Package 'epitweetr'

December 1, 2022

Title Early Detection of Public Health Threats from 'Twitter' Data

Version 2.2.13

Description It allows you to automatically monitor trends of tweets by time, place and topic aiming at detecting public health threats early through the detection of signals (e.g. an unusual increase in the number of tweets). It was designed to focus on infectious diseases, and it can be extended to all hazards or other fields of study by modifying the topics and keywords. More information is available in the 'epitweetr' peer-review publication (<https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2022.27.39.2200177>).

License EUPL

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

BugReports https://github.com/EU-ECDC/epitweetr/issues

Encoding UTF-8

Imports bit64, dplyr, crul, curl, DT, emayili, future, httpuv, httr, htmltools, jsonlite, keyring, knitr, lifecycle, ggplot2, janitor, magrittr, parallel, plotly, processx, rtweet, readxl, rgdal, rlang, rmarkdown, rnaturalearthdata, openxlsx, plyr, shiny, sp, stringr, stats, tibble, tidyverse, tidytext, tokenizers, tools, utils, xtable, xml2

RoxygenNote 7.2.0

Suggests taskscheduleR

VignetteBuilder knitr

NeedsCompilation no

Author Laura Espinosa [aut, fnd, cre] (<https://orcid.org/0000-0003-0748-9657>, Project manager, author of the design and concept of the package, and package maintainer), Francisco Orchard [aut, ctr] (<https://orcid.org/0000-0001-5793-3301>, Author of the package and original code), Ariana Wijermans [ctb] (Contributor to the design and concept of the package), Thomas Mollet [ctb, fnd] (Business owner of the project, and contributor to the design and concept of the package), Adrian Prodan [ctb],

Thomas Czernichow [ctb],
Maria Prieto Gonzalez [ctb],
Esther Kissling [ctb],
Michael Höhle [ctb]

Maintainer Laura Espinosa <laura.espinosa@ecdc.europa.eu>

Repository CRAN

Date/Publication 2022-12-01 00:40:03 UTC

R topics documented:

check_all . . . 3
create_map . . . 3
create_snapshot . . . 5
create_topchart . . . 6
create_topwords . . . 8
detect_loop . . . 9
download_dependencies . . . 11
ears_t_reweighted . . . 12
epitweetr_app . . . 13
fs_loop . . . 14
generate_alerts . . . 15
geolocate_text . . . 16
get_aggregates . . . 18
get_alerts . . . 19
get_tasks . . . 21
get_todays_sample_tweets . . . 22
health_check . . . 23
is_detect_running . . . 24
is_fs_running . . . 24
is_search_running . . . 25
json2lucene . . . 25
register_detect_runner_task . . . 26
register_fs_monitor . . . 27
register_fs_runner_task . . . 28
register_search_runner_task . . . 28
save_config . . . 29
search_loop . . . 30
search_tweets . . . 31
setup_config . . . 33
set_twitter_app_auth . . . 34
stop_detect_runner_task . . . 36
stop_fs_runner_task . . . 36
stop_search_runner_task . . . 37
trend_line . . . 38
update_geonames . . . 39
update_languages . . . 41

Index **43**

`check_all` *Run automatic sanity checks*

Description

It runs a set of automated sanity checks for helping the user to troubleshot issues

Usage

```
check_all()
```

Details

This function executes a series of sanity checks, concerning, Java, bitness, task status, dependencies and Twitter authentication.

Value

Data frame containing the statuses of all realized checks

Examples

```
if(FALSE){
   #importing epitweer
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   #running all tests
   check_all()
}
```

`create_map` *Plot the map report on the epitweetr dashboard*

Description

Generates a bubble map plot of number of tweets by countries, for one topic

Usage

```
create_map(
  topic = c(),
  countries = c(1),
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_retweets = FALSE,
  location_type = "tweet",
  caption = "",
  proj = NULL,
  forplotly = FALSE
)
```

Arguments

topic	Character(1) containing the topic to use for the report
countries	Character vector containing the name of the countries and regions to plot or their respective indexes on the Shiny app, default: c(1)
date_min	Date indicating start of the reporting period, default: "1900-01-01"
date_max	Date indicating end of the reporting period, default: "2100-01-01"
with_retweets	Logical value indicating whether to include retweets in the time series, default: FALSE
location_type	Character(1) vector indicating the location type. Possible values 'tweet', 'user' or 'both', default: 'tweet'
caption	Character(1) vector indicating a caption to print at the bottom of the chart, default: ""
proj	Parameter indicating the CRS (Coordinate Reference System) to use on PROJ4 format CRS-class? If null and all countries are selected +proj=robin is used (Robinson projection) otherwise the Lambert azimuthal equal-area projection will be chosen, default: NULL
forplotly	Logical(1) parameter indicating whether some hacks are activated to improve plotly rendering, default: FALSE

Details

Produces a bubble chart map for a particular topic on number of tweets collected based on the provided parameters. The map will display information at country level if more than one country is selected, otherwise it will display bubbles at the smallest possible location identified for each tweet within the period which could be any administrative level or city level.

Tweets associated with a country but with no finer granularity are omitted when displaying a single country.

When an aggregated zone is requested, all countries in that zone are included.

This functions requires that search_loop and detect_loop have already been run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the dataframe that was used to build the map.

See Also

trend_line create_topwords detect_loop search_loop spTransform,coordinates,is.projected,CRS-class fortify,geom_polygon,geom_point

Examples

```
if(FALSE){
   #Getting bubble chart for dengue for South America for last 30 days
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   create_map(
     topic = "dengue",
     countries = "South America",
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )
}
```

---

create_snapshot *Snapshot of your epitweetr installation*

Description

Creates a snapshot file of your epitweetr installation folder. This can include all or a subset of the data files.

