Package ‘EpiSignalDetection’

November 30, 2021

Type Package

Title Signal Detection Analysis

Version 0.1.2

Date 2021-10-27

Description Exploring time series for signal detection. It is specifically designed to detect possible outbreaks using infectious disease surveillance data at the European Union / European Economic Area or country level. Automatic detection tools used are presented in the paper "Monitoring count time series in R: aberration detection in public health surveillance", by Salmon (2016) <doi:10.18637/jss.v070.i10>. The package includes:
- Signal Detection tool, an interactive 'shiny' application in which the user can import external data and perform basic signal detection analyses;
- An automated report in HTML format, presenting the results of the time series analysis in tables and graphs.
  This report can also be stratified by population characteristics (see 'Population' variable).
  This project was funded by the European Centre for Disease Prevention and Control.

Depends R (>= 3.4.0)

License EUPL

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports dplyr, graphics, ISOweek, rmarkdown, shiny, surveillance, utils

Suggests DT, ggplot2, knitr (>= 1.20), pander

VignetteBuilder knitr

SystemRequirements pandoc (>= 1.12.3)

URL https://github.com/EU-ECDC/EpiSignalDetection

NeedsCompilation no
**aggAtlasExport**

**Description**

Aggregate filtered final Atlas export

**Usage**

`aggAtlasExport(x, input)`

**Arguments**

- **x**  
  dataframe

- **input**  
  list of parameters as defined in the Signal Detection Application (see `runEpiSDApp`)  
  (i.e. `list(disease, country, indicator, stratification, unit, daterange, algo, testingperiod)`)  

**Value**

dataframe aggregated by geographical level and time unit
## AlgoParam

**See Also**

filterAtlasExport SignalData stsSD

### Examples

```r
#-- Setting the parameters to run the report for
input <- list(
  disease = "Salmonellosis",
  country = "EU-EEA - complete series",
  indicator = "Reported cases",
  stratification = "Confirmed cases",
  unit = "Month",
  daterange = c("2010-01-01", "2016-12-31"),
  algo = "FarringtonFlexible",
  testingperiod = 5
)

#-- Example dataset
dataset <- EpiSignalDetection::SignalData

#-- Filtering on declared input parameters
dataset <- filterAtlasExport(dataset, input)

#-- Aggregating the data by geographical level and time point
dataset <- aggAtlasExport(dataset, input)
```

<table>
<thead>
<tr>
<th>AlgoParam</th>
<th>List of datasets containing the Farrington Flexible and GLRNB default parameters by time unit</th>
</tr>
</thead>
</table>

### Description

A list including two datasets containing the parameters used for Farrington Flexible and for GLRNB for each time unit available in the Signal Detection tool.

### Usage

AlgoParam

### Format

A list of 2 dataframes: one with 2 rows and 9 variables and GRLNB with 2 rows and 8 variables

1. **Default parameters for FarringtonFlexible algorithm**
   - **timeunit**: Time units available in the signal detection tool i.e. week, month
   - **w**: Window’s half-size, i.e. number of weeks to include before and after the current week in each year (w=2 for weeks, w=1 for months)
**reweight** Logical specifying whether to reweight past outbreaks or not (TRUE for both weeks and months, past outbreaks are always reweighted)

**trend** Logical specifying whether a trend should be included and kept in case the conditions in the Farrington et. al. paper are met. (TRUE for both weeks and months, a trend is always fit)

**weightsThreshold** Numeric defining the threshold for reweighting past outbreaks using the Anscombe residuals (2.85 for both weeks and months, as advised in the improved method)

**glmWarnings** Logical specifying whether to print warnings from the call to glm (TRUE for both weeks and months)

**pThresholdTrend** Numeric defining the threshold for deciding whether to keep trend in the model (0.05 for both weeks and months)

**limit54_1** Integer, the number of cases defining a threshold for minimum alarm, no alarm is sounded if fewer than 'limit54_1' cases were reported in the past 'limit54_2' weeks/months

**limit54_2** Integer, the number of periods defining a threshold for minimum alarm, no alarm is sounded if fewer than 'limit54_1' cases were reported in the past 'limit54_2' weeks/months

