Package ‘tweet2r’

July 11, 2018

Version 1.1

Date 2018-07-10

Title Twitter Collector for R and Export to 'SQLite', 'postGIS' and 'GIS' Format

Description This is an improved implementation of the package 'StreamR' to capture tweets and store it into R, SQLite, 'postGIS' data base or GIS format. The package performs a description of harvested data and performs space time exploratory analysis.

Author Pau Aragó, Pablo Juan, Jeroen Staab.

Depends R (>= 3.2),

Imports ROAuth, streamR, RPostgreSQL, rgdal, sp, RSQLite, ggmap, ggplot2, plyr, spacetime, spatstat, splancs, maptools

Maintainer Pau Aragó <parago@uji.es>

License GPL-3 | file LICENSE

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-07-11 12:20:03 UTC

R topics documented:

tweet2r-package 2
t2DataFrame 2
t2gis 3
t2pgis 4
t2SpatialPointDataFrame 5
t2sqlite 6
t2STIDF 7
t2summary 8
tglm 9
theatmap 10

tspan 11
tsubset 12
tweet2r 13
valjson 15

Index **16**

`tweet2r-package` *Access to Twitter Streaming API via R and perform a exploratory space-time analyis*

Description

A package base on streamR package which provides a set of functions that allow R users to access Twitter's stream API. The package `tweet2r` has a function to capture tweets using Twitter streaming API two functions to export the tweets to PostgreSQL , Sqlite shp, GML or KML get Tweets as SpatialPointDataFrame, DataFrame, a function to get a summary of the tweets retrieved and a exploratory space-time analyis. Also there is a function to validate the JSON files where the tweets has been stored

Author(s)

Pau Aragó Galindo <`parago@uji.es`>

See Also

`streamR`, `tweet2r`, `t2pgis`, `t2sqlite`, `t2gis`, `tsubset`, `tglm`, `theatmap`, `tspan`, `t2summary`, `t2DataFrame`, `t2SpatialPointDataFrame`,

---

`t2DataFrame` *Parse tweets from JSON files and import to R*

Description

A function to parse JSON files containing tweets stored using the function `tweet2r`. Tweets are stored as a data.frame ready to be use within R.

Usage

```
t2DataFrame(fileprefix, path = ".", pattern = ".json$")
```

Arguments

`fileprefix`	File prefix for JSON files where tweets has been stored after harvesting. If tweets have been retrieved with the `tweet2r` function it should be the same fileprefix name.
`path`	A character vector to folder. see ?list.files()
`pattern`	Limits possible file extensions, optional as regular expression. see ?list.files()

Value

`tweets`	Return harvested tweets with `tweet2r` as a `data.frame`.

Author(s)

Pau Aragó Galindo <`parago@uji.es`>

See Also

`tweet2r`, `streamR`

Examples

```
## Not run:

  #json file prefix names
  fileprefix="tweets"

  #function to parse tweets from JSON file and import to R
  tweets<-t2DataFrame(fileprefix)

## End(Not run)
```

`t2gis` *Export Tweeter with geotag in Sqlite table to KML, shapefiles or GML*

Description

This functions import geotagged tweets as SpatialPointDataFrame from a Sqlite database containing the tweets and export it in a GIS format

Usage

```
t2gis(dbname, export)
```

Arguments

`dbname`	It is the Sqlite Database name where the tweets are stored. Sqlite is embedded in R within the package `RSQLite`. Sqlite database have no field extension. More information about Sqlite it its web page.
`export`	Export the tweets stored in the sqlite database to shape file, GML or KML. It imports only the geotagged tweets. The options are; shp (by default option), kml, gml. Note that only will be exported if there are geotweets.

Author(s)

Pau Aragó Galindo <`parago@uji.es`>

See Also

RSQLite, t2sqlite

Examples

```
## Not run:

#File prefix name for json files and also the name of the database
dbname=tweets
#export option to a GIS format
export="shp"

t2sqlite(dbname, export)

## End(Not run)
```

---

t2pgis *Set up parameters to JSON parsing and export it to a postGIS database.*

Description

This function parse the JSON files (as is defined in streamR package). Once is parsed export it to a database format the timestamp column and creates to tables one with all the tweets and other only with geotagged tweets.