Usage

```
create_snapshot(
  destination_dir,
  types = c("settings", "dependencies", "machine-learning", "aggregations", "tweets",
    "logs"),
  tweets_period = get_aggregated_period(),
  aggregated_period = get_aggregated_period(),
  compress = TRUE,
  progress = function(v, m) message(paste(round(v * 100, 2), m))
)
```

Arguments

destination_dir,
: character(1) vector with the path of the destination folder to produce the snapshot

types, character vector indicating the types of data to include on a snapshot. Some of: "settings", "dependencies", "machine-learning", "aggregations", "tweets", "logs"

tweets_period, date(2) start and end dates to filter tweets to include on snapshot (if selected)

aggregated_period, date(2) start and end dates to filter time series to include on snapshot (if selected)

compress, logical(1) whether to compress or not the output file

progress, function to report progress during execution.

Details

This function can be used to create a a portable file to move your epitweetr installation in a single file, to backup your data, to archive your old data or to send information to technical team in order to reproduce an observed issue. Different kinds of data can be included on the snapshot depending on the type of parameter. Possible values are: - 'settings': Including all setting files of your installation (excluding passwords) - 'dependencies': All jars and winutils.exe on windows installations - 'machine-learning': All trained models and vectors and training data (this can include tweet text which is personal data) - 'aggregations': Epitweetr aggregated time series - 'tweets': Tweets collected by epitweetr - 'logs': Log files produced automatically on windows task scheduler tasks.

Value

Nothing

Examples

```
if(FALSE){
   #importing epitweer
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   #creating a compressed snapshot for settings and logs
   create_snapshot(getwd(), c("settings","dependencies"), compress = TRUE)
}
```

---

create_topchart *Plot the top elements for a specific series on the epitweetr dashboard*

Description

Generates a bar plot of most popular elements in tweets, for one topic. Top elements among ("top-words", "hashtags", "entities", "contexts", "urls")

Usage

```
create_topchart(
  topic,
  serie,
  country_codes = c(),
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_retweets = FALSE,
  location_type = "tweet",
  top = 25
)
```

Arguments

topic	Character(1) containing the topic to use for the report
serie	Character(1) name of the series to be used for the report. It should be one of ("topwords", "hashtags", "entities", "contexts", "urls")
country_codes	Character vector containing the ISO 3166-1 alpha-2 countries to plot, default: c()
date_min	Date indicating start of the reporting period, default: "1900-01-01"
date_max	Date indicating end of the reporting period, default: "2100-01-01"
with_retweets	Logical value indicating whether to include retweets in the time series, default: FALSE
location_type	Character(1) this parameter is currently being IGNORED since this report shows only tweet location and cannot show user or both locations for performance reasons, default: 'tweet'
top	numeric(1) Parameter indicating the number of words to show, default: 25

Details

Produces a bar chart showing the occurrences of the most popular words in the collected tweets based on the provided parameters. For performance reasons on tweet aggregation, this report only shows tweet location and ignores the location_type parameter

This functions requires that search_loop and detect_loop have already been run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the data frame that was used to build the map.

See Also

trend_line create_map detect_loop search_loop

Examples

```
if(FALSE){
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   #Getting topword chart for dengue for France, Chile, Australia for last 30 days
   create_topchart(
     topic = "dengue",
     serie = "topwords",
     country_codes = c("FR", "CL", "AU"),
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )
 }
```

`create_topwords` *Plot the top words report on the epitweetr dashboard*

Description

Generates a bar plot of most popular words in tweets, for one topic

Usage

```
create_topwords(
  topic,
  country_codes = c(),
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_retweets = FALSE,
  location_type = "tweet",
  top = 25
)
```

Arguments

`topic`	Character(1) containing the topic to use for the report
`country_codes`	Character vector containing the ISO 3166-1 alpha-2 countries to plot, default: c()
`date_min`	Date indicating start of the reporting period, default: "1900-01-01"
`date_max`	Date indicating end of the reporting period, default: "2100-01-01"
`with_retweets`	Logical value indicating whether to include retweets in the time series, default: FALSE
`location_type`	Character(1) this parameter is currently being IGNORED since this report shows only tweet location and cannot show user or both locations for performance reasons, default: 'tweet'
`top`	numeric(1) Parameter indicating the number of words to show, default: 25

Details

Produces a bar chart showing the occurrences of the most popular words in the collected tweets based on the provided parameters. For performance reasons on tweet aggregation this report only shows tweet location and ignores the location_type parameter

This report may be empty for combinations of countries and topics with very few tweets since for performance reasons, the calculation of top words is an approximation using chunks of 10.000 tweets.

This function requires that search_loop and detect_loop have already been run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the data frame that was used to build the map.

See Also

trend_line create_map detect_loop search_loop

Examples

```
if(FALSE){
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   #Getting topword chart for dengue for France, Chile, Australia for last 30 days
   create_topwords(
     topic = "dengue",
     country_codes = c("FR", "CL", "AU"),
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )
 }
```

---

detect_loop *Runs the detect loop*

Description

Infinite loop ensuring the daily signal detection and email alerts

Usage

```
detect_loop(data_dir = NA)
```

Arguments

data_dir	Path to the 'data directory' containing application settings, models and collected tweets. If not provided the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA

Details

The detect loop is composed of three 'one shot tasks' download_dependencies, update_geonames, update_languages ensuring the system has all necessary components and data to run the three recurrent tasks generate_alerts

The loop report progress on the 'tasks.json' file which is read or created by this function.

The recurrent tasks are scheduled to be executed each 'detect span' minutes, which is a parameter set on the Shiny app.

If any of these tasks fails it will be retried three times before going to abort status. Aborted tasks can be relaunched from the Shiny app.

