <- minmax()
norm <- fit(norm, train)
train_n <- transform(norm, train)

# establishing decision tree method
model <- hanc_ml(cla_dtree("event", slevels))

# fitting the model
model <- fit(model, train_n)

# evaluating the detections during testing
test_n <- transform(norm, test)

detection <- detect(model, test_n)
print(detection[(detection$event),])
```

---

hanr_arima *Anomaly detector using ARIMA.*

Description

Anomaly detection using ARIMA The ARIMA model adjusts to the time series. Observations distant from the model are labeled as anomalies. It wraps the ARIMA model presented in the forecast library.

Usage

```
hanr_arima()
```

Value

hanr_arima object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 1
dataset <- har_examples$example1
head(dataset)

# setting up time series regression model
model <- hanr_arima()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hanr_fbiad *Anomaly detector using FBIAD*

Description

Anomaly detector using FBIAD

Usage

```
hanr_fbiad(sw_size = 30)
```

Arguments

sw_size	Window size for FBIAD

Value

hanr_fbiad object Forward and Backward Inertial Anomaly Detector (FBIAD) detects anomalies in time series. Anomalies are observations that differ from both forward and backward time series inertia doi:10.1109/IJCNN55064.2022.9892088.

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 1
dataset <- har_examples$example1
head(dataset)

# setting up time series regression model
model <- hanr_fbiad()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hanr_garch *Anomaly detector using GARCH*

Description

Anomaly detection using GARCH The GARCH model adjusts to the time series. Observations distant from the model are labeled as anomalies. It wraps the ugarch model presented in the rugarch library.

Usage

```
hanr_garch()
```

Value

hanr_garch object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 14
dataset <- har_examples$example14
head(dataset)
```

```
# setting up time series regression model
model <- hanr_garch()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

`hanr_histogram` *Anomaly detector using histogram*

Description

Anomaly detector using histogram

Usage

```
hanr_histogram(density_threshold = 0.05)
```

Arguments

`density_threshold`
: It is the minimum frequency for a bin to not be considered an anomaly. Default value is 5%.

Value

hanr_histogram object histogram based method to detect anomalies in time series. Bins with smaller amount of observations are considered anomalies. Values below first bin or above last bin are also considered anomalies.>.

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 1
dataset <- har_examples$example1
head(dataset)

# setting up time series regression model
model <- hanr_histogram()
```

```
# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hanr_ml *Anomaly detector based on machine learning regression.*

Description

Anomaly detection using daltoolbox regression The regression model adjusts to the time series. Observations distant from the model are labeled as anomalies. A set of preconfigured regression methods are described in https://cefet-rj-dal.github.io/daltoolbox/. They include: ts_elm, ts_conv1d, ts_lstm, ts_mlp, ts_rf, ts_svm

Usage

```
hanr_ml(model, sw_size = 15)
```

Arguments

model	DALToolbox regression model
sw_size	sliding window size

Value

hanr_ml object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 1
dataset <- har_examples$example1
head(dataset)

# setting up time series regression model
model <- hanr_ml(ts_elm(ts_norm_gminmax(), input_size=4, nhid=3, actfun="purelin"))

# fitting the model
model <- fit(model, dataset$serie)
```

```
detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

han_autoencoder *Anomaly detector using autoencoder*

Description

Anomaly detector using autoencoder

Usage

```
han_autoencoder(input_size, encode_size)
```

Arguments

input_size	Establish the input size for the autoencoder anomaly detector. It is the size of the output also.
encode_size	The encode size for the autoencoder.

Value

han_autoencoder object histogram based method to detect anomalies in time series. Bins with smaller amount of observations are considered anomalies. Values below first bin or above last bin are also considered anomalies.>.

Examples

```
# setting up time series regression model
#Use the same example of hanr_fbiad changing the constructor to:
model <- han_autoencoder(3,1)
```

---

harbinger *Harbinger*

Description

Ancestor class for time series event detection

Usage