Usage

```
t2pgis(fileprefix, con, path = ".", pattern = ".json$")
```

Arguments

fileprefix	Setup file prefix for JSON files. If tweets have been retrieved with the tweet2r function it should be the same. Te file prefix is used to create the table name and the geotagged table which has geo as a prefix (example "geofileprefix")
con	postGIS connection parameters. For more information look at RPostgreSQL
path	A character vector to folder. see ?list.files()
pattern	Limits possible file extensions, optional as regular expression. see ?list.files()

Note

It is mandatory to configure the connection to the database. The procedure is well described at RPostgreSQL package

Author(s)

Pau Aragó Galindo <parago@uji.es>

See Also

`RPostgreSQL`, `streamR`

Examples

```
## Not run:

#create a postgres connection
connection <- con <- dbConnect(PostgreSQL(), host="urls host",port=5432,
                    user="user", password="assword", dbname="pgistweets")
fileprefix="tweets"

t2pgis(fileprefix, connection)


## End(Not run)
```

`t2SpatialPointDataFrame` *Extract geotagged tweets stored as a data.frame*

Description

A function to create a `SpatialPointsDataFrame` with only the tweets with lat lon columns filled. Not all the tweets are geotagged. Tweets should be subset from the original `data.frame` to get only tweets with coordinates.

Usage

```
t2SpatialPointDataFrame(tweets)
```

Arguments

`tweets`	A `data.frame` with tweets.

Details

Tweets `data.frame` can be created with the function `t2DataFrame`

Value

`geotweets`	Return geotagged tweets as a `SpatialPointsDataFrame`.

Author(s)

Pau Aragó Galindo <`parago@uji.es`>

See Also

tweet2r, t2DataFrame, sp

Examples

```
## Not run:

  #tweets data.frame
  tweets="tweets"

  #function to extract only geotagged tweets a SpatialPointsDataFrame
  geotweets<-t2SpatialPointDataFrame(tweets)

## End(Not run)
```

---

t2sqlite *Export TweeterJSON files to sqlite database and import to R as a Data Frame*

Description

Export Json files retrieved with `tweet2r` to a Sqlite database and import tweets as a Data Frame.

Usage

```
t2sqlite(fileprefix, import, path = ".", pattern = ".json$")
```

Arguments

`fileprefix`	Setup file prefix for JSON files. If tweets have been retrieved with the `tweet2r` function, it should be the same name. The file prefix is used to create the table name and the geotagged table which will have geo as a prefix (example "geofileprefix")n. By default the database file will be save in the working directory with no file extension name. Sqlite is embedded in R within the package `RSQLite`. More information about Sqlite it its web page.
`import`	TRUE to import the tweets stored in the Sqlite database to R. It can be imported as a data frame with all the tweets retrieved with the function `tweet2r`. FALSE to not import as Data Frame
`path`	A character vector to folder. see ?list.files()
`pattern`	Limits possible file extensions, optional as regular expression. see ?list.files()

Author(s)

Pau Aragó Galindo <parago@uji.es>

See Also

`RSQLite`, `tweet2r`, `valjson`

Examples

```
## Not run:

#File prefix name for json files and also the name of the database
fileprefix=tweets

t2sqlite(fileprefix, import, export)

## End(Not run)
```

`t2STIDF` *Create objects of class STIDF*

Description

Creates an object of class STIDF from geotweets. A class for unstructured spatio-temporal data; for n spatial locations and times, n observations are available .

Usage

```
t2STIDF(geotweets)
```

Arguments

`geotweets`	Geotagged tweets as a SpatialPointsDataFrame.

Value

`sttweets`	A STDIF object.

Author(s)

Pau Aragó

References

Pebesma, Edzer. «Spacetime: Spatio-Temporal Data in R». Journal of Statistical Software 51, n.º 7 (2012). doi:10.18637/jss.v051.i07.`https://www.jstatsoft.org/article/view/v051i07`

See Also

`spacetime`, `tweet2r`

Examples

```
## Not run:
#Create STDIF object
t2STIDF(geotweets)

## End(Not run)
```

`t2summary` *Summary from retrieved tweets*

Description

Summary and graphical output from the tweets retrieved with the `tweet2r` package. The function performs a short description numerical and graphical.

