Package ‘eventstream’

July 22, 2025

Type Package

Title Streaming Events and their Early Classification

Version 0.1.1

Maintainer Sevvandi Kandanaarachchi <sevvandik@gmail.com>

Description Implements event extraction and early classification of events in data streams in R. It has the functionality to generate 2-dimensional data streams with events belonging to 2 classes. These events can be extracted and features computed. The event features extracted from incomplete-events can be classified using a partial-observations-classifier (Kandanaarachchi et al. 2018) <doi:10.1371/journal.pone.0236331>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports abind, tensorA, glmnet, dbscan, MASS, changepoint, dplyr

URL https://sevvandi.github.io/eventstream/index.html

RoxygenNote 7.1.2

Suggests knitr, rmarkdown

Depends R (>= 3.4.0)

NeedsCompilation no

Author Sevvandi Kandanaarachchi [aut, cre] (ORCID: <https://orcid.org/0000-0002-0337-0395>)

Repository CRAN

Date/Publication 2022-05-16 08:10:02 UTC

Contents

`extract_event_ftrs` *Extracts events from a data stream and computes event features.*

Description

This function extracts events from a 2D or 3D data stream and computes a set of 30 features for 2D streams and 13 features for 3D streams, by using a moving window. 2D data streams with class labels can be generated by using the function `gen_stream`. To get the class labels of the extracted events for the supervised setting, the event position is matched with the `details` of the events, which is part of the output of the `gen_stream` function.

Usage

```
extract_event_ftrs(
  stream,
  supervised = FALSE,
  details = NULL,
  win_size = 200,
  step_size = 20,
  thres = 0.95,
  folder = NULL,
  vis = FALSE,
  tt = 10,
  epsilon = 5,
  miniPts = 10,
```

```
  rolling = TRUE
)
```

Arguments

stream	A data stream. This can be the output of either the gen_stream function or the stream_from_files function.
supervised	If TRUE, event class labels need to be given in details.
details	Event details. This is also an output of the gen_stream function. Event details are used to get the class labels of the extracted events, by matching the position.
win_size	The window length of the moving window model, default is set to 200.
step_size	The window is moved by the step_size, default is 20.
thres	The cut-off quantile. Default is set to 0.95. Values greater than the quantile will be clustered. The rest is not clustered.
folder	If set to a local folder, this is where the jpegs of window data and extracted events are saved for a 2D data stream.
vis	If TRUE, the window data and the extracted events are plotted for a 2D data stream.
tt	Related to event ages. For example if tt=10 then the event ages are 10, 20, 30 and 40.
epsilon	The eps parameter in dbscan function in the package dbscan
miniPts	The minPts parameter in dbscan function in the package dbscan
rolling	This parameter is set to TRUE if rolling windows are considered.

Value

An Nx22x4 array is returned for 2D data streams and an Nx13x4 array for 3D data streams. Here N is the total number of events extracted from all windows. The second dimension has m features and the class label for the supervised setting. The third dimension has 4 different event ages : tt, 2tt, 3tt, 4tt. For example, the element at [10,6,3] has the 6th feature, of the 10th extracted event when the age of the event is 3tt. The features for 2D streams are listed below. For 3D streams the features cluster_id, pixels, length, width, height, total_value, l2w_ratio, centroid_x, centroid_y, centroid_z, mean, std_dev and sd_from_global_mean are computed.

cluster_id	An identification number for each event.
pixels	The number of pixels of each event.
length	The length of the event.
width	The width of the event.
total_value	The total value of the pixels.
l2w_ratio	Length to width ratio of event.
centroid_x	x coordinate of event centroid.
centroid_y	y coordinate of event centroid.
mean	Mean value of event pixels.

`std_dev` Standard deviation of event pixels.

`avg_slope` The slope of an `lm` object fitted to the event pixels.

`quad_1` The linear coefficient of a second order polynomial fitted to event pixels using `lm`.

`quad_2` The quadratic coefficient of a second order polynomial fitted to event pixels using `lm`.

`2sd_from_mean` The proportion of event pixels/cells that has values greater than 2 global standard deviations from the global mean of the window.

