Package ‘EventDetectR’

October 12, 2022

Version 0.3.5
Title Event Detection Framework
Description Detect events in time-series data. Combines multiple well-known R packages like 'forecast' and 'neuralnet' to deliver an easily configurable tool for multivariate event detection.
Encoding UTF-8
LazyData yes
Type Package
ByteCompile TRUE

BugReports https://github.com/frehbach/EventDetectR/issues
URL https://github.com/frehbach/EventDetectR
Repository CRAN
Depends R (>= 3.1.0)
Imports imputeTS, forecast, ggplot2, gridExtra, neuralnet
Suggests testthat, utils, caret, e1071
License GPL-3
RoxygenNote 7.1.1
NeedsCompilation no

Author Margarita Rebolledo [aut],
Sowmya Chandrasekaran [aut, cre],
Frederik Rehbach [aut],
Steffen Moritz [aut] (<https://orcid.org/0000-0002-0085-1804>)

Maintainer Sowmya Chandrasekaran <sowzz.17@gmail.com>
Date/Publication 2020-10-10 15:50:02 UTC

R topics documented:

EventDetectR-package .................................................. 2
buildEDModel .......................................................... 2
detectEvents .......................................................... 4
EventDetectR-package

EventDetectR-package description

Description

Detect events/ anomalies in time-series data.

Details

The EventDetectR package enables detection of events/ anomalies in multivariate time-series data. It combines multiple well-known R packages like 'forecast, 'neuralnet' to deliver an easily configurable tool for event detection.

buildEDModel

build Event Detection Model

Description

Builds an event detection object (edObject) containing all models and configurations that are used to detect events in given data.

Usage

buildEDModel(
  x,
  dataPreparators = "ImputeTSInterpolation",
  dataPreparationControl = list(),
  buildModelAlgo = "ForecastETS",
  buildForecastModelControl = list(),
  buildNeuralNetModelControl = list(),
  postProcessors = "bedAlgo",
  postProcessorControl = list(),
  ignoreVarianceWarning = FALSE,
  oldModel = NULL
)
Arguments

- **x**: data.frame containing initial data on which the model will be fitted. Data should be free of events. The data should not include a timestamp column.

- **dataPreparators**: string or vector of strings, that defines which preparators to use. Lists are not accepted. Usage Example: dataPreparators = "ImputeTSInterpolation" results in the usage of imputeTS::na.interpolation as a data preparator. All possible preparators are listed via: getSupportedPreparations() Can also be set to NULL in order to shut off data preparation.

- **dataPreparationControl**: list, control-list containing all additional parameters that shall be passed to the dataPreparing.

- **buildModelAlgo**: string, model name to be used. All possible preparators are listed via: getSupportedModels().

- **buildForecastModelControl**: list, control-list containing all additional parameters that shall be passed to forecast modeling algorithm.

- **buildNeuralNetModelControl**: list, control-list containing all additional parameters that shall be passed to the neuralnet modeling algorithm.

- **postProcessors**: string or vector of strings, that defines which postProcessors to use. Lists are not accepted. Usage Example: postProcessors = "bedAlgo" results in the usage of bed as a event postprocessing tool. All possible preparators are listed via: getSupportedPostProcessors() Can also be set to NULL in order to shut off data postProcessing.

- **postProcessorControl**: list, control-list containing all additional parameters that shall be passed to the postProcessors.

- **ignoreVarianceWarning**: Ignores the continuously appearing warning for missing variance in some variable columns given a smaller windowSize.

- **oldModel**: If another model was previously fitted it can be passed to the next model fit. By doing so the eventHistory is preserved.

Value

model, event detection object (edObject) containing all models and configurations that are used to detect events in given data.

Examples

```r
## build a simple event detection model with standard configuration
x <- stationBData[100:200,-1]
buildEDModel(x, ignoreVarianceWarning = TRUE)

## Set up a more complex event detection model defining some additional configuration
buildEDModel(x, buildModelAlgo = "ForecastArima", ignoreVarianceWarning = TRUE)
```
## Set up a multivariate neural network model

`buildEDModel(x, buildModelAlgo = "NeuralNetwork", ignoreVarianceWarning = TRUE)`

---

**detectEvents**

**detectEvents in a given data.frame**

**Description**

detectEvents builds a prediction model (edObject) on the first `WindowSize` points of the given data `x`. The next `nIterationsRefit` data-points are classified as 'Event' or not. The window is moved iteratively and the next models are fitted. The first `WindowSize` points will always be classified as no Event and should only contain 'clean' data.