Value

nothing

See Also

download_dependencies

update_geonames

update_languages

detect_loop

generate_alerts

get_tasks

Examples

```
if(FALSE){
   #Running the detect loop
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   detect_loop()
}
```

download_dependencies *Updates Java dependencies*

Description

Download Java dependencies of the application mainly related to Apache SPARK and Lucene,

Usage

```
download_dependencies(tasks = get_tasks())
```

Arguments

tasks	Task object for reporting progress and error messages, default: get_tasks()

Details

Run a one-shot task consisting of downloading Java and Scala dependencies, this is separated by the following subtasks

- Download jar dependencies from configuration maven repo to project data folder. This includes, scala, spark, lucene. Packages to be downloaded are defined in package file 'sbtdeps.txt'
- Download winutils from configuration URL to project data folder. For more details on winutils please see https://issues.apache.org/jira/browse/HADOOP-13223 and https://issues.apache.org/jira/browse/HADOOP-16816

The URLs to download the JAR dependencies (maven package manager) and Winutils are on the configuration tab of the Shiny app.

Normally this function is not called directly by the user but from the detect_loop function.

Value

The list of tasks updated with produced messages

See Also

detect_loop

get_tasks

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
```

```
   # geolocating last tweets
   tasks <- download_dependencies()
}
```

`ears_t_reweighted` *algorithm for outbreak detection, extends the EARS algorithm*

Description

The simple 7 day running mean version of the Early Aberration Reporting System (EARS) algorithm is extended as follows:

- proper computation of the prediction interval
- downweighting of previous signals, similar to the approach by Farrington (1996)

Usage

```
ears_t_reweighted(
  ts,
  alpha = 0.025,
  alpha_outlier = 0.05,
  k_decay = 4,
  no_historic = 7L,
  same_weekday_baseline = FALSE
)
```

Arguments

`ts`	A numeric vector containing the counts of the univariate time series to monitor. The last time point in ts is investigated
`alpha`	The alpha is used to compute the upper limit of the prediction interval: (1-alpha) * 100%, default: 0.025
`alpha_outlier`	Residuals beyond 1-alpha_outlier quantile of the the t(n-k-1) distribution are downweighted, default: 0.05
`k_decay`	Power k in the expression (r_star/r_threshold)^k determining the weight, default: 4
`no_historic`	Number of previous values i.e -1, -2, ..., no_historic to include when computing baseline parameters, default: 7
`same_weekday_baseline`	whether to calculate baseline using same weekdays or any day, default: FALSE

Details

for algorithm details see package vignette.

Value

A dataframe containing the monitored time point, the upper limit and whether a signal is detected or not.

Author(s)

Michael Hoehle <https://www.math.su.se/~hoehle>

Examples

```
if(FALSE){
   library(epitweetr)
   #Running the modifies version of the ears algorithm for a particular data series
    ts <- c(150, 130, 122, 160, 155, 128, 144, 125, 300, 319, 289, 277, 500)
    show(ears_t_reweighted(ts))
}
```

`epitweetr_app` *Run the epitweetr Shiny app*

Description

Open the epitweetr Shiny app, used to setup the Data collection & processing pipeline, the Requirements & alerts pipeline and to visualise the outputs.

Usage

```
epitweetr_app(data_dir = NA)
```

Arguments

`data_dir`	Path to the 'data directory' containing application settings, models and collected tweets. If not provided the system will try to reuse the existing one from last session call of `setup_config` or use the EPI_HOME environment variable, default: NA

Details

The epitweetr app is the user entry point to the epitweetr package. This application will help the user to setup the tweet collection process, manage all settings, see the interactive dashboard visualisations, export them to Markdown or PDF, and setup the alert emails.

All its functionality is described on the epitweetr vignette.

Value

The Shiny server object containing the launched application

See Also

search_loop

detect_loop

Examples

```
if(FALSE){
   #Running the epitweetr app
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   epitweetr_app()
}
```

---

fs_loop *Runs the epitweetr embedded database loop*

Description

Infinite loop ensuring that the epitweetr embedded database is running (Lucene + akka-http)

Usage

```
fs_loop(data_dir = NA)
```

Arguments

data_dir	Path to the 'data directory' containing application settings, models and collected tweets. If not provided, the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA

Details

Launches the epitweetr embedded database which is accessed via a REST API located on http://localhost:8080. You can test that the database is running by accessing http://localhost:8080/ping the REST API provide epitweetr a way to send and retrieve data related with tweets and time series and to trigger geolocation or aggregation The database is implemented using Apache Lucene indexes allowing epitweetr to access its data as a search engine but also as a tabular database. health_check called each 60 seconds on a background process to send alerts to the administrator if some epitweetr components fail.

Value

nothing

See Also

detect_loop

search_loop

health_check

Examples

```
if(FALSE){
   #Running the detect loop
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   fs_loop()
}
```

---

generate_alerts *Execute the alert task*

Description

Evaluate alerts for the last collected day for all topics and regions and send email alerts to subscribers

Usage

```
generate_alerts(tasks = get_tasks())
```

Arguments

tasks	Current tasks for reporting purposes, default: get_tasks()

Details

This function calculates alerts for the last aggregated day and then send emails to subscribers.

The alert calculation is based on the country_counts time series which stores alerts by country, hour and topics.

For each country and region, the process starts by aggregating the last N days. A day is a block of consecutive 24 hours ending before the hour of the collected last tweet. N is defined by the alert baseline parameter on the configuration tab of the Shiny application (the default is N=7).

An alert will be produced when the number of tweets observed is above the threshold calculated by the modified version of the EARS algorithm (for more details see the package vignette). The behaviour of the alert detection algorithm is modified by the signal false positive rate (alpha), down-weighting of previous alerts and weekly or daily baseline parameters as defined on the configuration tab of the Shiny application and the topics file.

A prerequisite to this function is that the search_loop must already have stored collected tweets in the search folder and that the geotagging and aggregation tasks have already been run. Normally this function is not called directly by the user but from the detect_loop function.

Value

The list of tasks updated with produced messages

See Also

detect_loop

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())

   # calculating alerts for last day tweets and sending emails to subscribers
   generate_alerts()
}
```

