Package 'weed'

October 12, 2022

Title Wrangler for Emergency Events Database

Version 1.1.1

Maintainer Ram Kripa <ram.m.kripa@berkeley.edu>

Description Makes research involving EMDAT and related datasets easier. These Datasets are manually filled and have several formatting and compatibility issues. Weed aims to resolve these with its functions.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Imports readxl, dplyr, magrittr, tidytext, stringr, tibble, geonames, countrycode, purrr, tidyr, forcats, ggplot2, rgeos, sf, here

URL https://github.com/rammkripa/weed

BugReports https://github.com/rammkripa/weed/issues

NeedsCompilation no

Author Ram Kripa [aut, cre]

Repository CRAN

Date/Publication 2021-07-07 16:10:02 UTC

R topics documented:

geocode *GeoCodes text locations using the GeoNames API*

Description

Uses the `location_word` and `Country` columns of the data frame to make queries to the geonames API and geocode the locations in the dataset.

Note:

1. The Geonames API (for free accounts) limits you to 1000 queries an hour
2. You need a geonames username to make queries. You can learn more about that here

Usage

```
geocode(., n_results = 1, unwrap = FALSE, geonames_username)
```

Arguments

`.`	a data frame which has been locationized (see `weed::split_locations`)
`n_results`	number of lat/longs to get
`unwrap`	if true, returns lat1, lat2, lng1, lng2 etc. as different columns, otherwise one lat column and 1 lng column
`geonames_username`	Username for geonames API. More about getting one is in the note above.

Value

the same data frame with a lat column/columns and lng column/columns

Examples

```
df <- tibble::tribble(
   ~value,  ~location_word,                    ~Country,
   "mumbai region, district of seattle, sichuan province",  "mumbai","India",
   "mumbai region, district of seattle, sichuan province",  "seattle", "USA"
   )
geocode(df, n_results = 1, unwrap = TRUE, geonames_username = "rammkripa")
```

geocode_batches *Geocode in batches*

Description

Geocode in batches

Usage

```
geocode_batches(
  .,
  batch_size = 990,
  wait_time = 4800,
  n_results = 1,
  unwrap = FALSE,
  geonames_username
)
```

Arguments

.	data frame
batch_size	size of each batch to geocode
wait_time	in seconds between batches Note: default batch_size and wait_time were set to accomplish the geocoding task optimally within the constraints of geonames free access
n_results	same as geocode
unwrap	as in geocode
geonames_username	as in geocode

Value

df geocoded

Examples

```
df <- tibble::tribble(
   ~value,  ~location_word,                    ~Country,
   "mumbai region, district of seattle, sichuan province",  "mumbai","India",
   "mumbai region, district of seattle, sichuan province",  "seattle", "USA",
  "mumbai region, district of seattle, sichuan province", "sichuan",  "China, People's Republic"
   )

geocode_batches(df, batch_size = 2, wait_time = 0.4, geonames_username = "rammkripa")
```

`located_in_box` *Locations In the Box*

Description

Creates a new column (in_box) that tells whether the lat/long is in a certain box or not.

Usage

```
located_in_box(
  .,
  lat_column = "lat",
  lng_column = "lng",
  top_left_lat,
  top_left_lng,
  bottom_right_lat,
  bottom_right_lng
)
```

Arguments

`.`	Data Frame that has been locationized. see `weed::split_locations`
`lat_column`	Name of column containing Latitude data
`lng_column`	Name of column containing Longitude data
`top_left_lat`	Latitude at top left corner of box
`top_left_lng`	Longitude at top left corner of box
`bottom_right_lat`	Latitude at bottom right corner of box
`bottom_right_lng`	Longitude at bottom right corner of box

Value

A dataframe that contains the latlong box data

Examples

```
d <- tibble::tribble(
~value,  ~location_word,                    ~Country,     ~lat,       ~lng,
"city of new york",      "new york",                        "USA", 40.71427,  -74.00597,
"kerala, chennai municipality, and san francisco",  "kerala", "India", 10.41667,     76.5,
"kerala, chennai municipality, and san francisco",  "chennai",  "India", 13.08784,  80.27847)
located_in_box(d, lat_column = "lat",
lng_column = "lng",
top_left_lat = 45,
bottom_right_lat = 12,
top_left_lng = -80,
bottom_right_lng = 90)
```

`located_in_shapefile` *Locations In the Shapefile*

Description

Creates a new column (in_shape) that tells whether the lat/long is in a certain shapefile.

Usage

```
located_in_shapefile(
  .,
  lat_column = "lat",
  lng_column = "lng",
  shapefile = NA,
  shapefile_name = NA
)
```

Arguments

`.`	Data Frame that has been locationized. see `weed::split_locations`
`lat_column`	Name of column containing Latitude data
`lng_column`	Name of column containing Longitude data
`shapefile`	The shapefile itself (either shapefile or shapefile_name must be provided)
`shapefile_name`	FileName/Path to shapefile (either shapefile or shapefile_name must be provided)

Value

Data Frame with the shapefile data as well as the previous data

Examples

```
## Not run:
d <- tibble::tribble(
~value,  ~location_word,                    ~Country,     ~lat,       ~lng,
"city of new york",      "new york",                        "USA", 40.71427,  -74.00597,
"kerala, chennai municipality, and san francisco",  "kerala", "India", 10.41667,     76.5,
"kerala, chennai municipality, and san francisco",  "chennai",  "India", 13.08784,  80.2847)
located_in_shapefile(d,
lat_column = "lat",
lng_column = "lng",
shapefile_name = "~/dummy_name")

## End(Not run)
```

`nest_locations` *Nest Location Data into a column of Tibbles*

Description

Nest Location Data into a column of Tibbles

Usage

```
nest_locations(
  .,
  key_column = "Dis No",
  columns_to_nest = c("location_word", "lat", "lng"),
  keep_nested_cols = FALSE
)
```

Arguments

`.`	Locationized data frame (see `weed::split_locations`)
`key_column`	Column name for Column that uniquely IDs each observation
`columns_to_nest`	Column names for Columns to nest inside the mini-dataframes
`keep_nested_cols`	Boolean to Keep the nested columns externally or not.