`3sd_from_mean` The proportion of event pixels/cells that has values greater than 3 global standard deviations from the global mean of the window.

`4sd_from_mean` The proportion of event pixels/cells that has values greater than 4 global standard deviations from the global mean of the window.

`5iqr_from_median` A small portion of each window and its column medians and column IQRs are used to construct two smoothing splines: a median spline and an IQR spline. The value of the median smoothing spline at each event centroid is used as the local median for that event. Similarly, the value of the IQR smoothing spline at each event centroid is used as the local IQR for that event. This feature gives the proportion of event pixels/cells that has values greater than 5 local IQRs from the local median.

`6iqr_from_median` The proportion of event pixels/cells that has values greater than 6 local IQRs from the local median computed using splines.

`7iqr_from_median` The proportion of event pixels/cells that has values greater than 7 local IQRs from the local median computed using splines.

`8iqr_from_median` The proportion of event pixels/cells that has values greater than 8 local IQRs from the local median computed using splines.

`iqr_from_median` Let us denote the 75th percentile of the event pixels value by `x`. How many local IQRs is `x` is away from the local median? Both local IQR and local median are computed using splines. That value is given by this feature.

`sd_from_mean` Let us denote the 80th percentile of the event pixels value by `x`. How many global standard deviations is `x` is away from the global mean? Here both global values are computed from window data.

Examples

```
# 2D data stream example
out <- gen_stream(1, sd=15)
zz <- as.matrix(out$data)
features <- extract_event_ftrs(zz, supervised=TRUE, details = out$details)
features

# 3D data stream example
```

```
set.seed(1)
arr <- array(rnorm(12000),dim=c(40,25,30))
arr[25:33,12:20, 20:23] <- 10
# getting events
ftrs <- extract_event_ftrs(arr, supervised=FALSE, win_size=10, step_size = 2, tt=2, thres=0.985)
ftrs
```

`gen_stream` *Generates a two dimensional data stream containing events of two classes.*

Description

This function generates a two-dimensional data stream containing events of two classes. The data stream can be saved as separate files with images by specifying the argument `folder`.

Usage

```
gen_stream(
  n,
  folder = NULL,
  sd = 1,
  vis = FALSE,
  muAB = c(4, 3),
  sdAB = c(2, 3)
)
```

Arguments

`n`	The number of files to generate. Each file consists of a 350x250 data matrix.
`folder`	If this is set to a local folder, the data matrices are saved in `folder/data`, the images are saved in `folder/pics` and the event details are saved in `folder/summary`. The event details are needed to obtain the class labels of events, when event extraction is done.
`sd`	This specifies the seed.
`vis`	If `TRUE`, the images are plotted.
`muAB`	The starting event pixels of class A and B events are normally distributed with mean values specified by `muAB`. The default is `c(4,3)`.
`sdAB`	The starting standard deviations of class A and B events. Default set to `c(2,3)`.

Details

There are events of two classes in the data matrices : A and B. Events of class A have only one shape while events of class B have three different shapes, including class A's shape. This was motivated from a real world example. The details of events of each class are given below.

Feature	class A	class B
Starting cell/pixel values	N(4,2)	N(3,3)
Ending cell/pixel values	N(8,2)	N(5,3)
Maximum age of event - shape 1	U(20,30)	U(20,30)
Maximum age of event - shape 2	NA	U(100,150)
Maximum age of event - shape 3	NA	U(100,150)
Maximum width of event - shape 1	U(20,26)	U(20,26)
Maximum width of event - shape 2	NA	U(30,38)
Maximum width of event - shape 3	NA	U(50,58)

Value

A list with following components:

`data` The data stream returned as a data frame.

`details` A data frame containing the details of the events: their positions, class labels, etc.. . This is needed for identifying class labels of events during event extraction.

`eventlabs` A matrix with 1 at event locations and 0 elsewhere.

See Also

`stream_from_files`.

Examples

```
out <- gen_stream(1, sd=15)
zz <- as.matrix(out$data)
image(1:nrow(zz), 1:ncol(zz),zz, xlab="Time", ylab="Location")
```

`get_clusters` *Extracts events from a two-dimensional data stream*

Description

This function extracts events from a two-dimensional (1 spatial x 1 time) data stream.