---

geolocate_text *geolocate text in a data frame given a text column and optionally a language column*

Description

extracts geolocaion information on text on a column of the provided data frame and returns a new data frame with geolocation information

Usage

```
geolocate_text(df, text_col = "text", lang_col = NULL, min_score = NULL)
```

Arguments

df	A data frame containing at least character column with text, a column with the language name can be provided to improve geolocation quality
text_col	character, name of the column on the data frame containing the text to geolocalize, default:text
lang_col	character, name of the column on the data frame containing the language of texts, default: NULL
min_score,	numeric, the minimum score obtained on the Lucene scoring function to accept matches on GeoNames. It has to be empirically set default: NULL

Details

This function perform a call to the epitweetr database which includes functionality for geolocating for languages activated and successfully processed on the shiny app.

The geolocation process tries to find the best match in GeoNames database https://www.geonames.org/ including all local aliases for words.

If no language is associated to the text, all tokens will be sent as a query to the indexed GeoNames database.

If a language code is associated to the text and this language is trained on epitweetr, entity recognition techniques will be used to identify the best candidate in text to contain a location and only these tokens will be sent to the GeoNames query.

A custom scoring function is implemented to grant more weight to cities increasing with population to try to perform disambiguation.

Rules for forcing the geolocation choices of the algorithms and for tuning performance with manual annotations can be performed on the geotag tab of the Shiny app.

A prerequisite to this function is that the tasks download_dependencies update_geonames and update_languages has been run successfully.

This function is called from the Shiny app on geolocation evaluation tab but can also be used for manually evaluating the epitweetr geolocation algorithm.

Value

A new data frame containing the following geolocation columns: geo_code, geo_country_code, geo_country, geo_name, tags

See Also

download_dependencies

update_geonames

detect_loop

Examples

```
if(FALSE) {
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())

   # creating a test dataframe
  df <- data.frame(text = c("Me gusta Santiago de Chile es una linda ciudad"), lang = c("es"))
   geo <- geolocate_text(df = df, text_col = "text", lang_col="lang")

}
```

get_aggregates *Getting already aggregated time series produced by* detect_loop

Description

Read and returns the required aggregated dataset for the selected period and topics defined by the filter.

Usage

```
get_aggregates(
  dataset = "country_counts",
  cache = TRUE,
  filter = list(),
  top_field = NULL,
  top_freq = NULL
)
```

Arguments

dataset	A character(1) vector with the name of the series to request, it must be one of 'country_counts', 'geolocated', 'topwords', 'hashtags', 'entities', 'urls', 'contexts', default: 'country_counts'
cache	Whether to use the cache for lookup and storing the returned dataframe, default: TRUE
filter	A named list defining the filter to apply on the requested series, it should be on the shape of a named list e.g. list(tweet_geo_country_code=list('FR', 'DE')) default: list()
top_field	Name of the top field used with top_frequency to enable optimisation for getting only most frequent elements. It will only keep top 500 items after first 50k lines on reverse index order
top_freq	character, Name of the frequency fields used with top_field to enable optimisation for getting only most frequent elements. It will only keep top 500 items after first 50k rows on reverse index order

Details

This function returns data aggregated by epitweetr. The data is found on the 'series' folder, which contains Rds files per weekday and type of series. starting on v 1.0.x it will also look on Lucene indexes situated on fs folder. Names of files and folders are parsed to limit the files to be read. When using Lucene indexes, filters are directly applied on read. This is an improvement compared 'series' folder where filters are applied after read. All returned rows are joined in a single dataframe. If no filter is provided all data series is returned, which can end up with millions of rows depending on the time series. To limit by period, the filter list must have an element 'period' containing a date vector or list with two dates representing the start and end of the request.

To limit by topic, the filter list must have an element 'topic' containing a non-empty character vector or list with the names of the topics to return.

The available time series are:

- "country_counts" counting tweets and retweets by posted date, hour and country
- "geolocated" counting tweets and retweets by posted date and the smallest possible geolocated unit (city, administrative level or country)
- "topwords" counting tweets and retweets by posted date, country and the most popular words, (this excludes words used in the topic search)

The returned dataset can be cached for further calls if requested. Only one dataset per series is cached.

Value

A data frame containing the requested series for the requested period

See Also

detect_loop fs_loop

Examples

```
if(FALSE){
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   # Getting all country tweets between 2020-jan-10 and 2020-jan-31 for all topics
   df <- get_aggregates(
     dataset = "country_counts",
     filter = list(period = c("2020-01-10", "2020-01-31"))
   )

   # Getting all country tweets for the topic dengue
   df <- get_aggregates(dataset = "country_counts", filter = list(topic = "dengue"))

   # Getting all country tweets between 2020-jan-10 and 2020-jan-31 for the topic dengue
    df <- get_aggregates(
        dataset = "country_counts",
         filter = list(topic = "dengue", period = c("2020-01-10", "2020-01-31"))
    )
}
```

---

get_alerts *Getting signals produced by the task* generate_alerts *of* detect_loop

---

Description

Returns a data frame of signals produced by the detect_loop, which are stored on the signal folder.