```
harbinger()
```

Value

Harbinger object

Examples

```
# See ?hanc_ml for an example of anomaly detection using machine learning classification
# See ?hanr_arima for an example of anomaly detection using ARIMA
# See ?hanr_fbiad for an example of anomaly detection using FBIAD
# See ?hanr_garch for an example of anomaly detection using GARCH
# See ?hanr_kmeans for an example of anomaly detection using kmeans clustering
# See ?hanr_ml for an example of anomaly detection using machine learning regression
# See ?hanr_cf_arima for an example of change point detection using ARIMA
# See ?hanr_cf_ets for an example of change point detection using ETS
# See ?hanr_cf_lr for an example of change point detection using linear regression
# See ?hanr_cf_garch for an example of change point detection using GARCH
# See ?hanr_cf_scp for an example of change point detection using the seminal algorithm
# See ?hmo_sax for an example of motif discovery using SAX
# See ?hmu_pca for an example of anomaly detection in multivariate time series using PCA
```

---

har_conv1d *Conv1D*

Description

Creates a time series prediction object that uses the Conv1D. It wraps the pytorch library.

Usage

```
har_conv1d(preprocess = NA, input_size = NA, epochs = 10000L)
```

Arguments

preprocess	normalization
input_size	input size for machine learning model
epochs	maximum number of epochs

Value

a har_conv1d object.

Examples

```
library(daltoolbox)
data(sin_data)
ts <- ts_data(sin_data$y, 10)
ts_head(ts, 3)
```

```
samp <- ts_sample(ts, test_size = 5)
io_train <- ts_projection(samp$train)
io_test <- ts_projection(samp$test)

model <- har_conv1d(ts_norm_gminmax(), input_size=4, epochs = 10000L)
model <- fit(model, x=io_train$input, y=io_train$output)

prediction <- predict(model, x=io_test$input[1,], steps_ahead=5)
prediction <- as.vector(prediction)
output <- as.vector(io_test$output)

ev_test <- evaluate(model, output, prediction)
ev_test
```

`har_eval` *Evaluation of event detection*

Description

Evaluation of event detection (traditional hard evaluation)

Usage

```
har_eval()
```

Value

`har_eval` object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using the time series 14
dataset <- har_examples$example14
head(dataset)

# setting up time change point using GARCH
model <- hcp_garch()

# fitting the model
model <- fit(model, dataset$serie)

# making detections
detection <- detect(model, dataset$serie)
```

```
# filtering detected events
print(detection[(detection$event),])

# evaluating the detections
evaluation <- evaluate(har_eval(), detection$event, dataset$event)
print(evaluation$confMatrix)

# ploting the results
grf <- har_plot(model, dataset$serie, detection, dataset$event)
plot(grf)
```

`har_eval_soft` *Evaluation of event detection*

Description

Evaluation of event detection using SoftED doi:10.48550/arXiv.2304.00439

Usage

```
har_eval_soft(sw_size = 15)
```

Arguments

`sw_size`	tolerance window size

Value

`har_eval_soft` object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using the time series 14
dataset <- har_examples$example14
head(dataset)

# setting up time change point using GARCH
model <- hcp_garch()

# fitting the model
model <- fit(model, dataset$serie)

# making detections
detection <- detect(model, dataset$serie)
```

```
# filtering detected events
print(detection[(detection$event),])

# evaluating the detections
evaluation <- evaluate(har_eval_soft(), detection$event, dataset$event)
print(evaluation$confMatrix)

# ploting the results
grf <- har_plot(model, dataset$serie, detection, dataset$event)
plot(grf)
```

`har_examples` *Synthetic time series for event detection*

Description

A list of univariate time series for event detection

- example1: a single (uncommon value) anomaly example in a stationary time series
- example2: a single (common value) anomaly example in a stationary time series
- example3: a single anomaly example in a trend time series
- example4: a change point in a time series
- example5: a change point with anomaly (uncommon value) in a time series
- example6: a change point with anomaly (common value) in a time series
- example7: a change point with anomaly (uncommon value) in a seasonal time series
- example8: a change point with anomaly (common value) in a seasonal time series
- example9: multi behavior time series (1-200: stationary, 201-400: trend, 401-600: structural break; 601-800: heteroscedasticity, 801:1000 random walk)
- example10: multiple anomalies
- example11: stationary time series
- example12: trend time series
- example13: structural break
- example14: heterocedasticity
- example15: motif
- example16: discord
- example17: repetitive anomaly
- example18: repetitive anomaly

#'

Usage