2. **Default parameters for GLRNB algorithm**

**timeunit** Time units available in the signal detection tool i.e. week, month

**mu0** A vector of in-control values of the mean of the Poisson / negative binomial distribution with the same length as range - NULL for both weeks and months

**theta** Numeric, the pre-specified value for k or lambda is used in a recursive LR scheme - log(1.2) for both weeks and months corresponding to a 20 percent increase in the mean

**alpha** Numeric, the dispersion parameter of the negative binomial distribution. If alpha=NULL the parameter is calculated as part of the in-control estimation - alpha=NULL for both weeks and months

**cARL** Numeric, the threshold in the GLR test, i.e. c_gamma - cARL=0.25 for both weeks and months

**Mtilde** Integer, the number of observations needed before we have a full rank - Mtilde=1 for both weeks and months

**M** Integer defining the number of time instances back in time in the window-limited approach. To always look back until the first observation use M=-1. M=1 for both weeks and months

**Change** Character string specifying the type of the alternative. Currently the two choices are intercept and epi - Change=intercept for both weeks and months

See Also

surveillance::farringtonFlexible surveillance::glrnb

---

**algoSD**

Build algo object

---

**Description**

Build algo object from an sts object class using either FarringtonFlexible or GLRNB surveillance algorithm
Usage

algoSD(x.sts, algo = "FarringtonFlexible", timeUnit = "Month", testingPeriod = 5)

Arguments

x.sts
sts class object (see stsSD output)

galgo
character string containing the name of the algorithm to use. Options are "FarringtonFlexible" (default) or "GLRNB".

timeUnit
character string for the time unit of the time series. Options are "Week" or "Month".

testingPeriod
numeric: number of time units (months, weeks) back in time to test the algorithm on (to detect outbreaks in)

Value

sts

See Also

stsSD plotSD

Examples

#-- Setting the parameters to run the report for
input <- list(
disease = "Salmonellosis",
country = "EU-EEA - complete series",
indicator = "Reported cases",
stratification = "Confirmed cases",
unit = "Month",
daterange = c("2010-01-01", "2016-12-31"),
algo = "FarringtonFlexible",
testingperiod = 5
)

#-- Example dataset
dataset <- EpiSignalDetection::SignalData

#-- Filtering on declared input parameters
dataset <- filterAtlasExport(dataset, input)

#-- Aggregating the data by geographical level and time point
dataset <- aggAtlasExport(dataset, input)

#-- Building the corresponding sts object
dataset.sts <- stsSD(observedCases = dataset$NumValue,
    studyPeriod = dataset$StudyPeriod,
    timeUnit = input$unit,
    startYM = c(as.numeric(format(as.Date(input$daterange[1], "%Y-%m-%d"), "%Y")),
        as.numeric(format(as.Date(input$daterange[1], "%Y-%m-%d"), "%m"))))
cleanAtlasExport

#-- Building the corresponding algo object
dataset.algo <- algoSD(dataset.sts,
                        algo = input$algo,
                        timeUnit = input$unit,
                        testingPeriod =
                        input$testingperiod)

---

## cleanAtlasExport

**Clean the Atlas export dataframe**

### Description

Clean the Atlas data export dataframe before signal detection analysis

### Usage

```r
cleanAtlasExport(x)
```

### Arguments

- `x` dataframe, usually the output of the import function `importAtlasExport`

### Details

The function will:

- Filter only on case based indicators i.e. "Reported Cases"
- Create four additional time variables to ease the analysis:
  - `TimeUnit` ("Year", "Month", "Week")
  - `TimeYear` (xxxx)
  - `TimeMonth` (xx)
  - `TimeWeek` (xx)
- Keep only variables of interest i.e. "HealthTopic", "Population", "Time", "RegionName", "NumValue"

### Value

dataframe

### See Also

`importAtlasExport` `filterAtlasExport`
Examples

```r
## Not run:
dataset <- cleanAtlasExport( importAtlasExport(x = 'ECDC_surveillance_data_Anthrax.csv') )
## End(Not run)
```