Usage

```
t2summary(tweets, geotweets)
```

Arguments

`tweets`	Data frame with the tweets retrieved with `tweet2r`
`geotweets`	Spatial data frame with the geotweets retrieved with `tweet2r`

Value

`summt`	Number of tweets (geo and non geo) as 'ntweets', number of tweets with geotag as 'ngeotweets', number of tweets whit no geotag as diftweets, percentage of geotweets as 'pergeotweets'
`mapt`	Tweets' location Map
`ghour`	Plot of the number of tweets distributed by hour (UTC +000)
`gweekday`	Plot of the number of tweets distributed by days of the week (UTC +000)

Author(s)

Pau Aragó Galindo

See Also

`tweet2r`,`t2sqlite`,`t2pgis`

Examples

```
## Not run:

t2summary(tweets, geotweets)

#get summary description
summary

## End(Not run)
```

`tglm` *Generalized Linear Model for tweets*

Description

A fast time analysis. This function was done to get an exploratory time analysis base on a Generalized Linear Model (GLM), The function is based on `glm` function from. `glm` performs a GLM counting the tweets by hour, day, or week of the day.The results of this function is a `glm` object and the count of the tweets.

Usage

```
tglm(tweets,countby,family)
```

Arguments

`tweets`	Tweets harvested using `tweet2r` as DataFrame
`countby`	Tweets are count by default by weekday, also can be count by hour.Other options are "days" and "weekdays""
`family`	AvailableGLM family model in the function `glm`

Value

`tglm`	`glm` object
`countout`	Tweets count by as the `countby` option

Author(s)

Pau Aragó Galindo <`parago@uji.es`>

See Also

`glm`, `tweet2r`

Examples

```
## Not run:
myglm<-t2glm(tweets, "day", "poisson")

## End(Not run)
```

`theatmap` *Plot a heat map from geotweets or SpatialPointsDataFrame*

Description

Plot a heat maps base o 2d density estimation

Usage

```
theatmap(geotweets, vpoints=FALSE)
```

Arguments

`geotweets`	Geotagged tweets as a SpatialPointsDataFrame or a SpatialPointsDataFrame.
`vpoints`	By default FALSE. TRUE option will plot tweets points over the density map.

Details

This function is base on `ggmap`. It is a fast way to plot a density map, actually it can work with any SpatialPointsDataFrame. To customize your output, more information is available in the `ggmap` documentation page `https://cran.r-project.org/package=ggmap`

Note

It works only with small region if you have tweets for all over the world it won't work properly. To get a subsample of the geotweets for a specific region use `tsubset`.

Author(s)

Pau Arago

See Also

`tsubset`, `tweet2r`, `ggmap`

Examples

```
## Not run:
theatmap(geotweets)

## End(Not run)
```

tspan *Exploratory point pattern analysis.*

Description

This function is design to provide an exploratory point pattern analysis. Is base on `spatstat` package a to do a basic point pattern analysis of Homogeneous an Inhomogeneous Poisson.

Usage

```
tspan(geotweets,bw, cont, acontour)
```

Arguments

`geotweets`	Geotagged tweets as a SpatialPointsDataFrame or a SpatialPointsDataFrame.
`bw`	Bandwith for Kernel Smoothed Intensity. Note that if you are using directly geotweets coming from `tweet2r` and `t2SpatialPointDataFrame` units are degrees.
`cont`	FALSE by default, geotweets bounding box provide the contour. If TRUE a contour must be provided
`acontour`	Optional. A Spatial object with a defined bbox.

Details

In order to do a wider point pattern analysis is better to use directly the `spatstat` package

Value

`tweetspphp`	Simpliest Object of class "ppp" representing a point pattern dataset in the two-dimensional plane with no marks, (`ppp`)
`hp`	Homogeneous Poisson fitted point process model to an observed point pattern (`ppm`).
`ihp`	Inhomogeneous Poisson fitted point process model to an observed point pattern (`ppm`).
`int`	Computed kernel smoothed intensity function from a point pattern. (`density.ppp`).