Usage

```
get_clusters(
  dat,
  filename = NULL,
  thres = 0.95,
  vis = FALSE,
  epsilon = 5,
  miniPts = 10,
  rolling = TRUE
)
```

Arguments

`dat`	The data matrix
`filename`	If set, the figure of extracted events are saved in this name. The `filename` needs to include the correct folder and file name.
`thres`	The cut-off quantile. Default is set to `0.95`. Values greater than the quantile will be clustered. The rest is not clustered.
`vis`	If `TRUE`, the window data and the extracted events are plotted for a 2D data stream.
`epsilon`	The `eps` parameter in `dbscan` function in the package `dbscan`
`miniPts`	The `minPts` parameter in `dbscan` function in the package `dbscan`
`rolling`	This parameter is set to `TRUE` if rolling windows are considered.

Value

A list with following components

`clusters`	The cluster assignment according to DBSCAN output.
`data`	The data of this cluster assignment.

Examples

```
out <- gen_stream(2, sd=15)
zz <- as.matrix(out$data)
clst <- get_clusters(zz, vis=TRUE)
```

`get_clusters_3d` *Extracts events from a three-dimensional data stream*

Description

This function extracts events from a three-dimensional (2D spatial x 1D time) data stream.

Usage

```
get_clusters_3d(dat, thres = 0.95, epsilon = 3, miniPts = 15)
```

Arguments

`dat`	The data matrix
`thres`	The cut-off quantile. Default is set to `0.95`. Values greater than the quantile will be clustered. The rest is not clustered.
`epsilon`	The `eps` parameter in `dbscan` function in the package `dbscan`
`miniPts`	The `minPts` parameter in `dbscan` function in the package `dbscan`

Value

A list with following components

clusters	The cluster assignment according to DBSCAN output.
data	The data of this cluster assignment.

Examples

```
set.seed(1)
arr <- array(rnorm(12000),dim=c(40,25,30))
arr[25:33,12:20, 20:23] <- 10
# getting events
out <- get_clusters_3d(arr, thres=0.985)
# plots
oldpar <- par(mfrow=c(1,3))
plot(out$data[,c(1,2)], xlab="x", ylab="y", col=as.factor(out$clusters$cluster))
plot(out$data[,c(1,3)], xlab="x", ylab="z",col=as.factor(out$clusters$cluster))
plot(out$data[,c(2,3)], xlab="y", ylab="z",col=as.factor(out$clusters$cluster))
par(oldpar)
```

---

get_features *Computes event-features*

Description

This function computes event features of 2D events.

Usage

```
get_features(
  dat.xyz,
  res.cluster,
  normal.stats.splines,
  win_size = 200,
  tt = 10
)
```

Arguments

dat.xyz	The data in a cluster friendly format. The first two columns have y and x positions with the third column having the pixel value of that position.
res.cluster	Cluster details from dbscan.
normal.stats.splines	The background statistics, output from spline_stats.
win_size	The window length of the moving window model, default is set to 200.
tt	Related to event ages. For example if tt=10 then the event ages are 10, 20, 30 and 40.

Value

An Nx22x4 array is returned for 2D data streams and an Nx13x4 array for 3D data streams. Here N is the total number of events extracted from all windows. The second dimension has m features and the class label for the supervised setting. The third dimension has 4 different event ages : tt, 2tt, 3tt, 4tt. For example, the element at [10,6,3] has the 6th feature, of the 10th extracted event when the age of the event is 3tt. The features for 2D streams are listed below. For 3D streams the features cluster_id, pixels, length, width, height, total_value, l2w_ratio, centroid_x, centroid_y, centroid_z, mean, std_dev and sd_from_global_mean are computed.