```
data(har_examples)
```

Format

A list of time series.

Source

Harbinger package

References

Harbinger package

Examples

```
data(har_examples)
serie <- har_examples$example1
```

`har_examples_multi` *Synthetic time series for event detection*

Description

A list of multivariate time series for event detection

- example1: an single multivariate anomaly

#'

Usage

```
data(har_examples_multi)
```

Format

A list of multivariate time series.

Source

Harbinger package

References

Harbinger package

Examples

```
data(har_examples_multi)
serie <- har_examples_multi$example1
```

har_lstm *LSTM*

Description

Creates a time series prediction object that uses the LSTM. It wraps the pytorch library.

Usage

```
har_lstm(preprocess = NA, input_size = NA, epochs = 10000L)
```

Arguments

preprocess	normalization
input_size	input size for machine learning model
epochs	maximum number of epochs

Value

a har_lstm object.

Examples

```
library(daltoolbox)
data(sin_data)
ts <- ts_data(sin_data$y, 10)
ts_head(ts, 3)

samp <- ts_sample(ts, test_size = 5)
io_train <- ts_projection(samp$train)
io_test <- ts_projection(samp$test)

model <- har_lstm(ts_norm_gminmax(), input_size=4, epochs = 10000L)
model <- fit(model, x=io_train$input, y=io_train$output)

prediction <- predict(model, x=io_test$input[1,], steps_ahead=5)
prediction <- as.vector(prediction)
output <- as.vector(io_test$output)

ev_test <- evaluate(model, output, prediction)
ev_test
```

har_plot *Plot event detection on a time series*

Description

It accepts as harbinger, a time series, a data.frame of events, a parameter to mark the detected change points, a threshold for the y-axis and an index for the time series

Usage

```
har_plot(
  obj,
  serie,
  detection,
  event = NULL,
  mark.cp = TRUE,
  ylim = NULL,
  idx = NULL,
  pointsize = 0.5,
  colors = c("green", "blue", "red", "purple")
)
```

Arguments

obj	harbinger detector
serie	time series
detection	detection
event	events
mark.cp	show change points
ylim	limits for y-axis
idx	labels for x observations
pointsize	default point size
colors	default colors for event detection: green is TP, blue is FN, red is FP, purple means observations that are part of a sequence.

Value

A time series plot with marked events

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)
```

```
#Using the time series 1
dataset <- har_examples$example1
head(dataset)

# setting up time change point using GARCH
model <- hanr_arima()

# fitting the model
model <- fit(model, dataset$serie)

# making detections
detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])

# evaluating the detections
evaluation <- evaluate(har_eval_soft(), detection$event, dataset$event)
print(evaluation$confMatrix)

# ploting the results
grf <- har_plot(model, dataset$serie, detection, dataset$event)
plot(grf)
```

`hcd_page_hinkley` *Adapted Page Hinkley method*

Description

Change-point detection method works by computing the observed values and their mean up to the current moment doi:10.2307/2333009.