---

**filterAtlasExport**

*Filter clean Atlas export*

**Description**

Filter clean Atlas export according to input parameters

**Usage**

```r
filterAtlasExport(x, input, stratified)
```

**Arguments**

- `x`: dataframe, clean Atlas export (see `cleanAtlasExport`)
- `input`: list of parameters as defined in the Signal Detection Application (see `runEpiSDApp`) (i.e. `list(disease,country,indicator,stratification,unit,daterange,algo,testingperiod)`)
- `stratified`: a logical value indicating whether the report should be stratified by Population variable or not (default FALSE)

**Value**

dataframe filtered on the selected parameters (input list)

**See Also**

`cleanAtlasExport`  `aggAtlasExport`

**Examples**

```r
#-- Setting the parameters to run the report for
input <- list(
disease = "Salmonellosis",
country = "EU-EEA - complete series",
indicator = "Reported cases",
stratification = "Confirmed cases",
unit = "Month",
daterange = c("2010-01-01", "2016-12-31"),
algo = "FarringtonFlexible",
testingperiod = 5
)
```
### importAtlasExport

**Import ECDC Atlas export file (csv)**

**Description**

Import ECDC Atlas csv export file
e.g. "ECDC_surveillance_data_Anthrax.csv"

**Usage**

`importAtlasExport(x)`

**Arguments**

- `x`  
  file name of a csv file, export from the ECDC Atlas  
  (e.g. `x = 'ECDC_surveillance_data_Anthrax.csv'`)

**Details**

The function will interpret missing reports `''` as NA values

**Value**

dataframe

**See Also**

`cleanAtlasExport`

**Examples**

```r
## Not run:
dataset <- importAtlasExport(x = 'ECDC_surveillance_data_Anthrax.csv')
## End(Not run)
```
plotSD

Plot the Signal Detection time series

Description
Plot the Signal Detection time series including historical data, alarm detection period and alarms

Usage
plotSD(x, input, subRegionName, x.sts, x.algo)

Arguments
x
dataframe (default SignalData)
input
list of parameters as defined in the Signal Detection Application (see runEpiSDApp) (i.e. list(disease,country,indicator,stratification,unit,daterange,algo,testingperiod))
subRegionName
character string, region label to use in the plot, if different than input$RegionName (optional)
x.sts
sts object (optional), see stsSD
x.algo
algo object (optional), see algoSD

Value
plot

See Also
SignalData runEpiSDApp

Examples
#-- Setting the parameters to run the report for
input <- list(
disease = "Salmonellosis",
country = "EU-EEA - complete series",
indicator = "Reported cases",
stratification = "Confirmed cases",
unit = "Month",
daterange = c("2010-01-01", "2016-12-31"),
algo = "FarringtonFlexible",
testingperiod = 5
)

#-- Plotting the signal detection output
plotSD(input = input)
runEpiSDApp  Run the EpiSignalDectection 'shiny' application

Description

Run the 'shiny' interactive application for signal detection analysis using ECDC Atlas export data.

Usage

runEpiSDApp()

Details

Datasets to use in the tool:

- Default dataset included in the application (Salmonellosis 2007-2016 or Measles 1999-2018 data);
- External dataset using the "Browse" button in the application:
  On the ECDC "Surveillance Atlas of Infectious Diseases" web site:
    * 1- Choose the disease/health topic to analyse
    * 2- Export the data (csv) using the default settings
    * 3- Import the csv in the application
    * 4- You can now explore the disease time series for signal detection...
  - Any dataset specified as described in the package vignette.

Examples

## Not run:
# --- Run the 'shiny' app
# --- (NB: please open the app in an external browser
# --- in order to facilitate its use)
runEpiSDApp()

## End(Not run)
**runEpiSDReport**  
*Run the EpiSignalDetection report (HTML markdown)*

### Description

Function to render the markdown report of alarms in HTML format for ECDC Signal Detection Report

### Usage

```r
runEpiSDReport(input, stratified, outputfile, run_pandoc = TRUE)
```

### Arguments

- **input**: list of parameters as defined in the Signal Detection Application (see `runEpiSDApp`)
  (i.e. list(disease, country, indicator, stratification, unit, daterange, algo, testingperiod)
  (see also default parameters in `system.file("SignalDetectionReport_HTML","SignalDetectionReport.Rmd",package = "EpiSignalDetection")`)

- **stratified**: a logical value indicating whether the report should be stratified by Population variable or not (default FALSE)

- **outputfile**: output file name (e.g. 'C:/R/report.html')
  (default value is a temporary folder: `file.path(tempdir(),"SignalDetectionReport.html")`)

- **run_pandoc**: An option for whether to run pandoc to convert Markdown output.

