Package ‘globaltrends’

March 6, 2023

Type Package

Title Download and Measure Global Trends Through Google Search Volumes

Version 0.0.14

Description Google offers public access to global search volumes from its search engine through the Google Trends portal. The package downloads these search volumes provided by Google Trends and uses them to measure and analyze the distribution of search scores across countries or within countries. The package allows researchers and analysts to use these search scores to investigate global trends based on patterns within these scores. This offers insights such as degree of internationalization of firms and organizations or dissemination of political, social, or technological trends across the globe or within single countries. An outline of the package's methodological foundations and potential applications is available as a working paper: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3969013>.

License MIT + file LICENSE

URL https://github.com/ha-pu/globaltrends/

BugReports https://github.com/ha-pu/globaltrends/issues/

Depends R (>= 3.5.0)

Imports DBI (>= 1.1.0), dbplyr (>= 1.4.4), dplyr (>= 1.0.2), forcats (>= 0.5.0), forecast (>= 8.12), ggplot2 (>= 3.3.2), gtrendsR (>= 1.5.1), lubridate (>= 1.7.9), purrr (>= 0.3.4), rlang (>= 0.4.7), RSQLite (>= 2.2.0), stats (>= 3.5.0), stringr (>= 1.4.0), tibble (>= 3.0.3), tidyr (>= 1.0.0), utils (>= 3.5.0), zoo (>= 1.8.8)

Suggests knitr (>= 1.29), rmarkdown (>= 2.3), testthat (>= 2.3.2)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Harald Puhr [aut, cre],
Jakob Muellner [ccp]

Maintainer Harald Puhr <harald.puhr@gmail.com>

Repository CRAN

Date/Publication 2023-03-06 19:50:02 UTC

R topics documented:

`add_control_keyword` *Add batches of control or object keywords*

Description

The function adds one or more batches of keywords with a time period for downloads to the database. The batches serve as input for all download and computation functions.

Usage

```
add_control_keyword(keyword, time = "2010-01-01 2020-12-31")

add_object_keyword(keyword, time = "2010-01-01 2020-12-31")
```

Arguments

`keyword`	Keywords that should be added as batch. Vector of type `character` or a `list` of `character` vectors. The function also allows the usage of codes for search topics instead of search terms.
`time`	Time frame for which the batch data should be downloaded. Object of type `character` that takes the from "YYYY-MM-DD YYYY-MM-DD". Defaults to *"2010-01-01 2020-12-31"*.

Details

Since Google Trends allows a maximum of five keywords for each query, batches of control keywords can consist of up to five keywords. Since one control keyword is added to batches of object keywords for mapping, object batch length is limited to four keywords. When a `character` vector contains more than four (five) keywords, the vector is split into four-keyword (five-keyword) batches. A `list` must contain `character` vectors of length four (five) or less. Each batch of keywords is combined with a time period for which data will be downloaded. To change the time period for an existing batch, all downloads and computations must be rerun.

Value

Message that the batch has been created successfully. Batch data is written to tables *batch_keywords* and *batch_time*. Numeric vector containing the newly added batch numbers are returned.

Warning

If you use search topics for object keywords, make sure to use search topics for control keywords and vice versa. See Google's FAQ for additional information on search topics.

Note

To avoid trailing spaces `stringr::str_squish` is automatically applied to all keywords.

See Also

- `example_keywords()`
- `example_time()`
- `stringr::str_squish()`

Examples

```
## Not run:
add_control_keyword(
  keyword = c("gmail", "maps", "translate", "wikipedia", "youtube"),
  time = "2016-01-01 2019-12-31"
)
add_object_keyword(
  keyword = c("apple", "facebook", "google", "microsoft"),
  time = "2016-01-01 2019-12-31"
)

add_control_keyword(
  keyword = c("gmail", "maps", "news", "translate", "weather", "wikipedia", "youtube"),
  time = "2016-01-01 2019-12-31"
)
add_control_keyword(
 keyword = c("amazon", "apple", "facebook", "google", "microsoft", "netflix", "twitter"),
  time = "2016-01-01 2019-12-31"
)

add_control_keyword(
  keyword = list(
    c("gmail", "maps", "news"),
    c("translate", "weather", "wikipedia", "youtube")
  ),
  time = "2016-01-01 2019-12-31"
)
add_control_keyword(
  keyword = list(
    c("amazon", "apple", "facebook", "google"),
    c("microsoft", "netflix", "twitter")
  ),
  time = "2016-01-01 2019-12-31"
)

# search topics
add_control_keyword(
 keyword = c("%2Fm%2F02q_bk", "%2Fm%2F055t58", "%2Fm%2F025sndk", "%2Fm%2F0d07ph", "%2Fm%2F09jcvs"),
  time = "2016-01-01 2019-12-31"
)
# This adds the following topics: Gmail, Google Maps, Google Translate, Wikipedia, YouTube

## End(Not run)
```