Author(s)

Pau Aragó

References

Baddeley, Adrian, y Rolf Turner. «Spatstat: An R Package for Analyzing Spatial Point Patterns». Journal of Statistical Software 12, n.º 6 (2005). doi:10.18637/jss.v012.i06. `http://www.jstatsoft.org/v12/i06/`

See Also

spatstat

Examples

```
library(sp)
library(spatstat)

#loada a SpatialPointsDataFrame
data("meuse.grid_ll")


# run function without contour
tspan(meuse.grid_ll,bw=0.0005)

#providing a contour as SpatialPointDataFrame
data("meuse.area")

#build the acontour layer
cont<-SpatialPoints(meuse.area, proj4string = CRS("+init=epsg:28992"))
#transform to meuse.grid_ll reference system
cont<-spTransform(cont, CRS("+init=epsg:4326"))

# run function with contour
tspan(meuse.grid_ll,bw=0.0005, cont = TRUE, acontour=cont)
{
  }
```

`tsubset` *Subset points from geotweets or SaptialPointsDataFrame*

Description

This function do a subset points over a polygon defined by a SpatialPolygonsDataFrame. Takes the points within the Polygon and remove the points outside it.

Usage

```
tsubset(geotweets, pbox, transform=TRUE)
```

Arguments

`geotweets`	input Geotagged tweets as a SpatialPointsDataFrame.
`pbox`	A SpatialPolygonsDataFrame to get the points over it.
`transform`	Option TRUE by default to transform the pbox CRS (Coordinate Reference System) to geotweets CRS.

Author(s)

Pau Aragó

Examples

```
## Not run:
tweetsSubset<-tsubset(geotweets, pbox)

## End(Not run)
{
  }
```

`tweet2r` *Set up parameters to file streaming and store tweets in a JSON file using Twitter Streaming API .*

Description

Configuration of the start stop for retrieving tweets, also set up the search by bounding box or by key words, number of tweets retrieved (stored in a collection of JSON files) and set up of the file prefix to store the JSON files created during the connection to the to Twitter Streaming API.`tweet2r` Don't opens a connection to Twitter's Streaming API by himself but you can configure the tweeter connection as is described in streamR that will return public statuses that match your keywords or bounding box definition. It is very useful to set up start time and end time of the connection. Note that the end time will be delayed depending on the time needed to close the file for the number of tweets set up.

Usage

```
tweet2r(t_start,t_end, ntweets=NULL,keywords=NULL, bbox=NULL, fileprefix,
        consumerKey,consumerSecret,requestURL,accessURL,authURL)
```

Arguments

`t_start`	Configuration of the start time to retrieve tweets. It use the strptime function
`t_end`	Configuration of the end time to stop retrieving tweets. It use the strptime function
`ntweets`	Number of tweets retrieved per file. The stream connection remains open, but the tweets are being stared in the files each time arrives to number of tweets configured. Nevertheless the number of tweets retrieved could be less, that depends on twitter API)
`keywords`	Vector with the keywords used to retrieve tweets. More information in twitter developers web `https://dev.twitter.com/streaming/overview/request-parameters`
`bbox`	Vector with the bounding box coordinates from which you want to retrieve tweets. You will retrieve geotagged tweets as well as tweets from users that has in his profile this area as a residence `https://dev.twitter.com/streaming/overview/request-parameters`

`fileprefix`	Fileprefix for the JSON file where the tweets will be stored. Avoid number at the begging this will be confusing for SQL database
`consumerKey`	Twitter API consumer key `https://dev.twitter.com/oauth`
`consumerSecret`	Twitter API consumer secret `https://dev.twitter.com/oauth`
`requestURL`	Twitter API url to request data `https://dev.twitter.com`
`accessURL`	Twitter API url to access `https://dev.twitter.com`
`authURL`	Twitter API authentification `https://dev.twitter.com`