cluster_id	An identification number for each event.
pixels	The number of pixels of each event.
length	The length of the event.
width	The width of the event.
total_value	The total value of the pixels.
l2w_ratio	Length to width ratio of event.
centroid_x	x coordinate of event centroid.
centroid_y	y coordinate of event centroid.
mean	Mean value of event pixels.
std_dev	Standard deviation of event pixels.
avg_slope	The slope of an lm object fitted to the event pixels.
quad_1	The linear coefficient of a second order polynomial fitted to event pixels using lm.
quad_2	The quadratic coefficient of a second order polynomial fitted to event pixels using lm.
2sd_from_mean	The proportion of event pixels/cells that has values greater than 2 global standard deviations from the global mean of the window.
3sd_from_mean	The proportion of event pixels/cells that has values greater than 3 global standard deviations from the global mean of the window.
4sd_from_mean	The proportion of event pixels/cells that has values greater than 4 global standard deviations from the global mean of the window.
5iqr_from_median	A small portion of each window and its column medians and column IQRs are used to construct two smoothing splines: a median spline and an IQR spline. The value of the median smoothing spline at each event centroid is used as the local median for that event. Similarly, the value of the IQR smoothing spline at each event centroid is used as the local IQR for that event. This feature gives the proportion of event pixels/cells that has values greater than 5 local IQRs from the local median.
6iqr_from_median	The proportion of event pixels/cells that has values greater than 6 local IQRs from the local median computed using splines.
7iqr_from_median	The proportion of event pixels/cells that has values greater than 7 local IQRs from the local median computed using splines.

8iqr_from_median
: The proportion of event pixels/cells that has values greater than 8 local IQRs from the local median computed using splines.

iqr_from_median
: Let us denote the 75th percentile of the event pixels value by x. How many local IQRs is x is away from the local median? Both local IQR and local median are computed using splines. That value is given by this feature.

sd_from_mean
: Let us denote the 80th percentile of the event pixels value by x. How many global standard deviations is x is away from the global mean? Here both global values are computed from window data.

Examples

```
out <- gen_stream(1, sd=15)
zz <- as.matrix(out$data)
clst <- get_clusters(zz, vis=TRUE)
sstats <- spline_stats(zz[1:100,])
ftrs <- get_features(clst$data, clst$clusters$cluster, sstats)
```

---

get_features_3d — *Computes event-features*

Description

This function computes event features of 3D events.

Usage

```
get_features_3d(dat.xyz, res.cluster, normal.stats, win_size, tt)
```

Arguments

dat.xyz	The data in a cluster friendly format. The first three columns have t,x and y positions with the fourth column having the pixel value of that position.
res.cluster	Cluster details from dbscan.
normal.stats	The background statistics, output from stats_3d.
win_size	The window length of the moving window model.
tt	Related to event ages. For example if tt=10 then the event ages are 10, 20, 30 and 40.

Value

An Nx22x4 array is returned. Here N is the total number of events extracted in all windows. The second dimension has 30 features and the class label for the supervised setting. The third dimension has 4 different event ages : tt, 2tt, 3tt, 4tt. For example, the element at [10,6,3] has the 6th feature, of the 10th extracted event when the age of the event is 3tt. The features are listed below:

cluster_id	An identification number for each event.
pixels	The number of pixels of each event.
length	The length of the event.
width	The width of the event.
total_value	The total value of the pixels.
l2w_ratio	Length to width ratio of event.
centroid_x	x coordinate of event centroid.
centroid_y	y coordinate of event centroid.
centroid_z	z coordinate of event centroid.
mean	Mean value of event pixels.
std_dev	Standard deviation of event pixels.
slope	Slope of a linear model fitted to the event.
quad1	First coefficient of a quadratic model fitted to the event.
quad2	Second coefficient of a quadratic model fitted to the event.
sd_from_mean	Let us denote the 80th percentile of the event pixels value by x. How many standard deviations is x is away from the mean?

Examples

```
set.seed(1)
arr <- array(rnorm(12000),dim=c(40,25,30))
arr[25:33,12:20, 20:23] <- 10
# getting events
out <- get_clusters_3d(arr, thres=0.985)
mean_sd <- stats_3d(arr[1:20,1:6,1:8])
ftrs <- get_features_3d(out$data, out$cluster$cluster, mean_sd, win_size=40, tt=2 )
```

NO2_2010 *A dataset containing NO2 data for 2010*

Description

This dataset contains smoothed NO2 data from March to September 2010

Usage

```
NO2_2010
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each NO2_2010[t, ,] contains NO2 data for a given month with t=1 corresponding to March and t=7 corresponding to September

Dimensions 2,3 Each NO2_2010[,x, y] contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2011 *A dataset containing NO2 data for 2011*

Description

This dataset contains smoothed NO2 data from March to September 2011

Usage