---

add_locations *Add sets of locations*

Description

The function adds a new set of locations for downloads and computations to the database. The location set serves as input for all download and computation functions.

Usage

```
add_locations(locations, type, export = TRUE)
```

Arguments

`locations`	Locations that should be added as set of locations. Vector of type `character`.
`type`	Name of the location set that should be added. Object of type `character` of length 1.
`export`	Indicator whether the new location set should be directly exported to the package environment `gt.env`. Object of type `logical`, defaults to `TRUE`.

Details

Location sets control the locations for which data is downloaded or to which computations are applied. By adding new location sets, the default sets *countries* and *us_states* can be expanded by additional sets. Thereby, users can compute DOI within a region (e.g., adding EU countries as a set) or single countries (e.g., adding regions of France as a set). Download and computation functions check whether data for a location already exists. Therefore, data will not be duplicated when location data already exists from another set.

Value

Message that the location set has been created successfully. Location data is written to table *data_locations*.

Warning

Unfortunately, the Google Trends API cannot handle the location "NA - Namibia". Therefore, the location will be dropped automatically.

Examples

```
## Not run:
add_locations(locations = c("AT", "CH", "DE"), type = "DACH")

## End(Not run)
```

add_synonym *Add synonyms for object keywords*

Description

The function allows to add synonyms for object keywords. Sometimes, objects of interest can be searched with different keywords on Google e.g., FC Bayern for Bayern Munich. Search scores for keywords that are added as synonyms are aggregated when running `compute_score`. The function allows to add synonyms for a single keyword at a time.

Usage

```
add_synonym(keyword, synonym)

## S3 method for class 'character'
add_synonym(keyword, synonym)

## S3 method for class 'list'
add_synonym(keyword, synonym)
```

Arguments

`keyword`	Keyword of type `character` and length 1 for which the synonyms are added.
`synonym`	Synonym of type `character`.

Value

Message that the synonym has been added successfully. Synonym data is written to table *keyword_synonyms*.

Note

To avoid trailing spaces `stringr::str_squish` is automatically applied to all keywords and synonyms.

See Also

- `compute_score()`
- `stringr::str_squish()`

Examples

```
## Not run:
add_synonym(
  keyword = "fc bayern",
  synonym = "bayern munich"
)
```

```
## End(Not run)
```

`batch_keywords` *batch_keywords*

Description

The table *batch_keywords* contains the keywords for each batch. Each line contains one *keyword*, the *type* of the batch (i.e., control or object) and the id of the *batch* to which the keyword is assigned. Keywords can be added with the function `add_keywords`. The function `start_db` exports the table *batch_keywords* as objects `keywords_control` and `keywords_object` to the package environment `gt.env`.

Example data for the table *batch_keywords* is available as R object `example_keywords`.

Usage

```
example_keywords
```

Format

A tibble with 19 rows and 3 variables:

type Column of type `character` showing the type of each batch, takes "control" for control batches and "object" for object batches.

batch Column of type `integer` showing the number of each batch.

keyword Column of type `character` showing the keywords included in each batch.

See Also

- `add_control_keyword()`

`batch_time` *batch_time*

Description

The table *batch_time* contains the time period for which data is downloaded for each batch. Each line contains one *time* period, the *type* of the batch (i.e., control or object) and the id of the *batch* to which the time period is assigned. Time frames take the form `"YYYY-MM-DD YYYY-MM-DD"`. Time periods are added automatically through the function `add_keywords`. The function `start_db` exports the table *batch_time* as objects `time_control` and `time_object` to `.GlobalEnv`.