Usage

```
get_alerts(
  topic = character(),
  countries = numeric(),
  from = "1900-01-01",
  until = "2100-01-01",
  toptweets = 0,
  limit = 0,
  duplicates = "all",
  progress = function(a, b) {
 }
)
```

Arguments

topic	Character vector. When it is not empty it will limit the returned signals to the provided topics, default: character()
countries	Character vector containing the names of countries or regions or a numeric vector containing the indexes of countries as displayed at the Shiny App to filter the signals to return., default: numeric()
from	Date defining the beginning of the period of signals to return, default: '1900-01-01'
until	Date defining the end of the period of signals to return, default: '2100-01-01'
toptweets	Integer number of top tweets to be added to the alert. These are obtained from the tweet index based on topwords and Lucene score, default: 0
limit	Maximum number of alerts returned, default: 0
duplicates	Character, action to decide what to do with alerts generated on the same day. Options are "all" (keep all alerts), "first" get only first alert and "last" for getting only the last alert
progress	Function, function to report progress it should receive two parameter a progress between 0 and 1 and a message, default: empty function

Details

For more details see the package vignette.

Value

a data frame containing the calculated alerts for the period. If no alerts are found then NULL is returned

See Also

generate_alerts

detect_loop

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())

   # Getting signals produced for last 30 days for a particular country
   get_alerts(
     countries = c("Chile", "Australia", "France"),
     from = as.Date(Sys.time())-30,
     until = as.Date(Sys.time())
   )
}
```

`get_tasks` *Get the* `detect_loop` *task status*

Description

Reads the status of the `detect_loop` tasks and updates it with changes requested by the Shiny app

Usage

```
get_tasks(statuses = list())
```

Arguments

`statuses`	Character vector for limiting the status of the returned tasks, default: list()

Details

After reading the tasks.json file and parsing it with jsonlite, this function will update the necessary fields in the tasks for executing and monitoring them.

Value

A named list containing all necessary information to run and monitor the detect loop tasks.

See Also

`download_dependencies`

`update_geonames`

`update_languages`

`detect_loop`

`generate_alerts`

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   tasks <- get_tasks()
}
```

`get_todays_sample_tweets` *Get a sample of latest tweet geolocations (deprecated)*

Description

This function was removed from epitweetr v1.0.1. Please use search_tweets instead

Usage

```
get_todays_sample_tweets(limit = 1000, text_col = "text", lang_col = "lang")
```

Arguments

`limit`	Size of the sample, default: 100
`text_col`	Name of the tweet field to geolocate it should be one of the following ("text", "linked_text", "user_description", "user_location", "place_full_name", "linked_place_full_name"), default: 'text'
`lang_col`	Name of the tweet variable containing the language to evaluate. It should be one of the following ("lang", "linked_lang", NA), default: "lang"

Details

This function was removed from epitweetr v1.0.1. Please use search_tweets instead.

Value

Data frame containing the sampled tweets and the geolocation metrics

See Also

`download_dependencies`

`update_geonames`

`update_languages`

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())

   # geolocating today's tweets
   show(get_todays_sample_tweets())
}
```

`health_check` *Send email to administrator if a failure of epitweetr is detected*

Description

It validates if epitweetr is not collecting tweets, aggregating tweets or not calculating alerts

Usage

```
health_check(send_mail = TRUE, one_per_day = TRUE)
```

Arguments

`send_mail`	Boolean. Whether an email should be sent to the defined administrator, default: TRUE
`one_per_day`	Boolean. Whether a limit of one email per day will be applied, default: TRUE

Details

This function sends an email to the defined administrator if epitweetr is not collecting tweets, aggregating tweets or not calculating alerts

Value

A list of health check errors found

Examples

```
if(FALSE){
   #importing epitweetr
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
  #sending the email to the administrator if epitweetr components are not properly working
   health_check()
}
```

is_detect_running *Check whether the alert detection task is running*

Description

gets the alert detection runner execution status

Usage

```
is_detect_running()
```

Details

returns a logical value being TRUE if the alert detection task is actually running

Value

logical Whether the alert detection task is running

Examples

```
if(FALSE){
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   is_detect_running()
}
```

is_fs_running *Check whether the database is running*

Description

gets the database runner execution status

Usage

```
is_fs_running()
```

Details

returns a logical value being TRUE if the database runner is actually running

Value

logical Whether the database is running

Examples

```
if(FALSE){
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   is_fs_running()
}
```

`is_search_running` *Check whether the tweet collection task is running*

Description

gets the tweet collection execution status

Usage

```
is_search_running()
```

Details

returns a logical value being TRUE if the tweet collection is running

Value

logical Whether the tweet collection is running

Examples

```
if(FALSE){
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   is_search_running()
}
```

`json2lucene` *Function used for migrating tweets from to old to the new file system*

Description

migrates geolocated tweets from the old to the new file system allowing full text search using Apache Lucene Indexes

Usage

```
json2lucene(tasks = get_tasks(), chunk_size = 400)
```

Arguments

tasks	named list, current tasks for logging and updating progress default: get_tasks()
chunk_size,	integer, the chunk size for indexing tweets, default: 400

Details

This function can be called manually to perform the migration of tweets between v0.0.x to v2+ It iterates over existing tweets collected with epitweetr v0.0.x series joins base tweets and geolocated tweets and then sends themes to the Lucene index via the dedicated REST API. Migrated files will be moved to search_archive and geo_archive folders. Users can backup and remove these folders when migration ends to gain disk space. Series folders are maintained for migrated tweets

Value

the updated tasks.

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   # runnint the migration
   json2lucene()
}
```

---

register_detect_runner_task

Registers the alert detection task

Description

registers the alert detection task or stops if no configuration has been set or if it is already running