Value

Data Frame with A column of data frames

Examples

```
d <- tibble::tribble(
~value,  ~location_word,                    ~Country,    ~lat,   ~lng,
"city of new york","new york","USA",  c(40.71427, 40.6501),   c(-74.00597, -73.94958),
"kerala", "kerala", "India",c(10.41667, 8.4855), c(76.5, 76.94924),
"chennai municipality","chennai","India", c(13.08784, 12.98833),c(80.27847, 80.16578),
"san francisco", "san francisco","USA", c(37.77493, 37.33939), c(-122.41942, -121.89496))
nest_locations(d, key_column = "value")
```

`percent_located_disasters` *Percent of Disasters Successfully Geocoded*

Description

Tells us how successful the geocoding is.

How many of the disasters in this data frame have non NA coordinates?

Usage

```
percent_located_disasters(
  .,
  how = "any",
  lat_column = "lat",
  lng_column = "lng",
  plot_result = TRUE
)
```

Arguments

`.`	Data Frame that has been locationized. see `weed::split_locations`
`how`	takes in a function, "any", or "all" to determine how to count the disaster as being geocoded if any, at least one location must be coded, if all, all locations must have lat/lng if a function, it must take in a logical vector and return a single logical
`lat_column`	Name of column containing Latitude data
`lng_column`	Name of column containing Longitude data
`plot_result`	Determines output type (Plot or Summarized Data Frame)

Value

The percent and number of Locations that have been geocoded (see `plot_result` for type of output)

Examples

```
d <- tibble::tribble(
~`Dis No`, ~value,  ~location_word,                    ~Country,     ~lat,       ~lng,
1, "city of new york",      "new york",                      "USA", 40.71427,  -74.00597,
2, "kerala, chennai municipality, and san francisco", "kerala", "India", 10.41667,     76.5,
2, "kerala, chennai municipality, and san francisco", "chennai", "India", 13.08784,  80.27847)
percent_located_disasters(d,
how = "any",
lat_column = "lat",
lng_column = "lng",
plot_result = FALSE)
```

`percent_located_locations` *Percent of Locations Successfully Geocoded*

Description

Tells us how successful the geocoding is.

How many of the locations in this data frame have non NA coordinates?

Usage

```
percent_located_locations(
  .,
  lat_column = "lat",
  lng_column = "lng",
  plot_result = TRUE
)
```

Arguments

`.`	Data Frame that has been locationized. see `weed::split_locations`
`lat_column`	Name of column containing Latitude data
`lng_column`	Name of column containing Longitude data
`plot_result`	Determines output type (Plot or Summarized Data Frame)

Value

The percent and number of Locations that have been geocoded (see `plot_result` for type of output)

Examples

```
d <- tibble::tribble(
~value,  ~location_word,                    ~Country,     ~lat,       ~lng,
"city of new york",      "new york",                        "USA", 40.71427,  -74.00597,
"kerala, chennai municipality, and san francisco",  "kerala", "India", 10.41667,     76.5,
"kerala, chennai municipality, and san francisco",  "chennai",  "India", 13.08784,  80.27847)
percent_located_locations(d,
lat_column = "lat",
lng_column = "lng",
plot_result = FALSE)
```

read_emdat *Reads Excel Files obtained from EM-DAT Database*

Description

Reads Excel files downloaded from the EMDAT Database linked here

Usage

```
read_emdat(path_to_file, file_data = TRUE)
```

Arguments

path_to_file	A String, the Path to the file downloaded.
file_data	A Boolean, Do you want information about the file and how it was created?

Value

Returns a list containing one or two tibbles, one for the Disaster Data, and one for File Metadata.

Examples

```
## Not run:
read_emdat(path_to_file = "~/dummy", file_data = TRUE)

## End(Not run)
```

split_locations *Splits string of manually entered locations into one row for each location*

Description

Changes the unit of analysis from a disaster, to a disaster-location. This is useful as preprocessing before geocoding each disaster-location pair.

Can be used in piped operations, making it tidy!

Usage

```
split_locations(
  .,
  column_name = "locations",
  dummy_words = c("cities", "states", "provinces", "districts", "municipalities",
   "regions", "villages", "city", "state", "province", "district", "municipality",
    "region", "township", "village", "near", "department"),
  joiner_regex = ",|\\(|\\)|;|\\+|( and )|( of )"
)
```

Arguments

.	data frame of disaster data
column_name	name of the column containing the locations
dummy_words	a vector of words that we don't want in our final output.
joiner_regex	a regex that tells us how to split the locations

Value

same data frame with the location_word column added as well as a column called uncertain_location_specificity where the same location could be referred to in varying levels of specificity

Examples

```
locs <- c("city of new york", "kerala, chennai municipality, and san francisco",
"mumbai region, district of seattle, sichuan province")
d <- tibble::as_tibble(locs)
split_locations(d, column_name = "value")
```

Index