Example data for the table *batch_time* is available as R object `example_time`.

Usage

```
example_time
```

Format

A tibble with 5 rows and 3 variables:

type Column of type `character` showing the type of each batch, takes "control" for control batches and "object" for object batches.

batch Column of type `integer` showing number of each batch.

time Column of type `character` showing the time period for each batch as "YYYY-MM-DD YYYY-MM-DD".

See Also

- `add_control_keyword()`

`compute_doi` *Aggregate keyword-country data and compute DOI*

Description

The function computes degree of internationalization (DOI) for object keywords. Degree of internationalization is measured based on the distribution of country search scores.

Usage

```
compute_doi(object, control = 1, locations = "countries")

## S3 method for class 'numeric'
compute_doi(object, control = 1, locations = "countries")

## S3 method for class 'list'
compute_doi(object, control = 1, locations = "countries")
```

Arguments

`object`	Object batch for which the keyword-country data is aggregated and DOI is computed. Object of type `numeric`.
`control`	Control batch for which the search score is used. Object of type `numeric`.
`locations`	List of locations for which the search score is used. Object of type `character`. Defaults to *"countries"*.

Details

The function uses an inverted Gini-coefficient as measure for the degree of internationalization. The more uniform the distribution of search scores across all countries, the higher the inverted Gini-coefficient and the greater the degree of internationalization. In addition to the Gini-coefficient, the package uses inverted Herfindahl index and inverted Entropy as measures for internationalization.

Value

Message that data was aggregated successfully. Data is written to table *data_doi*.

See Also

- `example_doi()`

Examples

```
## Not run:
compute_doi(
  object = 1,
  control = 1,
  locations = "countries"
)
compute_doi(
  object = as.list(1:5),
  control = 1,
  locations = "countries"
)

## End(Not run)
```

`compute_score` *Compute keyword-country search score*

Description

The function computes search scores for object keywords. Search volumes for control and object batches are mapped to the same base. Next, search volumes for object batches are divided by the sum of search volumes for the respective control batch. `compute_voi` computes volume of internationalization (VOI) as global search scores.

Usage

```
compute_score(object, control = 1, locations = gt.env$countries)

## S3 method for class 'numeric'
compute_score(object, control = 1, locations = gt.env$countries)
```

```
## S3 method for class 'list'
compute_score(object, control = 1, locations = gt.env$countries)

compute_voi(object, control = 1)
```

Arguments

`object`	Object batch for which the data is downloaded. Object of type `numeric` or object of type `list` containing single objects of type `numeric`.
`control`	Control batch for which the data is downloaded. Object of type `numeric`. Defaults to 1.
`locations`	List of countries or regions for which the data is downloaded. Refers to lists generated in `start_db`. Defaults to `countries`.

Details

The search score computation proceeds in four steps. First, the function aggregates all search volumes to monthly data. Then, it applies some optional time series adjustments: seasonally adjusted `forecast::seasadj` and trend only `stats::stl`. Next, it follows the procedure outlined by Castelnuovo and Tran (2017, pp. A1-A2) to map control and object data. After the mapping, object search volumes are divided by the sum of control search volumes in the respective control batch. We use the sum of search volumes for a set of control keywords, rather than the search volumes for a single control keyword, to smooth-out variation in the underlying control data.

Castelnuovo, E. & Tran, T. D. 2017. Google It Up! A Google Trends-based Uncertainty index for the United States and Australia. Economics Letters, 161: 149-153.

Value

Message that data has been computed successfully. Data is written to table *data_score*.

Note

When synonyms were specified through `add_synonym`, search scores for synonyms are added to the main keyword.

