Package ‘harbinger’

May 2, 2025

Title A Unified Time Series Event Detection Framework

Version 1.1.727

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

BugReports https://github.com/cefet-rj-dal/harbinger/issues

Encoding UTF-8

Depends R (>= 4.1.0)

RoxygenNote 7.3.2

Imports changepoint, daltoolbox, dplyr, dtwclust, forecast, ggplot2, hht, RcppHungarian, rugarch, stats, stringr, strucchange, tsmp, wavelets, zoo

NeedsCompilation no

Author Eduardo Ogasawara [aut, ths, cre] (ORCID: <https://orcid.org/0000-0002-0466-0626>),
Antonio Castro [aut],
Antonio Mello [aut],
Ellen Paixão [aut],
Fernando Fraga [aut],
Heraldo Borges [aut],
Janio Lima [aut],
Jessica Souza [aut],
Lais Baroni [aut],
Lucas Tavares [aut],
Michel Reis [aut],

Rebecca Salles [aut],
Diego Carvalho [aut],
Eduardo Bezerra [aut],
Rafaelli Coutinho [aut],
Esther Pacitti [aut],
Fabio Porto [aut],
Federal Center for Technological Education of Rio de Janeiro (CEFET/RJ) [cph] (organization: CEFET/RJ)

Maintainer Eduardo Ogasawara <eogasawara@ieee.org>

Repository CRAN

Date/Publication 2025-05-02 19:10:02 UTC

Contents

`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 examples of detectors for anomalies, change points, and motifs
#at https://cefet-rj-dal.github.io/harbinger
```

examples_anomalies — *Time series for anomaly detection*

Description

A list of time series for anomaly detection

- simple: a simple synthetic time series
- contextual: a contextual synthetic time series
- simple: a trend synthetic time series
- trend: a simple synthetic time series
- multiple: a multiple anomalies synthetic time series
- sequence: a sequence synthetic time series
- tt: a train-test synthetic time series
- tt_warped: a warped train-test synthetic time series

#'

Usage

```
data(examples_anomalies)
```

Format

A list of time series for anomaly detection.

Source

Harbinger package

References

Harbinger package

Examples

```
data(examples_anomalies)
serie <- examples_anomalies$simple
```

examples_changepoints *Time series for change point detection*

Description

A list of time series for change point

- simple: a simple synthetic time series
- sinusoidal: a sinusoidal synthetic time series
- incremental: a incremental synthetic time series
- abrupt: a abrupt synthetic time series
- volatility: a volatility synthetic time series

#'

Usage

```
data(examples_changepoints)
```

Format

A list of time series for change point detection.

Source

Harbinger package

References

Harbinger package

Examples

```
data(examples_changepoints)
serie <- examples_changepoints$simple
```

examples_harbinger *Time series for event detection*

Description

A list of time series for event detection

- nonstationarity: a synthetic nonstationarity time series
- global_temperature_yearly: yearly global temperature of the world
- global_temperature_monthly: monthly global temperature of the world
- multidimensional: multidimensional time series with a change point
- seattle_week: Seattle weakly temperature in 2019
- seattle_daily: Seattle daily temperature in 2019

#'

Usage

```
data(examples_harbinger)
```

Format

A list of time series.

Source

Harbinger package

References

Harbinger package

Examples

```
data(examples_harbinger)
serie <- examples_harbinger$seattle_daily
```

`examples_motifs` *Time series for change point detection*

Description

A list of time series for change point

- simple: a simple synthetic time series
- mitdb100: sample of mitdb 100 time series
- mitdb102: sample of mitdb 102 time series

#'

Usage

```
data(examples_motifs)
```

Format

A list of time series for motif discovery.

Source

Harbinger package

References

Harbinger package

Examples

```
data(examples_motifs)
serie <- examples_motifs$simple
```

`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(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
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(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
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, threshold = 0.5)
```

Arguments

model	DALToolbox classification model
threshold	threshold for classification

Value

`hanc_ml` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using example tt
dataset <- examples_anomalies$tt
dataset$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(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
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_emd — *Anomaly detector using EMD*

Description

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

Usage

```
hanr_emd(noise = 0.1, trials = 5)
```

Arguments

`noise`	nosie
`trials`	trials

Value

`hanr_emd` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series emd detector
model <- hanr_emd()