### Details

Datasets to use in the report:

- Default dataset included in the package (Salmonellosis 2007-2016 or Measles 1999-2018 data)
  (i.e. `input$file = NULL`);

- External dataset:
  - An export (csv format) from the ECDC Surveillance Atlas of Infectious Diseases:
  - On the ECDC "Surveillance Atlas of Infectious Diseases" web site:
    1. Choose the disease/health topic to analyse
    2. Export the data (csv) using the default settings
    3. Specify the location of this external dataset in the input argument of the `runEpiSDReport()` function (e.g. `input <- list(file = list(datapath = "C:/Users/Downloads/ECDC_surveillance_data_Pertussis.csv"),disease = "Pertussis",country = "Greece",indicator = "Reported cases",stratification = "All cases",unit = "Month",daterange = c("2011-12-01","2016-12-01"),algo = "FarringtonFlexible",testingperiod = 3))
    4. You can now render the re markdown report... (e.g. `runEpiSDReport(input = input))`
  - Any dataset specified as described in the package vignette.

---
Value

An HTML report. When `run_pandoc = TRUE`, the compiled document is written into the output file, and the path of the output file is returned. When `run_pandoc = FALSE`, the path of the Markdown output file, with attributes `knit_meta` (the `knitr` meta data collected from code chunks) and intermediates (the intermediate files/directories generated by `render()`) is returned.

See Also

Default dataset used in the report `SignalData`

Signal Detection Application `runEpiSDApp`

Examples

```r
## Not run:
#-- Running the report as a standalone function
runEpiSDReport()  # Definition of each input parameter
                  # is done one by one through the R console

#-- OR

#-- First setting the parameters to run the report for
input <- list(
  disease = "Salmonellosis",
  country = "Portugal",
  indicator = "Reported cases",
  stratification = "Confirmed cases",
  unit = "Month",
  daterange = c("2011-01-01", "2016-12-31"),
  algo = "FarringtonFlexible",
  testingperiod = 6
)

#-- Second running the report based on the EpiSignalDetection::SignalData dataset
#-- and store it in a temporary folder
runEpiSDReport(input = input)

#-- Running the report based on the EpiSignalDetection::SignalData dataset
#-- and store the HTML output 'test.html' in the folder 'C:/R/
runEpiSDReport(input = input, outputfile = "C:/R/test.html")

#-- Running the report based on external data
input <- list(
  file = list(datapath = "C:/Users/Downloads/ECDC_surveillance_data_Pertussis.csv"),
  disease = "Pertussis",
  country = "Greece",
  indicator = "Reported cases",
  stratification = "All cases",
  unit = "Month",
  daterange = c("2011-12-01", "2016-12-01"),
  algo = "FarringtonFlexible",
  testingperiod = 3
)```
runEpiSDReport(input = input, stratified = TRUE)

## End(Not run)

SignalData  

Dataset for Signal Detection Analysis, reported cases, 1999-2018  
(ECDC Atlas export)

Description

A dataset containing an export from the ECDC Atlas for salmonellosis and measles data. This export is cleaned and ready for Signal Detection Analysis (see. cleanAtlasExport() )

Usage

SignalData

Format

A data frame with 80,834 rows and 11 variables:

- **HealthTopic**: Disease name e.g. Salmonellosis or Measles
- **Population**: Population characteristics e.g. All cases, Confirmed cases, Serotype AGONA, Serotype BAREILLY etc.
- **Indicator**: Indicator e.g. Hospitalised cases, Reported cases, Number of deaths, etc.
- **Time**: Time variable including both yearly data from 1999 to 2017, and monthly data from 1999-01 to 2018-02
- **RegionName**: Geographical level including country names e.g. Austria, Belgium, Bulgaria, etc.
- **NumValue**: Number of cases
- **TimeUnit**: Time unit corresponding to the format of the date in the 'Time' variable e.g. Year or Month
- **TimeYear**: Year of the date available in the ‘Time’ variable, regardless of the date format i.e. 1999 to 2018
- **TimeMonth**: Month of the date available in the 'Time’ variable, regardless of the date format i.e. 1 to 12
- **TimeWeek**: Week of the date available in the 'Time’ variable, regardless of the date format i.e. NA since this dataset does not include any weekly data
- **TimeDate**: Approximated date corresponding to the date available in the 'Time’ variable (daily format)

Source

Description

Build sts surveillance object

Usage