See Also

- `example_score()`
- `add_synonym()`
- `stats::stl()`
- `forecast::seasadj()`

Examples

```
## Not run:
compute_score(
  object = 1,
  control = 1,
  locations = countries
)
compute_voi(
  object = 1,
  control = 1
)
compute_score(
  object = as.list(1:5),
  control = 1,
  locations = countries
)

## End(Not run)
```

`countries` *countries*

Description

A character vector that includes ISO2 codes for all countries with a share in global GDP >= 0.1\ World Development Indicators database. The data includes:

- AE
- AO
- AR
- AT
- AU
- BD
- BE
- BR
- CA
- CH
- CL
- CN
- CO
- CU
- CZ
- DE
- DK
- DO
- DZ

- EC
- EG
- ES
- ET
- FI
- FR
- GB
- GR
- HK
- HU
- ID
- IE
- IL
- IN
- IQ
- IR
- IT
- JP
- KR
- KW
- KZ
- LK
- MA
- MX
- MY
- NG
- NL
- NO
- NZ
- OM
- PE
- PH
- PK
- PL
- PR
- PT
- QA

- RO
- RU
- SA
- SD
- SE
- SG
- SK
- TH
- TR
- TW
- UA
- US
- UZ
- VN
- ZA

Usage

```
countries
```

Format

An object of class `character` of length 71.

`countries_wdi` *countries_wdi*

Description

A data.frame that includes ISO2 codes and country names for all countries and locations in the World Bank's World Development Indicators database.

Usage

```
countries_wdi
```

Format

An object of class `data.frame` with 299 rows and 2 columns.

`data_control` *data_control*

Description

The table *data_control* contains the downloaded data for each control batch. Each line contains the search *hits* for each *keyword* in a control *batch* for a given *location* and *date*. Global data gets the value *world* as location. Data is downloaded and automatically written to the table through the function `download_control`. The function `start_db` exports the table *data_control* as database connection `tbl_control` to the package environment `gt.env`. Users can access the database table through `dplyr::tbl`. The sample data included in `data_control` was simulated based on actual Google Trends data.

Example data for the table *data_control* is available as R object `example_control`.

Usage

```
example_control
```

Format

A tibble with 2,400 rows and 5 variables:

location Column of type `character` showing the ISO2 code of the country or region for which the data was downloaded.

keyword Column of type `character` showing the keyword for which the data was downloaded.

date Column of type `integer` showing the date for which the data was downloaded. Can be transformed into date format with `lubridate::as_date.\item{hits}{Column of typedoubleshowing search volumes f` showing the number of each batch.

Source

https://trends.google.com/trends/

See Also

- `download_control()`
- `dplyr::tbl()`

`data_doi` *data_doi*

Description

The table *data_doi* contains the degree of internationalization (DOI) for each object batch. Each line contains the DOI computed as inverted *gini* coefficient, as inverted *hhi*, or inverted *entropy* for each *keyword* in an object *batch_o* for a given *date* and *type* of search score. The column *batch_c* indicates the control batch that has been used as baseline for mapping. Column *locations* indicates which set of locations was used to compute the distribution of search scores. DOI is computed and automatically written to the table with the function `compute_doi`. The function `start_db` exports the table *data_doi* as database connection `tbl_doi` to the package environment `gt.env`. Users can access the database table through `dplyr::tbl`. The sample data included in `data_doi` was simulated based on actual Google Trends data.

Usage

```
example_doi
```

Format

A tibble with 4,320 rows and 9 variables:

keyword Column of type `character` showing the keyword for which the data was computed.

date Column of type `integer` showing the date for which the data was computed Can be transformed into date format with `lubridate::as_date`.

type Column of type `character` indicating the type of time series-column from `data_score` that is used for DOI computation, takes either "score_obs", "score_sad", or "score_trd".

gini Column of type `double` showing the DOI computed as inverted Gini coefficient of the search score distribution from `data_score`.

hhi Column of type `double` showing the DOI computed as inverted Herfindahl-Hirschman index of the search score distribution from `data_score`.

entropy Column of type `double` showing the DOI computed as inverted Entropy measure for the search score distribution from `data_score`.

batch_c Column of type `integer` showing the number of each control batch.

batch_o Column of type `integer` showing the number of each object batch.

locations Column of type `character` showing the list of locations for which the search score distribution is used.

See Also

- `compute_doi()`
- `dplyr::tbl()`

`data_object` *data_object*

Description

The table *data_object* contains the downloaded data for each object batch. Each line contains the search *hits* for each *keyword* in an object *batch_o* for a given *location* and *date*. The column *batch_c* indicates the control batch to which the data will be mapped. Global data takes the value *world* as location. Data is downloaded and automatically written to the table through the function `download_object`. The function `start_db` exports the table *data_object* as database connection `tbl_object` to the package environment `gt.env`. Users can access the database table through `dplyr::tbl`. The sample data included in `data_object` was simulated based on actual Google Trends data.