# 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(examples_anomalies)
```

```
#Using simple example
dataset <- examples_anomalies$simple
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_fft *Anomaly detector using FFT*

Description

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

Usage

```
hanr_fft()
```

Value

`hanr_fft` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series fft detector
model <- hanr_fft()

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

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

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

---

hanr_fft_amoc *Anomaly Detector using FFT with AMOC Cutoff*

Description

This function implements an anomaly detection method that uses the Fast Fourier Transform (FFT) combined with an automatic frequency cutoff strategy based on the AMOC (At Most One Change) algorithm. The model analyzes the power spectrum of the time series and detects the optimal cutoff frequency — the point where the frequency content significantly changes — using a changepoint detection method from the `changepoint` package.

All frequencies below the cutoff are removed from the spectrum, and the inverse FFT reconstructs a filtered version of the original signal that preserves only high-frequency components. The resulting residual signal is then analyzed to identify anomalous patterns based on its distance from the expected behavior.

This function extends the HARBINGER framework and returns an object of class `hanr_fft_amoc`.

Usage

```
hanr_fft_amoc()
```

Value

`hanr_fft_amoc` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series fft detector
model <- hanr_fft_amoc()

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

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

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

`hanr_fft_amoc_cusum` *Anomaly Detector using FFT with AMOC and CUSUM Cutoff*

Description

This function implements an anomaly detection method based on the Fast Fourier Transform (FFT) and a changepoint-based cutoff strategy using the AMOC (At Most One Change) algorithm applied to the cumulative sum (CUSUM) of the power spectrum.

The method first computes the FFT of the input time series and extracts the power spectrum. It then applies a CUSUM transformation to the spectral data to emphasize gradual changes or shifts in spectral energy. Using the AMOC algorithm, it detects a single changepoint in the CUSUM-transformed spectrum, which serves as a cutoff index to remove the lower-frequency components.

The remaining high-frequency components are then reconstructed into a time-domain signal via inverse FFT, effectively isolating rapid or local deviations. Anomalies are detected by evaluating the distance between this filtered signal and the original series, highlighting points that deviate significantly from the expected pattern.

This method is suitable for series where spectral shifts are subtle and a single significant change in behavior is expected.

This function extends the HARBINGER framework and returns an object of class `hanr_fft_amoc_cusum`.

Usage

```
hanr_fft_amoc_cusum()
```

Value

`hanr_fft_amoc_cusum` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series fft detector
model <- hanr_fft_amoc_cusum()
```

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

detection <- detect(model, dataset$serie)

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

---

hanr_fft_binseg *Anomaly Detector using FFT with Binary Segmentation Cutoff*

Description

This function implements an anomaly detection method that combines the Fast Fourier Transform (FFT) with a spectral cutoff strategy based on the Binary Segmentation (BinSeg) algorithm for changepoint detection.

The method analyzes the power spectrum of the input time series and applies the BinSeg algorithm to identify a changepoint in the spectral density, corresponding to a shift in the frequency content. Frequencies below this changepoint are considered part of the underlying trend or noise and are removed from the signal.

The modified spectrum is then transformed back into the time domain via inverse FFT, resulting in a high-pass filtered version of the series. Anomalies are identified by measuring the distance between the original and the filtered signal, highlighting unusual deviations from the dominant signal behavior.

This function is part of the HARBINGER framework and returns an object of class `hanr_fft_binseg`.

Usage

```
hanr_fft_binseg()
```

Value

`hanr_fft_binseg` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series fft detector
model <- hanr_fft_binseg()
```

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

detection <- detect(model, dataset$serie)

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

`hanr_fft_binseg_cusum` *Anomaly Detector using FFT with BinSeg and CUSUM Cutoff*

Description

This function implements an anomaly detection method that combines the Fast Fourier Transform (FFT) with a changepoint-based cutoff strategy using the Binary Segmentation (BinSeg) method applied to the cumulative sum (CUSUM) of the frequency spectrum.

The method first computes the FFT of the input time series and obtains its power spectrum. Then, it applies a CUSUM transformation to the spectral density to enhance detection of gradual transitions or accumulated changes in energy across frequencies. The Binary Segmentation method is applied to the CUSUM-transformed spectrum to identify a changepoint that defines a cutoff frequency.