Usage

```
register_detect_runner_task()
```

Details

Registers the alert detection task or stops if no configuration has been set or if it is already running. To generate alerts, this task needs the tweet collection to had successfully run since the last time it ran. This function will use the task scheduler on windows and will fall back to launching the runner as a separate process (attached to this session) on Linux.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   register_detect_runner_task()
}
```

`register_fs_monitor` *Registers the fs_monitor for the current process or exits*

Description

registers the fs_monitor (by writing fs.monitor.PID file) for the current process or stops if no configuration has been set or if it is already running

Usage

```
register_fs_monitor()
```

Details

Registers the fs_monitor (by writing fs.monitor.PID file) for the current process or stops if no configuration has been set or if it is already running This function has been exported so it can be properly called from the future package on the database runner, but it is not intended to be directly called by end users.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   register_fs_monitor()
}
```

register_fs_runner_task *Registers the epitweetr database task*

Description

registers the epitweetr database task or stops if no configuration has been set or if it is already running

Usage

```
register_fs_runner_task()
```

Details

Registers the epitweetr database task or stops if no configuration has been set or if it is already running. This task need the dependencies, geonames and languages steps to have been successfully ran. This can be done on the shiny app configuration page or by manually running the detect_runner_task. This function will try to use the task scheduler on windows and will fall back to launching the runner as a separate process (attached to this session) on Linux.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   register_fs_runner_task()
}
```

register_search_runner_task *Registers the tweet collection task*

Description

registers the tweet collection task or stops if no configuration has been set or if it is already running

Usage

```
register_search_runner_task()
```

Details

Registers the tweet collextion task or stops if no configuration has been set or if it is already running. Twitter authentication needs to be previously set on the shiny app or by calling set_twitter_app_auth(). You can test if authentication is working on the shiny app troubleshot page or by calling (with dplyr): epitweetr::check_all() This function will use the task scheduler on windows and will fall back to launching the runner as a separate process (attached to this session) on Linux.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   register_search_runner_task()
}
```

---

save_config *Save the configuration changes*

Description

Permanently saves configuration changes to the data folder (excluding Twitter credentials, but not SMTP credentials)

Usage

```
save_config(data_dir = conf$data_dir, properties = TRUE, topics = TRUE)
```

Arguments

data_dir	Path to a directory to save configuration settings, Default: conf$data_dir
properties	Whether to save the general properties to the properties.json file, default: TRUE
topics	Whether to save topic download plans to the topics.json file, default: TRUE

Details

Permanently saves configuration changes to the data folder (excluding Twitter credentials, but not SMTP credentials) to save Twitter credentials please use set_twitter_app_auth

Value

Nothing

See Also

setup_config set_twitter_app_auth

Examples

```
if(FALSE){
   library(epitweetr)
   #load configuration
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   #make some changes
   #conf$collect_span = 90
   #saving changes
   save_config()
}
```

---

search_loop *Runs the search loop*

Description

Infinite loop ensuring the permanent collection of tweets

Usage

```
search_loop(data_dir = NA)
```

Arguments

data_dir	optional path to the 'data directory' containing application settings, models and collected tweets. If not provided it will reuse the last set on the current session. If not provided the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, Default: NA

Details

The detect loop is a pure R function designed for downloading tweets from the Twitter search API. It can handle several topics ensuring that all of them will be downloaded fairly using a round-robin philosophy and respecting Twitter API rate-limits.

The progress of this task is reported on the 'topics.json' file which is read or created by this function. This function will try to collect tweets respecting a 'collect_span' window in minutes, which is defined on the Shiny app and defaults to 60 minutes.

To see more details about the collection algorithm please see epitweetr vignette.

In order to work, this task needs Twitter credentials, which can be set on the Shiny app or using set_twitter_app_auth

Value

Nothing

See Also

set_twitter_app_auth

Examples

```
if(FALSE){
   #Running the search loop
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   search_loop(file.choose())
}
```

---

search_tweets *perform full text search on tweets collected with epitweetr*

Description

perform full text search on tweets collected with epitweetr (tweets migrated from epitweetr v<1.0.x are also included)

Usage

```
search_tweets(
  query = NULL,
  topic = NULL,
  from = NULL,
  to = NULL,
  countries = NULL,
  mentioning = NULL,
  users = NULL,
  hide_users = FALSE,
  action = NULL,
  max = 100,
  by_relevance = FALSE
)
```

Arguments

query	character. The query to be used if a text it will match the tweet text. To see how to match particular fields please see details, default:NULL
topic	character, Vector of topics to include on the search default:NULL
from	an object which can be converted to '"POSIXlt"' only tweets posted after or on this date will be included, default:NULL

to	an object which can be converted to '"POSIXlt"' only tweets posted before or on this date will be included, default:NULL
countries	character or numeric, the position or name of epitweetr regions to be included on the query, default:NULL
mentioning	character, limit the search to the tweets mentioning the given users, default:NULL
users	character, limit the search to the tweets created by the provided users, default:NULL
hide_users	logical, whether to hide user names on output replacing them by the USER keyword, default:FALSE
action	character, an action to be performed on the search results respecting the max parameter. Possible values are 'delete' or 'anonymise' , default:NULL
max	integer, maximum number of tweets to be included on the search, default:100
by_relevance	logical, whether to sort the results by relevance of the matching query or by indexing order, default:FALSE If not provided the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA

Details

epitweetr translate the query provided by all parameters into a single query that will be executed on tweet indexes which are weekly indexes. The q parameter should respect the syntax of the Lucene classic parser https://lucene.apache.org/core/8_5_0/queryparser/org/apache/lucene/queryparser/classic/QueryParser.html So other than the provided parameters, multi field queries are supported by using the syntax field_name:value1;value2 AND, OR and -(for excluding terms) are supported on q parameter. Order by week is always applied before relevance so even if you provide by_relevance = TRUE all of the matching tweets of the first week will be returned first