**Usage**

detectEvents(
  x,
  windowSize = 100,
  nIterationsRefit = 1,
  verbosityLevel = 0,
  dataPrepators = "ImputeTSInterpolation",
  dataPreparationControl = list(),
  buildModelAlgo = "ForecastETS",
  buildForecastModelControl = list(),
  buildNeuralNetModelControl = list(),
  postProcessors = "bedAlgo",
  postProcessorControl = list(),
  ignoreVarianceWarning = TRUE
)

**Arguments**

- **x**  
  data.frame, data which shall be classified as event or not

- **WindowSize**  
  amount of data points to consider in each prediction model

- **nIterationsRefit**  
  amount of points into the future which will be predicted without fitting a new model. E.g. if `nIterationsRefit = 10` then the next five dataPoints are classified without refitting.

- **verbosityLevel**  
  Print output of function progress. 0 -> No output, 1 -> every 100th model building iteration, 2 -> every 10th, 3 -> every iteration

- **dataPrepators**  
  string or vector of strings, that defines which preparators to use. Lists are not accepted. Usage Example: dataPrepators = "ImputeTSInterpolation" results in the usage of imputeTS::na.interpolation as a data preparator. All possible preparators are listed via: getSupportedPreparations()
detectEvents

dataPreparationControl
  list, control-list containing all additional parameters that shall be passed to the
dataPreparators.

buildModelAlgo
  string, model name to be used. All possible preparators are listed via: getSupportedModels().

buildForecastModelControl
  list, control-list containing all additional parameters that shall be passed to the
  forecast modelling algo.

buildNeuralNetModelControl
  list, control-list containing all additional parameters that shall be passed to the
  neuralnet modelling algo.

postProcessors
  string or vector of strings, that defines which postProcessors to use. Lists are
  not accepted. Usage Example: postProcessors = "bedAlgo" results in the usage
  of bed as a event postProcessing tool. All possible preparators are listed via:
  getSupportedPostProcessors()

postProcessorControl
  list, control-list containing all additional parameters that shall be passed to the
  postProcessirs.

ignoreVarianceWarning
  Ignores the continously appearing warning for missing variance in some variable
  columns given a smaller windowSize

Value

edsResults edObject, list of results. $classification -> data.frame containing the T/F event classification

Examples

## Run event detection with default settings:
def <- detectEvents(x = stationBData[1:100,-1])

## Refit the model at every new datapoint,
## have someoutput with verbosityLevel = 2 and ignore
## the variance warning
ed <- detectEvents(stationBData[1:110,-1],nIterationsRefit = 1,
  verbosityLevel = 2,ignoreVarianceWarning = TRUE)

## Switch to another model: Arima
ed2 <- detectEvents(stationBData[1:110,-1],nIterationsRefit = 1,
  verbosityLevel = 0,ignoreVarianceWarning = TRUE,
  buildModelAlgo = "ForecastArima")

## Switch to multivariate model: NeuralNetwork
ed3 <- detectEvents(stationBData[1:110,-1],nIterationsRefit = 1, buildModelAlgo = "NeuralNetwork")
getSupportedModels

Description


Description


Description

Get a list of all data modelling methods that are currently supported in package ‘eventDetectR’.

Usage

getSupportedModels()

Value

allSupportedModels a list of strings with each supported method name. The strings can be copied and used in calls to ‘eventDetect’ or ‘buildEDModel’

Examples

models <- getSupportedModels()
getSupportedPostProcessors

Description
Get a list of all data postprocessing methods that are currently supported in package 'eventDetectR'.

Usage
getSupportedPostProcessors()

Value
allSupportedPostProcessors a list of strings with each supported method name. The strings can be
copied and used in calls to 'eventDetect' or 'buildEDModel'

Examples
preps <- getSupportedPostProcessors()

getSupportedPreparations

Description
Get a list of all data preparation methods that are currently supported in package 'eventDetectR'.