Example data for the table *data_object* is available as R object `example_object`.

Usage

```
example_object
```

Format

A tibble with 8,640 rows and 6 variables:

location Column of type `character` showing the ISO2 code of the country or region for which the data was downloaded.

keyword Column of type `character` showing the keyword for which the data was downloaded.

date Column of type `integer` showing the date for which the data was downloaded. Can be transformed into date format with `lubridate::as_date`.

hits Column of type `double` showing search volumes for the respective location-keyword-date combination.

batch_c Column of type `integer` showing the number of each control batch.

batch_o Column of type `integer` showing the number of each object batch.

Source

https://trends.google.com/trends/

See Also

- `download_object()`
- `dplyr::tbl()`

`data_score` *data_score*

Description

The table *data_score* contains the search scores for each object batch. Each line contains the observed search score (*score_obs*), the seasonally adjusted search score (*score_sad*), and the trend only search score (*score_trd*) for each *keyword* in an object *batch_o* for a given *location* and *date*. The column *batch_c* indicates the control batch that has been used as baseline for mapping. Global data takes the value *world* as location. Search scores are computed and automatically written to the table with the function `compute_score`. The function `start_db` exports the table *data_score* as database connection `tbl_score` to the package environment `gt.env`. Users can access the database table through `dplyr::tbl`. The sample data included in `data_score` was simulated based on actual Google Trends data.

Example data for the table *data_score* is available as R object `example_score`.

Usage

```
example_score
```

Format

A tibble with 6,000 rows and 8 variables:

location Column of type `character` showing the ISO2 code of the country or region for which the data was computed.

keyword Column of type `character` showing the keyword for which the data was downloaded.

date Column of type `integer` showing the date for which the data was computed Can be transformed into date format with `lubridate::as_date`.

score_obs Column of type `double` showing search score for the respective location-keyword-date combination - no time series adjustment.

score_sad Column of type `double` showing the search score for the respective location-keyword-date combination - seasonally adjusted time series.

score_trd Column of type `double` showing the search score for the respective location-keyword-date combination - trend-only time series.

batch_c Column of type `integer` showing the number of each control batch.

batch_o Column of type `integer` showing the number of each object batch.

synonym Column of type `integer` showing whether the line will be aggregated as synonym.

See Also

- `compute_score()`
- `compute_voi()`
- `dplyr::tbl()`

`disconnect_db` *Disconnect from database*

Description

The function closes the connection to the database file *db/globaltrends_db.sqlite* in the working directory.

Usage

```
disconnect_db()
```

Value

Message that disconnection was successful.

Warning

SQLite databases only allow one writer at any instant in time. To run parallel downloads use one database for each download client and merge them once all downloads are complete.

See Also

- `initialize_db()`
- `start_db()`

Examples

```
## Not run:
disconnect_db()

## End(Not run)
```

`download_control` *Download data for control keywords*

Description

The function downloads search volumes from Google Trends for a *control* batch in a set of *locations*. Data is automatically written to table *data_control*. For `download_control_global` the input *location* is automatically set to *world*.

Usage

```
download_control(control, locations = gt.env$countries, ...)

## S3 method for class 'numeric'
download_control(control, locations = gt.env$countries, ...)

## S3 method for class 'list'
download_control(control, locations = gt.env$countries, ...)

download_control_global(control, ...)
```

Arguments

`control`	Control batch for which the data is downloaded. Object of type `numeric` or object of type `list` containing single objects of type `numeric`.
`locations`	List of countries or regions for which the data is downloaded. Refers to lists generated in `start_db`. Defaults to `gt.env$countries`.
`...`	Arguments that are passed on to the `gtrendsR::gtrends` function.

Details

Downloads through the Google Trends API are made through `gtrendsR::gtrends`. Each control batch can consist of up to five keywords and is predefined in tables *batch_keywords* and *batch_time* through `add_keywords`. The download for a single keyword batch for a single location takes about 30 seconds. This includes a randomized waiting period of 5-10 seconds between downloads. Depending on the frequency of downloads, Google Trends might block users for some time. In this case, `download_control` waits 60 minutes before it retries the download.