Value

a data frame containing the tweets matching the selected filters, the data frame contains the following columns: linked_user_location, linked_user_name, linked_user_description, screen_name, created_date, is_geo_located, user_location_loc, is_retweet, text, text_loc, user_id, hash, user_description, linked_lang, linked_screen_name, user_location, totalCount, created_at, topic_tweet_id, topic, lang, user_name, linked_text, tweet_id, linked_text_loc, hashtags, user_description_loc

See Also

search_loop

detect_loop

Examples

```
if(FALSE){
   #Running the detect loop
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   df <- search_tweets(
```

```
        q = "vaccination",
        topic="COVID-19",
        countries=c("Chile", "Australia", "France"),
        from = Sys.Date(),
        to = Sys.Date()
    )
    df$text
}
```

`setup_config` *Load epitweetr application settings*

Description

Load epitweetr application settings from the designated data directory

Usage

```
setup_config(
  data_dir = if (exists("data_dir", where = conf)) conf$data_dir else if
   (Sys.getenv("EPI_HOME") != "") Sys.getenv("EPI_HOME") else file.path(getwd(),
    "epitweetr"),
  ignore_keyring = FALSE,
  ignore_properties = FALSE,
  ignore_topics = FALSE,
  save_first = list()
)
```

Arguments

`data_dir`	Path to the directory containing the application settings (it must exist). If not provided it takes the value of the latest call to setup_config in the current session, or the value of the EPI_HOME environment variable or epitweetr subdirectory in the working directory, default: if (exists("data_dir", where = conf)) conf$data_dir else if (Sys.getenv("EPI_HOME") != "") Sys.getenv("EPI_HOME") else file.path(getwd(), "epitweetr")
`ignore_keyring`	Whether to skip loading settings from the keyring (Twitter and SMTP credentials), default: FALSE
`ignore_properties`	Whether to skip loading settings managed by the Shiny app in properties.json file, Default: FALSE
`ignore_topics`	Whether to skip loading settings defined in the topics.xlsx file and download plans from topics.json file, default: FALSE
`save_first`	Whether to save current settings before loading new ones from disk, default: list()

Details

epitweetr relies on settings and data stored in a system folder, so before loading the dashboard, collecting tweets or detecting alerts the user has to designate this folder. When a user wants to use epitweetr from the R console they will need to call this function for initialisation. The 'data_folder' can also be given as a parameter for program launch functions `epitweetr_app`, `search_loop` or `detect_loop`, which will internally call this function.

This call will fill (or refresh) a package scoped environment 'conf' that will store the settings. Settings stored in conf are:

- General properties of the Shiny app (stored in properties.json)
- Download plans from the Twitter collection process (stored in topics.json merged with data from the topics.xlsx file
- Credentials for Twitter API and SMTP stored in the defined keyring

When calling this function and the keyring is locked, a password will be prompted to unlock the keyring. This behaviour can be changed by setting the environment variable 'ecdc_twitter_tool_kr_password' with the password.

Changes made to conf can be stored permanently (except for 'data_dir') using:

- `save_config`, or
- `set_twitter_app_auth`

Value

Nothing

See Also

`save_config` `set_twitter_app_auth` `epitweetr_app` `search_loop` `detect_loop`

Examples

```
if(FALSE){
   library(epitweetr)
   #loading system settings
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
}
```

---

set_twitter_app_auth *Save Twitter App credentials*

Description

Update Twitter authentication tokens in a configuration object

Usage

```
set_twitter_app_auth(
  app = "",
  access_token = "",
  access_token_secret = "",
  api_key = "",
  api_secret = "",
  bearer = ""
)
```

Arguments

app	Application name
access_token	Access token as provided by Twitter
access_token_secret	Access token secret as provided by Twitter
api_key	API key as provided by Twitter
api_secret	API secret as provided by Twitter
bearer	the bearer token of the application

Details

Update Twitter authentication tokens in configuration object

Value

Nothing

See Also

save_config

Examples

```
if(FALSE){
 #Setting the configuration values
   set_twitter_app_auth(
     app = "my super app",
     access_token = "123456",
     access_token_secret = "123456",
     api_key = "123456",
     api_secret = "123456"
   )
   set_twitter_app_auth(
     bearer = "123456"
   )
}
```

stop_detect_runner_task *Stops the alert detection task*

Description

stops the alert detection task

Usage

```
stop_detect_runner_task()
```

Details

Stops the alert detection task if it is already running This function will try also deactivate the respective scheduled task on Windows.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   stop_detect_runner_task()
}
```

stop_fs_runner_task *Stops the epitweetr database task*

Description

stops the epitweetr database task

Usage

```
stop_fs_runner_task()
```

Details

Stops the epitweetr database task if it is already running This function will try also deactivate the respective scheduled task on Windows.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   stop_fs_runner_task()
}
```

`stop_search_runner_task`

Stops the tweet collection task

Description

stops the tweet collection task

Usage

```
stop_search_runner_task()
```

Details

Stops the tweet collection task if it is already running This function will try also deactivate the respective scheduled task on Windows.