```
NO2_2011
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2011[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2011[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2012 *A dataset containing NO2 data for 2012*

Description

This dataset contains smoothed NO2 data from March to September 2012

Usage

```
NO2_2012
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2012[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2012[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2013 *A dataset containing NO2 data for 2013*

Description

This dataset contains smoothed NO2 data from March to September 2013

Usage

`NO2_2013`

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2013[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2013[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2014 *A dataset containing NO2 data for 2014*

Description

This dataset contains smoothed NO2 data from March to September 2014

Usage

`NO2_2014`

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2014[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2014[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2015 *A dataset containing NO2 data for 2015*

Description

This dataset contains smoothed NO2 data from March to September 2015

Usage

```
NO2_2015
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2015[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2015[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2016 *A dataset containing NO2 data for 2016*

Description

This dataset contains smoothed NO2 data from March to September 2016

Usage

```
NO2_2016
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2016[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2016[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2017 *A dataset containing NO2 data for 2017*

Description

This dataset contains smoothed NO2 data from March to September 2017

Usage

```
NO2_2017
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2017[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2017[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2018 *A dataset containing NO2 data for 2018*

Description

This dataset contains smoothed NO2 data from March to September 2018

Usage

```
NO2_2018
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2018[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2018[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

NO2_2019 *A dataset containing NO2 data for 2019*

Description

This dataset contains smoothed NO2 data from March to September 2019

Usage

```
NO2_2019
```

Format

An array of 4 x 179 x 360 dimensions.

Dimension 1 Each `NO2_2019[t, , ]` contains NO2 data for a given month with `t=1` corresponding to March and `t=7` corresponding to September

Dimensions 2,3 Each `NO2_2019[ ,x, y]` contains NO2 concentration for a given position in the world map.

Source

https://neo.gsfc.nasa.gov/

---

predict_tdl *Prediction with incomplete-event-classifier*

Description

Predicts using the incomplete-event-classifier.

Usage

```
predict_tdl(model, t, X, probs = FALSE)
```

Arguments

`model`	The fitted incomplete-event-classifier.
`t`	The age of events.
`X`	The event features.
`probs`	If `TRUE`, probabilities are returned.

Value

The predicted values using the model object. If `prob = TRUE`, then the probabilities are returned.

Examples

```
# Generate data
N <- 1000
t <- sort(rep(1:10, N))
set.seed(821)
for(kk in 1:10){
  if(kk==1){
     X <- seq(-11,9,length=N)
  }else{
     temp <- seq((-11-kk+1),(9-kk+1),length=N)
     X <- c(X,temp)
  }
}
real.a.0 <- seq(2,20, by=2)
real.a.1 <- rep(2,10)
Zstar <-real.a.0[t] + real.a.1[t]*X + rlogis(N, scale=0.5)
Z <- 1*(Zstar > 0)

# Plot data for t=1 and t=8
oldpar <- par(mfrow=c(1,2))
plot(X[t==1],Z[t==1], main="t=1 data")
abline(v=-1, lty=2)
plot(X[t==8],Z[t==8],main="t=8 data")
abline(v=-8, lty=2)
par(oldpar)

# Fit model
train_inds <-  c()
for(i in 0:9){train_inds <- c(train_inds , i*N + 2*(1:499))}
model_td <- td_logistic(t[train_inds],X[train_inds],Z[train_inds])

# Prediction
preds <- predict_tdl(model_td,t[-train_inds],X[-train_inds] )
sum(preds==Z[-train_inds])/length(preds)
```

`real_details` *A dataset containing the details of class A events in the dataset real_stream.*

Description

This dataset contains the location of class A events in the real_stream dataset. This can be used for classifying the events in real_stream.

Usage