Value

Message that data has been downloaded successfully. Data is written to table *data_control*.

Warning

We advise against the usage of *category codes* in downloads. If you use *categories* to narrow the context of keyword usage, these categories are applied to **ALL** keywords in the batch. This applies to *control* keywords as well as *object* keywords and can result in unintended behavior.

See Also

- `example_control()`
- `gtrendsR::gtrends()`

Examples

```
## Not run:
download_control(
  control = 1,
  locations = countries
)
download_control(
  control = as.list(1:5),
  locations = countries
)

## End(Not run)
```

---

download_object *Download data for object batch*

Description

The function downloads search volumes from Google Trends for an object batch (*batch_o*) and one keyword from a control batch (*batch_c*) in a set of *locations*. Data is automatically written to table *data_object*. For `download_object_global` the input *location* is automatically set to *world*.

Usage

```
download_object(object, control = 1, locations = gt.env$countries, ...)

## S3 method for class 'numeric'
download_object(object, control = 1, locations = gt.env$countries, ...)

## S3 method for class 'list'
download_object(object, control = 1, locations = gt.env$countries, ...)

download_object_global(object, control = 1, ...)
```

Arguments

`object`	Object batch for which the data is downloaded. Object of type `numeric` or object of type `list` containing single object of type `numeric`.
`control`	Control batch that is used for mapping. Object of type `numeric`. Defaults to `1`.
`locations`	List of countries or regions for which the data is downloaded. Refers to lists generated in `start_db`. Defaults to `countries`.
`...`	Arguments that are passed on to the `gtrendsR::gtrends` function.

Details

Downloads through the Google Trends API are made through `gtrendsR::gtrends`. Each object batch can consist of up to four keywords and is predefined in tables *batch_keywords* and *batch_time* through `add_keywords`. In addition, one control keyword is added to each object batch. The control keyword then allows a mapping between search volumes for control keywords stored in *data_control* and search volumes for object keywords. The download for a single keyword batch for a single location takes about 30 seconds. This includes a randomized waiting period of 5-10 seconds between downloads. Depending on the frequency of downloads, Google Trends might block users for some time. In this case, `download_object` waits 60 minutes before it retries the download.

Value

Message that data was downloaded successfully. Data is written to table *data_object*.

Warning

We advise against the usage of *category codes* in downloads. If you use *categories* to narrow the context of keyword usage, these categories are applied to **ALL** keywords in the batch. This applies to *control* keywords as well as *object* keywords and can result in unintended behavior.

See Also

- `example_object()`
- `gtrendsR::gtrends()`

Examples

```
## Not run:
download_object(
  object = 1,
  locations = countries
)
download_object(
  object = as.list(1:5),
  locations = countries
)

## End(Not run)
```

`export_control` *Export data from database table*

Description

The function allows to export data from database tables. In combination with various *write* functions in R, the functions allow exports from the database to local files.

Usage

```
export_control(control = NULL, location = NULL)

export_control_global(control = NULL)
```

```
export_object(keyword = NULL, object = NULL, control = NULL, location = NULL)

export_object_global(keyword = NULL, object = NULL, control = NULL)

export_score(keyword = NULL, object = NULL, control = NULL, location = NULL)

export_voi(keyword = NULL, object = NULL, control = NULL)

export_doi(
  keyword = NULL,
  object = NULL,
  control = NULL,
  locations = NULL,
  type = c("obs", "sad", "trd")
)
```

Arguments

`control`	Control batch number for which data should be exported. Only for `export_control` and `export_control_global`, input is also possible as list.
`location`	List of locations for which the data is exported. Refers to lists generated in `start_db` or `character` objects in these lists. Only for `export_control`, `export_object`, or `export_score`.
`keyword`	Object keywords for which data should be exported. Object or list of objects of type `character`.
`object`	Object batch number for which data should be exported.
`locations`	List of locations for which the data is exported. Refers to names of lists generated in `start_db` as an object of type `character`. Only for `export_doi`.
`type`	Type of time series for which data should be exported. Element of type `character`. Relevant only for `export_global` and `export_doi`. Takes one of the following values: *obs* - observed search scores, *sad* - seasonally adjusted search scores, *trd* - trend only search scores.