Value

Nothing

Examples

```
if(FALSE){
   #getting tasks statuses
   library(epitweetr)
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   stop_search_runner_task()
}
```

`trend_line` *Plot the trendline report of epitweetr dashboard*

Description

Generates a trendline chart of number of tweets by region, for one topic, including alerts using the reweighted version of the EARS algorithm

Usage

```
trend_line(
  topic,
  countries = c(1),
  date_type = "created_date",
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_retweets = FALSE,
  location_type = "tweet",
  alpha = 0.025,
  alpha_outlier = 0.05,
  k_decay = 4,
  no_historic = 7,
  bonferroni_correction = FALSE,
  same_weekday_baseline = FALSE
)
```

Arguments

`topic`	Character(1) containing the topic to use for the report
`countries`	Character vector containing the name of the countries and regions to plot or their respective indexes on the Shiny app select, default: c(1)
`date_type`	Character vector specifying the time granularity of the report either 'created_weeknum' or 'created_date', default: 'created_date'
`date_min`	Date indicating start of the reporting period, default: "1900-01-01"
`date_max`	Date indicating end of the reporting period, default: "2100-01-01"
`with_retweets`	Logical value indicating whether to include retweets in the time series, default: FALSE
`location_type`	Character(1) vector indicating the location type. Possible values 'tweet', 'user' or 'both', default: 'tweet'
`alpha`	Numeric(1) value indicating the alert detection confidence, default: 0.025
`alpha_outlier`	Numeric(1) value indicating the outliers detection confidence for downweighting, default: 0.05
`k_decay`	Strength of outliers downweighting, default: 4
`no_historic`	Number of observations to build the baseline for signal detection, default: 7

bonferroni_correction
: Logical value indicating whether to apply the Bonferroni correction for signal detection, default: FALSE

same_weekday_baseline
: Logical value indicating whether to use same day of weeks for building the baseline or consecutive days, default: FALSE

Details

Produces a multi-region line chart for a particular topic of number of tweets collected based on the provided parameters. Alerts will be calculated using a modified version of the EARS algorithm that applies a Farrington inspired downweighting of previous outliers.

Days in this function are considered as contiguous blocks of 24 hours starting for the previous hour of the last collected tweet.

This function requires search_loop and detect_loop to have already run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the data frame that was used to build the chart.

See Also

create_map create_topwords generate_alerts detect_loop search_loop

Examples

```
if(FALSE){
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())
   #Getting trendline for dengue for South America for the last 30 days
   trend_line(
     topic = "dengue",
     countries = "South America",
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )
}
```

---

update_geonames *Updates the local copy of the GeoNames database*

Description

Downloading and indexing a fresh version of the GeoNames database from the provided URL

Usage

```
update_geonames(tasks = get_tasks())
```

Arguments

tasks	Tasks object for reporting progress and error messages, default: get_tasks()

Details

Run a one shot task to download and index a local copy of the GeoNames database. The GeoNames geographical database covers all countries and contains over eleven million place names that are available; Creative Commons Attribution 4.0 License.

The URL to download the database from is set on the configuration tab of the Shiny app, in case it changes.

The indexing is developed in Spark and Lucene

A prerequisite to this function is that the search_loop must already have stored collected tweets in the search folder and that the task download_dependencies has been successfully run.

Normally this function is not called directly by the user but from the detect_loop function.

Value

The list of tasks updated with produced messages

See Also

download_dependencies

detect_loop

get_tasks

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())

   # geolocating last tweets
   tasks <- update_geonames()
}
```

`update_languages` *Updates local copies of languages*

Description

Downloading and indexing a fresh version of language models tagged for update on the Shiny app configuration tab

Usage

```
update_languages(tasks = get_tasks())
```

Arguments

`tasks`	Tasks object for reporting progress and error messages, default: get_tasks()

Details

Run a one shot task to download and index a local fasttext pretrained models. A fasttext model is a collection of vectors for a language automatically produced scrolling a big corpus of text that can be used to capture the semantic of a word.

The URL to download the vectors from are set on the configuration tab of the Shiny app.

This task will also update SVM models to predict whether a word is a location that will be used in the geolocation process.

The indexing is developed in SPARK and Lucene.

A prerequisite to this function is that the `search_loop` must already have stored collected tweets in the search folder and that the tasks `download_dependencies` and `update_geonames` has been run successfully.

Normally this function is not called directly by the user but from the `detect_loop` function.

Value

The list of tasks updated with produced messages

See Also

`download_dependencies`

`update_geonames`

`detect_loop`

`get_tasks`

Examples

```
if(FALSE){
   library(epitweetr)
   # setting up the data folder
   message('Please choose the epitweetr data directory')
   setup_config(file.choose())

   # updating language tasks
   tasks <- update_languages()
}
```

Index

check_all, 3
coordinates, *5*
create_map, 3, *7*, *9*, *39*
create_snapshot, 5
create_topchart, 6
create_topwords, *5*, 8, *39*

detect_loop, *4, 5*, *7*, *9*, 9, *10, 11*, *14–21*, *32*, *34*, *39–41*
download_dependencies, *10*, 11, *17*, *21, 22*, *40, 41*

ears_t_reweighted, 12
epitweetr_app, 13, *34*

fortify, *5*
fs_loop, 14, *19*

generate_alerts, *10*, 15, *19–21*, *39*
geolocate_text, 16
geom_point, *5*
geom_polygon, *5*
get_aggregates, 18
get_alerts, 19
get_tasks, *10, 11*, 21, *40, 41*
get_todays_sample_tweets, 22

health_check, *14, 15*, 23

is.projected, *5*
is_detect_running, 24
is_fs_running, 24
is_search_running, 25

json2lucene, 25

register_detect_runner_task, 26
register_fs_monitor, 27
register_fs_runner_task, 28
register_search_runner_task, 28

save_config, 29, *34, 35*
search_loop, *4, 5*, *7*, *9*, *14, 15*, 30, *32*, *34*, *39–41*
search_tweets, 31
set_twitter_app_auth, *29–31*, *34*, 34
setup_config, *10*, *13, 14*, *30*, *32*, 33
spTransform, *5*
stop_detect_runner_task, 36
stop_fs_runner_task, 36
stop_search_runner_task, 37

trend_line, *5*, *7*, *9*, 38

update_geonames, *10*, *17*, *21, 22*, 39, *41*
update_languages, *10*, *17*, *21, 22*, 41