Frequencies below this cutoff are removed from the spectrum, and the signal is reconstructed using the inverse FFT. This produces a filtered signal that retains only the high-frequency components, emphasizing potential anomalies.

Anomalies are then detected by measuring the deviation of the filtered signal from the original one, and applying an outlier detection mechanism based on this residual.

This function extends the HARBINGER framework and returns an object of class `hanr_fft_binseg_cusum`.

Usage

```
hanr_fft_binseg_cusum()
```

Value

`hanr_fft_binseg_cusum` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)
```

```
# setting up time series fft detector
model <- hanr_fft_binseg_cusum()

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

detection <- detect(model, dataset$serie)

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

`hanr_fft_sma` *Anomaly Detector using Adaptive FFT and Moving Average*

Description

This function implements an anomaly detection model based on the Fast Fourier Transform (FFT), combined with an adaptive moving average filter. The method estimates the dominant frequency in the input time series using spectral analysis and then applies a moving average filter with a window size derived from that frequency. This highlights high-frequency deviations, which are likely to be anomalies.

The residuals (original signal minus smoothed version) are then processed to compute the distance from the expected behavior, and points significantly distant are flagged as anomalies. The detection also includes a grouping strategy to reduce false positives by selecting the most representative point in a cluster of consecutive anomalies.

This function extends the HARBINGER framework and returns an object of class `hanr_fft_sma`.

Usage

```
hanr_fft_sma()
```

Value

`hanr_fft_sma` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)
```

```
# setting up time series fft detector
model <- hanr_fft_sma()

# 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(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
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(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
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(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
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),])
```

hanr_red *Anomaly and change point detector using RED*

Description

Anomaly and change point detection using RED The RED model adjusts to the time series. Observations distant from the model are labeled as anomalies. It wraps the EMD model presented in the hht library.

Usage

```
hanr_red(sw_size = 30, noise = 0.001, trials = 5)
```

Arguments

sw_size	sliding window size (default 30)
noise	noise
trials	trials

Value

hanr_red object

Examples

```
library(daltoolbox)
library(zoo)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series emd detector
model <- hanr_red()

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

detection <- detect(model, dataset$serie)

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

hanr_remd *Anomaly detector using REMD*

Description

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

Usage

```
hanr_remd(noise = 0.1, trials = 5)
```

Arguments

noise	nosie
trials	trials

Value

hanr_remd object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series emd detector
model <- hanr_remd()

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

detection <- detect(model, dataset$serie)

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

hanr_wavelet *Anomaly detector using Wavelet*

Description

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

Usage

```
hanr_wavelet(filter = "haar")
```

Arguments

filter	Availables wavelet filters: haar, d4, la8, bl14, c6

Value

hanr_wavelet object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series fft detector
model <- hanr_wavelet()

# 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, encoderclass = autoenc_ed, ...)
```

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.
`encoderclass`	The class of daltoolbox encoder-decoder.
`...`	optional arguments for encoder-decoder class.

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

```
#See an example of using `autoenc_ed` at this
#https://github.com/cefet-rj-dal/harbinger/blob/master/anomalies/han_autoenc_ed
```

`harbinger` *Harbinger*

Description

Ancestor class for time series event detection

Usage

```
harbinger()
```

Value

Harbinger object

Examples

```
#See examples of detectors for anomalies, change points, and motifs
#at https://cefet-rj-dal.github.io/harbinger
```

---

harutils *Harbinger Utils*

Description

Utility class that contains major distance measures, threshold limits, and outliers grouping functions

Usage

```
harutils()
```

Value

Harbinger Utils

Examples

```
# See ?hanc_ml for an example of anomaly detection using machine learning classification
```

---

har_ensemble *Harbinger Ensemble*

Description

Ensemble detector

Usage

```
har_ensemble(...)
```

Arguments

`...` list of detectors

Value

Harbinger object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using simple example
dataset <- examples_anomalies$simple
head(dataset)

# setting up time series emd detector
model <- har_ensemble(hanr_fbiad(), hanr_arima(), hanr_emd())

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

detection <- detect(model, dataset$serie)

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

---

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(examples_anomalies)

dataset <- examples_anomalies$simple
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(examples_anomalies)

#Using the simple
dataset <- examples_anomalies$simple
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_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 = NULL,
  event = NULL,
  mark.cp = TRUE,
  ylim = NULL,
  idx = NULL,
  pointsize = 0.5,
  colors = c("green", "blue", "red", "purple"),
  yline = NULL
)
```