Details

Exports can be filtered by *keyword*, *object*, *control*, *location*, *locations*, or *type*. Not all filters are applicable for all functions. When filter *keyword* and *object* are used together, *keyword* overrules *object*. When supplying `NULL` as input, no filter is applied to the variable.

Value

The functions export and filter the respective database tables.

- `export_control` and `export_control_global` export data from table *data_control`with columns location, keywo` port_objectandexport_object_global `export data from table *data_object` with columns location, keyword, date, hits, object. Object of class `"data.frame"`. Methods are applied based on input *keyword*.

- `export_score` exports data from table *data_scorewith columns `location, keyword, date, score_obs, score_sa` "data.frame"). Methods are applied based on input *keyword*. \item export_voi exports data from table * with columns keyword, date, hits, control, filters for `location == "world"`. Object of class `c("exp_voi", "data.frame")`. Methods are applied based on input *keyword*.
- `export_doi` exports data from table *data_doiwith columns `keyword, date, type, gini, hhi, entropy, control` "data.frame")‘. Methods are applied based on input *keyword*.

See Also

- `example_control()`
- `example_object()`
- `example_score()`
- `example_doi()`

Examples

```
## Not run:
export_control(control = 2)

export_object(
  keyword = "manchester united",
  locations = countries
)

export_object(
  keyword = c("manchester united", "real madrid")
)

export_object(
  keyword = list("manchester united", "real madrid")
)

export_score(
  object = 3,
  control = 1,
  location = us_states
) %>%
  readr::write_csv("data_score.csv")

export_doi(
  keyword = "manchester united",
  control = 2,
  type = "sad",
  locations = "us_states"
) %>%
  writexl::write_xlsx("data_doi.xlsx")

## End(Not run)
```

get_abnorm_hist *Compute abnormal changes in data - historic baseline*

Description

The function allows to compute changes in search scores, voi, and doi and shows percentile of changes to identify abnormal changes. In combination with various *write* functions in R, the functions allow exports from the database to local files.

Usage

```
get_abnorm_hist(data, ...)

## S3 method for class 'exp_score'
get_abnorm_hist(
  data,
  train_win = 12,
  train_break = 0,
  type = c("obs", "sad", "trd"),
  ...
)

## S3 method for class 'exp_voi'
get_abnorm_hist(
  data,
  train_win = 12,
  train_break = 0,
  type = c("obs", "sad", "trd"),
  ...
)

## S3 method for class 'exp_doi'
get_abnorm_hist(
  data,
  train_win = 12,
  train_break = 0,
  measure = c("gini", "hhi", "entropy"),
  ...
)
```

Arguments

`data`	Object of class `exp_score`, `exp_voi` or `exp_doi` generated through `export_...` functions.
`...`	Further arguments passed to or from other methods.
`train_win`	Object of type `numeric`. Length of rolling average training window in months. Defaults to 12.

`train_break`	Object of type `numeric`. Length of break between rolling average training window and date in months. Defaults to 1.
`type`	Object of type `character` indicating the type of time series-column from data_score, takes either *obs*, *sad*, or *trd*. Defaults to *"obs"*.
`measure`	Object of type `character` indicating the measure used for DOI computation for which abnormal changes should be analyzed. Takes either *gini*, *hhi*, or *entropy*. Defaults to *"gini"*.

Details

The function computes abnormal changes in search scores, VOI, or DOI for each date. We define "abnormal" in terms of deviation from a historic baseline value. To compute the historic baseline value, the function computes a moving average. Users can specify the window for moving average training `train_win` and a break between training and the given date `train_break`. Abnormal changes are the difference between the moving average and the respective search score, VOI, or DOI. To highlight abnormal changes, the function computes a historic percentile rank for each abnormal change within the distribution of abnormal changes. Low percentile ranks signify abnormally high negative changes. High percentile ranks signify abnormally high positive changes. The function uses the output from `export_...` functions as input. As `get_abnorm_hist` offers no additional filters, users are advised to use filters in the `export_...` functions or to pre-process data before using `get_abnorm_hist`.

Value

The functions export and filter the respective database tables and return objects of class `"tbl_df", "tbl", "data.frame"`.