Note

There is a API configuration by default, therefore it is not mandatory to configure a tweeter API's connection. Nevertheless this default connection could be overload sometimes or a user could prefer to use its own connection. The procedure is well described in streamR package

Author(s)

Pau Aragó Galindo <`parago@uji.es`>

See Also

`tweet2r`,`parseTweets`,`streamR`

Examples

```
## Not run:

#Configuration fo twitter API connection
  #this could be avoid and use the default configuration
requestURL <- "https://api.twitter.com/oauth/request_token"
accessURL <- "https://api.twitter.com/oauth/access_token"
authURL <- "https://api.twitter.com/oauth/authorize"
consumerKey <- sys.getenv("consumer_key")
consumerSecret <- sys.getenv("consumer_secret")

#definition of the start time and end time
t_start<-"2015-09-11 9:45:00"
t_end<-"2015-09-11 23:59:59"

#definition of the file prefix
fileprefix="tweets"
key=c("keyword1", "keyword2")

   #definition number of tweets per file
   number=3000

#running the function
tweet2r(t_start=t_start,t_end=t_end,ntweets=number,keywords=key,fileprefix = fileprefix,
requestURL,accessURL,authURL,consumerKey,consumerSecret)
```

```
#running the function using bbox
#set up a bbox
bbox=c(-0.1644,39.8485,0.6916,40.0034)

tweet2r(t_start=t_start,t_end=t_end,ntweets=number,bbox=bbox,fileprefix = fileprefix,
requestURL,accessURL,authURL,consumerKey,consumerSecret)

## End(Not run)
```

valjson *Json validation function*

Description

Function to validate json files created by tweet2r. This function deletes json files with Tweeter streaming API messages and rename json files consecutively according to a fileprefix.

Usage

```
valjson(fileprefix, path = ".", pattern = ".json$")
```

Arguments

`fileprefix`	Fileprefix name from json files
`path`	A character vector to folder. see ?list.files()
`pattern`	Limits possible file extensions, optional as regular expression. see ?list.files()

Author(s)

Pau Aragó Galindo <parago@uji.es>

See Also

`tweet2r`

Examples

```
## Not run:
valjson(fileprefix)

## End(Not run)
```

Index

∗Topic **GML**
    t2gis, 3
∗Topic **data.frame**
    t2DataFrame, 2
    t2SpatialPointDataFrame, 5
∗Topic **glm**
    tglm, 9
∗Topic **heat map**
    theatmap, 10
∗Topic **json**
    valjson, 15
∗Topic **kML**
    t2gis, 3
∗Topic **parse**
    t2DataFrame, 2
    t2SpatialPointDataFrame, 5
∗Topic **point pattern**
    tspan, 11
∗Topic **shp**
    t2gis, 3
∗Topic **spacetime**
    t2STIDF, 7
∗Topic **sqlite**
    t2gis, 3
    t2sqlite, 6
∗Topic **subset**
    tsubset, 12
∗Topic **summary**
    t2summary, 8
∗Topic **tweet2r**
    tglm, 9
    valjson, 15
∗Topic **tweeter**
    t2gis, 3
    t2sqlite, 6
∗Topic **tweets**
    t2DataFrame, 2
    t2SpatialPointDataFrame, 5
    t2summary, 8

data.frame, *3*, *5*
density.ppp, *11*

ggmap, *10*
glm, *9*

parseTweets, *14*
ppm, *11*
ppp, *11*

sp, *6*
SpatialPointsDataFrame, *5*
spatstat, *11, 12*
streamR, *2–5*, *13, 14*
strptime, *13*

t2DataFrame, *2*, 2, *5, 6*
t2gis, *2*, 3
t2pgis, *2*, 4, *8*
t2SpatialPointDataFrame, *2*, 5, *11*
t2sqlite, *2*, 6, *8*
t2STIDF, 7
t2summary, *2*, 8
tglm, *2*, 9
theatmap, *2*, 10
tspan, *2*, 11
tsubset, *2*, *10*, 12
tweet2r, *2–4*, *6–8*, *10, 11*, 13, *14, 15*
tweet2r-package, 2

valjson, *7*, 15