Usage

```
hcd_page_hinkley(
  min_instances = 30,
  delta = 0.005,
  threshold = 50,
  alpha = 1 - 1e-04
)
```

Arguments

`min_instances`	The minimum number of instances before detecting change
`delta`	The delta factor for the Page Hinkley test
`threshold`	The change detection threshold (lambda)
`alpha`	The forgetting factor, used to weight the observed value and the mean

Value

hcp_page_hinkley object

Examples

```
library("daltoolbox")

n <- 100  # size of each segment
serie1 <- c(sin((1:n)/pi), 2*sin((1:n)/pi), 10 + sin((1:n)/pi),
           10-10/n*(1:n)+sin((1:n)/pi)/2, sin((1:n)/pi)/2)
serie2 <- 2*c(sin((1:n)/pi), 2*sin((1:n)/pi), 10 + sin((1:n)/pi),
           10-10/n*(1:n)+sin((1:n)/pi)/2, sin((1:n)/pi)/2)
data <- data.frame(serie1, serie2)#'
event <- rep(FALSE, nrow(data))

model <- fit(hcd_page_hinkley(threshold=3), data)
detection <- detect(model, data)
print(detection[(detection$event),])
```

---

hcp_amoc *At most one change (AMOC) method*

Description

Change-point detection method that focus on identify one change point in mean/variance doi:10.1093/biomet/57.1.1. It wraps the amoc implementation available in the changepoint library.

Usage

```
hcp_amoc()
```

Value

hcp_amoc object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up time series regression model
model <- hcp_amoc()
```

```
# fitting the model
model <- fit(model, dataset$serie)

# execute the detection method
detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hcp_binseg *Binary segmentation (BinSeg) method*

Description

Change-point detection method that focus on identify change points in mean/variance doi:10.2307/2529204. It wraps the BinSeg implementation available in the changepoint library.

Usage

```
hcp_binseg(Q = 2)
```

Arguments

Q	The maximum number of change-points to search for using the BinSeg method

Value

hcp_binseg object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up time series regression model
model <- hcp_binseg()

# fitting the model
model <- fit(model, dataset$serie)

# execute the detection method
detection <- detect(model, dataset$serie)
```

```
# filtering detected events
print(detection[(detection$event),])
```

---

hcp_cf_arima *Change Finder using ARIMA*

Description

Change-point detection is related to event/trend change detection. Change Finder ARIMA detects change points based on deviations relative to ARIMA model doi:10.1109/TKDE.2006.1599387. It wraps the ARIMA model presented in the forecast library.

Usage

```
hcp_cf_arima(sw_size = NULL)
```

Arguments

sw_size	Sliding window size

Value

hcp_cf_arima object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up change point method
model <- hcp_cf_arima()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

hcp_cf_ets *Change Finder using ETS*

Description

Change-point detection is related to event/trend change detection. Change Finder ETS detects change points based on deviations relative to trend component (T), a seasonal component (S), and an error term (E) model doi:10.1109/TKDE.2006.1599387. It wraps the ETS model presented in the forecast library.

Usage

```
hcp_cf_ets(sw_size = 7)
```

Arguments

sw_size	Sliding window size

Value

hcp_cf_ets object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up change point method
model <- hcp_cf_ets()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

hcp_cf_lr *Change Finder using LR*

Description

Change-point detection is related to event/trend change detection. Change Finder LR detects change points based on deviations relative to linear regression model doi:10.1109/TKDE.2006.1599387.

Usage

```
hcp_cf_lr(sw_size = 30)
```

Arguments

sw_size	Sliding window size

Value

hcp_cf_lr object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up change point method
model <- hcp_cf_lr()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

hcp_chow *Chow test method*

Description

Change-point detection method that focus on identifying structural changes doi:10.18637/jss.v007.i02. It wraps the Fstats and breakpoints implementation available in the strucchange library.

Usage

```
hcp_chow()
```

Value

hcp_chow object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up change point method
model <- hcp_chow()

# fitting the model
model <- fit(model, dataset$serie)

# execute the detection method
detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

hcp_garch *Change Finder using GARCH*

Description

Change-point detection is related to event/trend change detection. Change Finder GARCH detects change points based on deviations relative to linear regression model doi:10.1109/TKDE.2006.1599387. It wraps the GARCH model presented in the rugarch library.

Usage