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.
yline	values for plotting horizontal dashed lines

Value

A time series plot with marked events

Examples

```
library(daltoolbox)

#loading the example database
data(examples_anomalies)

#Using the simple time series
dataset <- examples_anomalies$simple
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)
```

---

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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
head(dataset)

# setting up change point method
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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
head(dataset)

# setting up change point method
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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
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(examples_changepoints)
```

```
#Using simple example
dataset <- examples_changepoints$simple
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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
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(examples_changepoints)

#Using volatility example
dataset <- examples_changepoints$volatility
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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
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_red — *Anomaly and change point detector using RED*

Description

Anomaly and change point detection using RED The RED model adjusts to the time series. Observations distant from the model are labeled as anomalies. It wraps the EMD model presented in the hht library.

Usage

```
hcp_red(
  sw_size = 30,
  noise = 0.001,
  trials = 5,
  red_cp = TRUE,
  volatility_cp = TRUE,
  trend_cp = TRUE
)
```

Arguments

`sw_size`	sliding window size (default 30)
`noise`	noise
`trials`	trials
`red_cp`	red change point
`volatility_cp`	volatility change point
`trend_cp`	trend change point

Value

`hcp_red` object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
head(dataset)

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

# 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(examples_changepoints)

#Using simple example
dataset <- examples_changepoints$simple
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),])
```

`hdis_mp` *Discord discovery using Matrix Profile*

Description

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

Usage

```
hdis_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 discords

Value

hdis_mp object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_motifs)

#Using sequence example
dataset <- examples_motifs$simple
head(dataset)

# setting up discord discovery method
model <- hdis_mp("stamp", 4, 3)

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

detection <- detect(model, dataset$serie)

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

---

hdis_sax *Discord discovery using SAX*

Description

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

Usage

```
hdis_sax(a, w, qtd = 2)
```

Arguments

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

Value

hdis_sax object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_motifs)

#Using sequence example
dataset <- examples_motifs$simple
head(dataset)

# setting up discord discovery method
model <- hdis_sax(26, 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(examples_motifs)

#Using sequence example
dataset <- examples_motifs$simple
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 = 2)
```

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(examples_motifs)

#Using sequence example
dataset <- examples_motifs$simple
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),])
```

---

hmo_xsax *Motif discovery using xsax*

Description

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

Usage

```
hmo_xsax(a, w, qtd)
```

Arguments

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

Value

hmo_xsax object

Examples

```
library(daltoolbox)

#loading the example database
data(examples_motifs)

#Using sequence example
dataset <- examples_motifs$simple
head(dataset)

# setting up motif discovery method
model <- hmo_xsax(37, 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(examples_harbinger)

#Using the time series 9
dataset <- examples_harbinger$multidimensional
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
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)
```

`mas` *Moving average smoothing*

Description

The `mas()` function returns a simple moving average smoother of the provided time series.

Usage

```
mas(x, order)
```

Arguments

`x`	A numeric vector or univariate time series.
`order`	Order of moving average smoother.

Details

The moving average smoother transformation is given by

$$(1/k) * (x[t] + x[t+1] + ... + x[t+k-1])$$

where `k=order`, `t` assume values in the range `1:(n-k+1)`, and `n=length(x)`. See also the `ma` of the `forecast` package.

Value

Numerical time series of length `length(x)-order+1` containing the simple moving average smoothed values.

References

R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

Examples

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

#Using simple example
dataset <- examples_changepoints$simple
head(dataset)

# setting up change point method
ma <- mas(dataset$serie, 5)
```

`trans_sax` *SAX*

Description

SAX

Usage

```
trans_sax(alpha)
```

Arguments

`alpha`	alphabet

Value

obj

Examples

```
library(daltoolbox)
vector <- 1:52
model <- trans_sax(alpha = 26)
model <- fit(model, vector)
xvector <- transform(model, vector)
print(xvector)
```

`trans_xsax` *XSAX*

Description

XSAX

Usage

```
trans_xsax(alpha)
```

Arguments

`alpha`	alphabet

Value

obj

Examples

```
library(daltoolbox)
vector <- 1:52
model <- trans_xsax(alpha = 52)
model <- fit(model, vector)
xvector <- transform(model, vector)
print(xvector)
```

Index