```
real_details
```

Format

A data frame with 4 rows and 3 variables:

filename Orignal file name

class class of event, A or B

file_x `y` coordinate of file, relating to the location of event

file_y `x` coordinate of file, relating to the start time of event

stream_x `x` coordinate of `real_stream`, relating to the start time of event

stream_y `y` coordinate of `real_stream`, relating to the location of event

---

`real_stream` *A data stream from a real world application*

Description

A dataset containing fibre optic cable signals. A pulse is periodically sent through the cable and this results in a data matrix where each horizontal row (`real_stream[x, ]`) gives the strength of the signal at a fixed location `x`, and each vertical column (`real_stream[ ,t]`) gives the strength of the signal along the cable at a fixed time `t`.

Usage

```
real_stream
```

Format

A matrix with 587 rows and 379 columns.

`spline_stats` *Computes background quantities using splines*

Description

This function computes 4 splines, from median, iqr, mean and standard deviation values.

Usage

```
spline_stats(dat)
```

Arguments

`dat`	The data matrix

Value

A list with following components

`med.spline`	The spline computed from the median values.
`iqr.spline`	The spline computed from IQR values.
`mean.spline`	The spline computed from mean values.
`sd.spline`	The spline computed from standard deviation values.
`mean.dat`	The mean of the data matrix.
`sd.dat`	The standard deviation of the data matrix.

Examples

```
out <- gen_stream(1, sd=15)
zz <- as.matrix(out$data)
sstats <- spline_stats(zz[1:100,])
oldpar <- par(mfrow=c(2,1))
image(1:ncol(zz), 1:nrow(zz),t(zz), xlab="Location", ylab="Time" )
plot(sstats[[1]], type="l")
par(oldpar)
```

`stats_3d` *Computes mean and standard deviation*

Description

This function is used for 3D event extraction and feature computation.

Usage

```
stats_3d(dat)
```

Arguments

`dat`	The data array

Value

A list with following components

`mean.dat`	The mean of the data array
`sd.dat`	The standard deviation of the data array

Examples

```
set.seed(1)
arr <- array(rnorm(12000),dim=c(40,25,30))
arr[25:33,12:20, 20:23] <- 10
mean_sd <- stats_3d(arr[1:20,1:6,1:8])
mean_sd
```

`stream_from_files` *Generates a two dimensional data stream from data files in a given folder.*

Description

Generates a two dimensional data stream from data files in a given folder.

Usage

```
stream_from_files(folder)
```

Arguments

`folder`	The folder with the data files.

See Also

`gen_stream`.

Examples

```
## Not run:
folder <- tempdir()
out <- gen_stream(2, folder = folder)
stream <- stream_from_files(paste(folder, "/data", sep=""))
dim(stream)
unlink(folder, recursive = TRUE)

## End(Not run)
```

---

td_logistic *Classification with incomplete-event-classifier*

Description

This function does classification of incomplete events. The events grow with time. The input vector `t` denotes the age of the event. The classifier takes the growing event features, `X` and combines with a `L2` penalty for smoothness.

Usage

```
td_logistic(
  t,
  X,
  Y,
  lambda = 1,
  scale = TRUE,
  num_bins = 4,
  quad = TRUE,
  interact = FALSE,
  logg = TRUE
)
```

Arguments

`t`	The age of events.
`X`	The event features.
`Y`	The class labels. `Y` needs to be binary output.
`lambda`	The penalty coefficient. Default is 1.
`scale`	If `TRUE`, each column of `X` is scaled to zero mean and standard deviation 1.
`num_bins`	The number of time slots to use.
`quad`	If `TRUE`, the squared attributes `X^2` are included.
`interact`	if `TRUE`, the most relevant interactions are included.
`logg`	If `TRUE` logarithms of positive attributes will be computed.

Value

A list with following components:

`par`	The parameters of the incomplete-event-classifier, after its fitted.
`convergence`	The difference between the final two output values.
`scale`	If `scale=TRUE`, contains the mean and the standard deviation of each column of `X`.

t	The age of events t is split into bins. This list element contains the boundary values of the bins.
quad	The value of quad in arguments.
interact	The value of interact in arguments.

See Also

predict_tdl for prediction.

Examples