```
hcp_garch(sw_size = 5)
```

Arguments

sw_size	Sliding window size

Value

hcp_garch object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 14
dataset <- har_examples$example14
head(dataset)

# setting up change point method
model <- hcp_garch()

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hcp_gft *Generalized Fluctuation Test (GFT)*

Description

GFT detection method focuses on identifying structural changes doi:10.18637/jss.v007.i02. It wraps the breakpoints implementation available in the strucchange library.

Usage

```
hcp_gft()
```

Value

hcp_chow object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up change point method
model <- hcp_gft()

# fitting the model
model <- fit(model, dataset$serie)

# execute the detection method
detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hcp_pelt *Pruned exact linear time (PELT) method*

Description

Change-point detection method that focus on identifying multiple exact change points in mean/variance doi:10.1080/01621459.2012.737745. It wraps the BinSeg implementation available in the changepoint library.

Usage

```
hcp_pelt()
```

Value

hcp_pelt object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
```

```
head(dataset)

# setting up change point method
model <- hcp_pelt()

# fitting the model
model <- fit(model, dataset$serie)

# execute the detection method
detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hcp_scp *Seminal change point*

Description

Change-point detection is related to event/trend change detection. Seminal change point detects change points based on deviations of linear regression models adjusted with and without a central observation in each sliding window <10.1145/312129.312190>.

Usage

```
hcp_scp(sw_size = 30)
```

Arguments

sw_size	Sliding window size

Value

hcp_scp object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 6
dataset <- har_examples$example6
head(dataset)

# setting up change point method
model <- hcp_scp()
```

```
# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

---

hmo_base36 *Motif discovery using base36*

Description

Motif discovery using base36 doi:10.1007/s10618-007-0064-z

Usage

```
hmo_base36(a, w, qtd)
```

Arguments

a	alphabet size
w	word size
qtd	number of occurrences to be classified as motifs

Value

hmo_base36 object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 15
dataset <- har_examples$example15
head(dataset)

# setting up motif discovery method
model <- hmo_base36(37, 3, 3)

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)
```

```
# filtering detected events
print(detection[(detection$event),])
```

---

hmo_mp *Motif discovery using Matrix Profile*

Description

Motif discovery using Matrix Profile doi:10.32614/RJ-2020-021

Usage

```
hmo_mp(mode = "stamp", w, qtd)
```

Arguments

mode	mode of computing distance between sequences. Available options include: "stomp", "stamp", "simple", "mstomp", "scrimp", "valmod", "pmp"
w	word size
qtd	number of occurrences to be classified as motifs

Value

hmo_mp object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 15
dataset <- har_examples$example15
head(dataset)

# setting up motif discovery method
model <- hmo_mp("stamp", 4, 3)

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

hmo_sax *Motif discovery using SAX*

Description

Motif discovery using SAX doi:10.1007/s10618-007-0064-z

Usage

```
hmo_sax(a, w, qtd)
```

Arguments

a	alphabet size
w	word size
qtd	number of occurrences to be classified as motifs

Value

hmo_sax object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples)

#Using example 15
dataset <- har_examples$example15
head(dataset)

# setting up motif discovery method
model <- hmo_sax(26, 3, 3)

# fitting the model
model <- fit(model, dataset$serie)

detection <- detect(model, dataset$serie)

# filtering detected events
print(detection[(detection$event),])
```

hmu_pca *Multivariate anomaly detector using PCA*

Description

Multivariate anomaly detector using PCA doi:10.1016/0098-3004(93)90090-R

Usage

```
hmu_pca()
```

Value

hmu_pca object

Examples

```
library(daltoolbox)

#loading the example database
data(har_examples_multi)

#Using the time series 9
dataset <- har_examples_multi$example1
head(dataset)

# establishing hmu_pca method
model <- hmu_pca()

# fitting the model using the two columns of the dataset
model <- fit(model, dataset[,1:2])

# making detections using hmu_pca
detection <- detect(model, dataset[,1:2])

# filtering detected events
print(detection[(detection$event),])

# evaluating the detections
evaluation <- evaluate(model, detection$event, dataset$event)
print(evaluation$confMatrix)
```

Index