Usage
getSupportedPreparations()

Value
allSupportedPreparations a list of strings with each supported method name. The strings can be
copied and used in calls to 'eventDetect' or 'buildEDModel'

Examples
preps <- getSupportedPreparations()
plot.edObject  
*Plot an Event Detection Object*

**Description**
Plot an Event Detection Object

**Usage**
```r
## S3 method for class 'edObject'
plot(x, varsToPlot = names(edObject$classification), ...)
```

**Arguments**
- `x`: edObject
- `varsToPlot`: vars
- `...`: Additional parameters

**Value**
A Plot

---

print.edObject  
*Print an Event Detection Object*

**Description**
Prints the last classification results for an event detection object. If `nLast` (integer) is given, it specifies the amount of rows to be printed.

**Usage**
```r
## S3 method for class 'edObject'
print(x, ...)```

**Arguments**
- `x`: edObject, the event detection object that shall be printed
- `...`: any additional parameters
**qualityStatistics**

**Description**

Wrapper function for caret::confusionMatrix. qualityStatistics calculates statistics for judging the quality of the eventDetection based on the fitted edModel and a reference dataset.

**Usage**

```r
qualityStatistics(edObject, reference)
```

**Arguments**

- `edObject`: The eventdetection object you obtain by running 'detectEvents'
- `reference`: true/false vector, reference vector based on labeled data: which datapoints are real events.

**Value**

list, Confusion Matrix and Statistics

**Examples**

```r
train <- geccoIC2018Train[15000:17000,]
edObject <- detectEvents(train[,-c(1,11)],windowSize = 1000,
nIterationsRefit = 500,verbosityLevel = 2,
postProcessorControl = list(nStandardDeviationseventThreshold = 3))
qualityStatistics(edObject, train$EVENT)
```

---

**simulateEvents**

*Imposes simulated events on the top of the data*

**Description**

Simulates Events on columns of a data frame or a matrix by applying different transformations. The events of type sinusoidal, square, binomial or ramp can be used.
Usage

simulateEvents(
  Data,
  Params,
  Event_type,
  Event_strength = NULL,
  Start_index = NULL,
  Event_duration = NULL,
  Percentage = NULL
)

Arguments

Data  Data frame or matrix containing the data to which the events will be introduced

Params  Numeric vector or vector of strings indicating the column names (in case Data is a data frame) or the column numbers (in case Data is a matrix) of the parameters in which an event will be simulated

Event_type  String vector indicating which type of transformation the parameters will undergo. Current valid options include sinusoidal, square, ramp and slowsinusoidal. If Params contains more than one element and Event_type only contains one element the same transformation will be applied to all given Params

Event_strength  (Optional) Numeric Vector indicating the amplitude. Only valid for sinusoidal and square transformations. When specified for other type of transformations it will have no effect. However it must have the same number of elements as Params.

Start_index  Numeric, indicates the index where the event should start

Event_duration  Numeric, indicates the number of steps the transformation should last. Default is 100

Percentage  (Optional) Numeric value from 0 to 1. Alternative input indicating the percentage of data that should be affected by the transformation. Either Event_duration or Percentage should be specified.

Value

Matrix or data frame containing the selected columns with simulated events

Examples

#Generate event of type sinusoidal and ramp on two columns of the stationBData data set
simupar<-c("B_PH_VAL", "B_TEMP_VAL")
SimulatedEvents<-simulateEvents(stationBData,
  simupar, Event_type = c("sinusoidal", "ramp"),
  Start_index = 2500)

#When specifying Event_strength the length of the vector needs to match the number of elements in Params.
SimulatedEvents<-simulateEvents(stationBData,
stationBData

```
simpar, Event_type = c("sinusoidal","ramp"),
Start_index = 2500,
Percentage = 0.2,
Event_strength = c(4,1))
```

---

**Description**

Data for package testing purposes
Index

* data
  geccoIC2018Test, 6
  geccoIC2018Train, 6
  stationBData, 11

buildEDModel, 2

detectEvents, 4

EventDetectR-package, 2

geccoIC2018Test, 6
geccoIC2018Train, 6
getSupportedModels, 6
getSupportedPostProcessors, 7
getSupportedPreparations, 7

plot.edObject, 8
print.edObject, 8

qualityStatistics, 9

simulateEvents, 9
stationBData, 11