```
stsSD(observedCases, studyPeriod, timeUnit = "Month", startYM = c(2000, 1) )
```

Arguments

- `observedCases`: numeric vector of the number of cases by time unit (y axis of the time series)
- `studyPeriod`: vector of dates of length(observedCases) (x axis of the time series)
- `timeUnit`: character string for the time unit of the time series. Options are Week or Month.
- `startYM`: numeric vector including Year and Month of start of the historical data

Value

- `sts`

See Also

- `aggAtlasExport`
- `algoSD`

Examples

```
#-- Setting the parameters to run the report for
input <- list( 
  disease = "Salmonellosis",
  country = "EU-EEA - complete series",
  indicator = "Reported cases",
  stratification = "Confirmed cases",
  unit = "Month",
  daterange = c("2010-01-01", "2016-12-31"),
  algo = "FarringtonFlexible",
  testingperiod = 5
)

#-- Example dataset
dataset <- EpiSignalDetection::SignalData

#-- Filtering on declared input parameters
dataset <- filterAtlasExport(dataset, input)

#-- Aggregating the data by geographical level and time point
dataset <- aggAtlasExport(dataset, input)
```
#-- Building the corresponding sts object

dataset.sts <- stsSD(observedCases = dataset$NumValue,
                   studyPeriod = dataset$StudyPeriod,
                   timeUnit = input$unit,
                   startYM = c(as.numeric(format(as.Date(input$daterange[1], "%Y-%m-%d"), "%Y")),
                               as.numeric(format(as.Date(input$daterange[2], "%Y-%m-%d"), "%m"))))

studyPeriod

Compute the study period

Description

Compute a dataframe including two types of dates corresponding to the study period defined in the
list of parameters input
(i.e. StudyPeriod = approximated daily date; Time = exact date in the format according to the time
unit parameter)

Usage

studyPeriod(input)

Arguments

input list of parameters as defined in the Signal Detection Application (see runEpiSDApp)
(i.e. list(disease,country,indicator,stratification,unit,daterange,algo,testingperiod)

Value

Dataframe including the complete time series with no gaps:

StudyPeriod approximated daily date e.g. 2010-01-01
Time exact date in the format according to the time unit parameter e.g. 2010-01

Examples

#-- Setting the parameters to run the report for
input <- list(
  disease = "Salmonellosis",
  country = "EU-EEA - complete series",
  indicator = "Reported cases",
  stratification = "Confirmed cases",
  unit = "Month",
  daterange = c("2010-01-01", "2016-12-31"),
  algo = "FarringtonFlexible",
  testingperiod = 5
)
studyPeriod

```r
StudyPeriod <- studyPeriod(input)
head(StudyPeriod)
```
Index

* datasets
  AlgoParam, 3
  SignalData, 13
  aggAtlasExport, 2, 7, 14
  AlgoParam, 3
  algoSD, 4, 9, 14
  cleanAtlasExport, 6, 7, 8
  filterAtlasExport, 3, 6, 7
  importAtlasExport, 6, 8
  plotSD, 5, 9
  runEpiSDApp, 2, 7, 9, 10, 11, 12, 15
  runEpiSDReport, 11
  SignalData, 3, 9, 12, 13
  stsSD, 3, 5, 9, 14
  studyPeriod, 15