```
# Generate data
N <- 1000
t <- sort(rep(1:10, N))
set.seed(821)
for(kk in 1:10){
  if(kk==1){
     X <- seq(-11,9,length=N)
  }else{
     temp <- seq((-11-kk+1),(9-kk+1),length=N)
     X <- c(X,temp)
  }
}
real.a.0 <- seq(2,20, by=2)
real.a.1 <- rep(2,10)
Zstar <-real.a.0[t] + real.a.1[t]*X + rlogis(N, scale=0.5)
Z <- 1*(Zstar > 0)

# Plot data for t=1 and t=8
oldpar <- par(mfrow=c(1,2))
plot(X[t==1],Z[t==1], main="t=1 data")
abline(v=-1, lty=2)
plot(X[t==8],Z[t==8],main="t=8 data")
abline(v=-8, lty=2)
par(oldpar)

# Fit model
model_td <- td_logistic(t,X,Z)
```

---

tune_cpdbee_2D *Tunes 2D event detection using labeled data*

Description

This function finds best parameters for 2D event detection using labeled data.

Usage

```
tune_cpdbee_2D(
  x,
  cl,
  alpha_min = 0.95,
  alpha_max = 0.98,
  alpha_step = 0.01,
  epsilon_min = 2,
  epsilon_max = 12,
  epsilon_step = 2,
  minPts_min = 4,
  minPts_max = 12,
  minPts_step = 2
)
```

Arguments

x	The data in an mxn matrix or dataframe.
cl	The actual locations of the events.
alpha_min	The minimum threshold value.
alpha_max	The maximum threshold value.
alpha_step	The incremental step size for alpha.
epsilon_min	The minimum epsilon value for DBSCAN clustering.
epsilon_max	The maximum epsilon value for DBSCAN clustering.
epsilon_step	The incremental step size for epsilon for DBSCAN clustering.
minPts_min	The minimum minPts value for for DBSCAN clustering.
minPts_max	The maximum minPts value for for DBSCAN clustering.
minPts_step	The incremental step size for minPts for DBSCAN clustering.

Value

A list with following components

best	The best threshold, epsilon and MinPts for 2D event detection and the associated Jaccard Index.
all	All parameter values used and the associated Jaccard Index values.

Examples

```
## Not run:
out <- gen_stream(1, sd=15)
zz <- as.matrix(out$data)
clst <- get_clusters(zz, filename = NULL, thres = 0.95,
                     vis = TRUE, epsilon = 5, miniPts = 10,
                     rolling = FALSE)
clst_loc <- clst$data[ ,1:2]
```

```
out <- tune_cpdbee_2D(zz, clst_loc)
out$best

## End(Not run)
```

---

tune_cpdbee_3D *Tunes 3D event detection using labeled data*

Description

This function finds best parameters for 3D event detection using labeled data.

Usage

```
tune_cpdbee_3D(
  x,
  cl,
  alpha_min = 0.95,
  alpha_max = 0.98,
  alpha_step = 0.01,
  epsilon_min = 2,
  epsilon_max = 12,
  epsilon_step = 2,
  minPts_min = 8,
  minPts_max = 16,
  minPts_step = 2
)
```

Arguments

x	The data in an mxn matrix or dataframe.
cl	The actual locations of the events.
alpha_min	The minimum threshold value.
alpha_max	The maximum threshold value.
alpha_step	The incremental step size for alpha.
epsilon_min	The minimum epsilon value for DBSCAN clustering.
epsilon_max	The maximum epsilon value for DBSCAN clustering.
epsilon_step	The incremental step size for epsilon for DBSCAN clustering.
minPts_min	The minimum minPts value for for DBSCAN clustering.
minPts_max	The maximum minPts value for for DBSCAN clustering.
minPts_step	The incremental step size for minPts for DBSCAN clustering.

Value

A list with following components

best
: The best threshold, epsilon and MinPts for 2D event detection and the associated Jaccard Index.

all
: All parameter values used and the associated Jaccard Index values.

Examples

```
## Not run:
set.seed(1)
arr <- array(rnorm(12000),dim=c(40,25,30))
arr[25:33,12:20, 20:23] <- 10
# Getting events
out <- get_clusters_3d(arr, thres=0.985)
out <- tune_cpdbee_3D(arr, out$data[ ,1:3])
out$best

## End(Not run)
```

Index