Package ‘epitweetr’

October 1, 2020

Title Early Detection of Public Health Threats from Twitter Data

Version 0.1.20

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.

License EUPL

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

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

Encoding UTF-8

LazyData true

Imports bit64, dplyr, plyr, DT, httpuv, httr, jsonlite, keyring, emayili, ggplot2, magrittr, parallel, plotly, rtweet, readxl, rgeos, rgdal, rmarkdown, rnaturalearthdata, shiny, sp, stringr, stats, tidyverse, tidytext, tokenizers, tools, utils, xtable, xml2

RoxygenNote 7.1.0

Suggests knitr, taskscheduleR

VignetteBuilder knitr

NeedsCompilation no

Author Francisco Orchard [aut, ctr] (<https://orcid.org/0000-0001-5793-3301>, Author of the package and original code),
Laura Espinosa [aut, cre, fnd] (Project manager, author of the design and concept of the package, and package maintainer),
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 2020-10-01 08:30:09 UTC

R topics documented:

`aggregate_tweets` *Execute the aggregation task*

Description

Get all the tweets from the Twitter Standard Search API json files and the geolocated tweets json files obtained by calling (`geotag_tweets`) and store the results in the series folder as daily Rds files

Usage

```
aggregate_tweets(
  series = list("country_counts", "geolocated", "topwords"),
  tasks = get_tasks()
)
```

Arguments

`series`	List of series to aggregate, default: list("country_counts", "geolocated", "topwords")
`tasks`	Current tasks for reporting purposes, default: get_tasks()

Details

This function will launch a SPARK task of aggregating data collected from the Twitter Search API and geolocated from geotag tweets. By doing the following steps: - Identify the last aggregates date by looking into the series folder

- Look for date range of tweets collected since that day by looking at the stat json files produced by the search loop

- For each day that has to be updated a list of all geolocated and search files to load will be produced by looking at the stat files

- For each series passed as a parameter and for each date to update:

- a Spark task will be called that will deduplicate tweets for each topic, join them with gelocation information, and aggregate them to the required level and return to the standard output as json lines

- the result of this task is parsed using jsonlite and saved into RDS files in the series folder

A prerequisite to this function is that the `search_loop` must have already collected tweets in the search folder and that geotag_tweets has already run. Normally this function is not called directly by the user but from the `detect_loop` function.

Value

the list of tasks updated with aggregate messages

See Also

`detect_loop`

`geotag_tweets`

`generate_alerts`

Examples

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

   # aggregating all geolocated tweets collected since last aggregation for producing
   # all time series
   aggregate_tweets()
}
```

`check_all` *Run automatic sanity checks*

Description

run a set of automated sanity checks for helping the user to troubleqhot issues

Usage

```
check_all()
```

Details

This function executes a series of sanity checks, concerninr, java, bitness, task statusn dependencies and Twitter authentication.

Value

Dataframe 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 (Corrdinate 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 aggregate_tweets geotag_tweets 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_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 showed 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 chat showing the occurrences of the most popular words in the collected tweets based on the provided parameters. For performance reasons on the `aggregate_tweets` function, 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 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_map` `aggregate_tweets` `geotag_tweets` `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, `geotag_tweets`, `aggregate_tweets`, `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 a abort status. Aborted tasks can be relauched from the Shiny app.

Value

nothing

See Also

`download_dependencies`

`update_geonames`

`update_languages`

`detect_loop`

`geotag_tweets`

`aggregate_tweets`

`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 'sbt-deps.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 Höhle <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 search loop, the detect loop 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()
}
```

---

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), downweighting 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`

`geotag_tweets`

`aggregate_tweets`

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 subscriptors
   generate_alerts()
}
```

`geotag_tweets` *Launches the geo-tagging loop*

Description

This function will geolocate all tweets before the current hour that have not been already geolocated

Usage

```
geotag_tweets(tasks = get_tasks())
```

Arguments

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

Details

Geolocates tweets by collection date, and stores the result in the tweets/geolocated folder. It starts from the last geolocated date until the last collected tweet. When running on a day that has been partially geolocated, it will ignore tweets that have already been processed.

The geolocation is applied to several fields of tweets: text, original text (if retweet or quote), user description, user declared location, user biography, API location. For each field it will perform the following steps:

- Evaluate the part of the text which is more likely to be a location using an unsupervised machine learning and language dependent model trained during `update_languages`

- Match the selected text against a Lucene index of GeoNames database built during update_geonames
- Return the location with the highest matching score. For more information about the scoring process please refer to the epitweetr vignette

This algorithm has mainly been developed in Spark.

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, update_geonames and update_languages have successfully 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

download_dependencies

update_geonames

update_languages

detect_loop

aggregate_tweets

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 <- geotag_tweets()
}
```

get_aggregates *Getting already aggregated time series produced by detect_loop*

Description

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())
```

Arguments

`dataset`	A character(1) vector with the name of the series to request, it must be one of 'country_counts', 'geolocated' or 'topwords', 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, default: list()

Details

This function will look in the 'series' folder, which contains Rds files per weekday and type of series. It will parse the names of file and folders to limit the files to be read. Then it will apply the filters on each dataset for finally joining the matching results in a single dataframe. If no filter is provided all data series are 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, adminitrative 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 dataframe containing the requested series for the requested period

See Also

`detect_loop` `geotag_tweets`

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 dataframe 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"
)
```

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'

Details

For more details see the package vignette.

Value

a dataframe 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

geotag_tweets

aggregate_tweets

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

Description

Get a sample of today's tweets for evaluation of geolocation threshold

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 will take a sample of tweets collected on the current date for testing the geolocation algorithm and giving the user the possibility to evaluate the optimal score.

In order for this function to work the search loop will have had to run on the current day and the tasks download_dependencies, update_geonames and update_languages will have had to successfully been run.

Value

Dataframe containing the sampled tweets and the geolocation metrics

See Also

`download_dependencies`

`update_geonames`

`update_languages`

`geotag_tweets`

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())
}
```

`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`	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 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())
}
```

`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 enviroment variable 'ecdc_wtitter_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
)
```

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

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"
   )
}
```

---

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 dataframe that was used to build the chart.

See Also

`create_map` `create_topwords` `generate_alerts` `aggregate_tweets` `geotag_tweets` `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)
```

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)
```

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())

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

Index