- Input class `exp_score` computes abnormal changes in search scores with columns keyword, location, date, control, object, score, score_abnorm, quantile. Object of class `c("abnorm_score", "data.frame")`.
- Input class `exp_voi` computes abnormal changes in VOI with columns keyword, date, control, object, voi, voi_abnorm, quantile. Object of class `c("abnorm_voi", "data.frame")`.
- Input class `exp_doi` computes abnormal changes in DOI with columns keyword, locations, date, control, object, doi, doi_abnorm, quantile. Object of class `c("abnorm_doi", "data.frame")`.

See Also

- `export_score()`
- `export_voi()`
- `export_doi()`
- `dplyr::filter()`

Examples

```
## Not run:
data <- export_score(keyword = "amazon")
get_abnorm_hist(data, train_win = 12, train_break = 0, type = "obs")

data <- export_voi(keyword = "amazon")
```

```
get_abnorm_hist(data, train_win = 12, train_break = 0, type = "obs")

data <- export_score(keyword = "amazon")
get_abnorm_hist(data, train_win = 12, train_break = 0, measure = "gini")

## End(Not run)
```

gt.env *globaltrends package environment*

Description

The environment `gt.env` contains all package-related data objects, such as the handle for the SQLite database file or connections to tables. The object contains:

- globaltrends_db: Handle for the SQLite database file.
- tbl_locations: Connection to table that contains the lists of locations saved in the database.
- tbl_keywords: Connection to table that contains the lists of keywords saved in the database.
- tbl_time: Connection to table that contains the lists of batch times saved in the database.
- tbl_synonyms: Connection to table that contains the lists of keyword synonyms saved in the database.
- tbl_doi: Connection to table that contains the DOI data saved in the database.
- tbl_control: Connection to table that contains data on search volume for control terms saved in the database.
- tbl_object: Connection to table that contains data on search volume for object terms saved in the database.
- tbl_score: Connection to table that contains data on search scores saved in the database.
- keywords_control: Tibble that contains all keywords per control batch.
- time_control: Tibble that contains all batch times per control batch.
- keywords_object: Tibble that contains all keywords per object batch.
- time_object: Tibble that contains all batch times per object batch.
- keyword_synonyms: Tibble that contains all keyword/synonym combinations.

Usage

```
gt.env
```

Format

An object of class `environment` of length 14.

See Also

- `example_control()`
- `example_object()`
- `example_score()`
- `example_doi()`

`initialize_db` *Initialize database*

Description

The function creates a new database for the `globaltrends` package and creates all necessary tables within the database.

Usage

```
initialize_db()
```

Details

The function creates a new SQLite database for the `globaltrends` package. The database is saved as file *db/globaltrends_db.sqlite* in the working directory. If the folder *db* does not exists in the working directory, the folder is created. If the database already exists in the working directory, the function exits with an error. Within the database all tables are created and the default location sets are added to the respective table:

- *countries* - all countries with a share in global GDP >= 0.1\ in 2018.
- *us_states* - all US federal states and Washington DC.

After creating the database, the function disconnects from the database.

Value

Database is created.

Warning

SQLite databases only allow one writer at any instant in time. To run parallel downloads use one database for each download client and merge them once all downloads are complete.

See Also

- start_db()
- disconnect_db()
- countries()
- us_states()
- example_keywords()
- example_time()
- example_control()
- example_object()
- example_score()
- example_doi()
- https://www.sqlite.org/index.html

Examples

```
## Not run:
initialize_db()

## End(Not run)
```

---

plot_bar *Create barplot for cross-sectional globaltrends data*

Description

The function creates barplots for cross-sectional search score data. It uses the output of `export_score` to prepare a bar plot of search scores for the top 10 countries. For output from `get_abnorm_hist` the plot shows five locations with the highest and lowest abnormal changes each. When the output includes more than one keyword, only the first keyword is used.

Usage

```
plot_bar(data, ...)

## S3 method for class 'exp_score'
plot_bar(data, type = c("obs", "sad", "trd"), ...)

## S3 method for class 'abnorm_score'
plot_bar(data, ...)
```

Arguments

`data`	Data exported from `export_...` or `compute_abnorm` functions.
`...`	Further arguments passed to or from other methods.
`type`	Object of type `character` indicating the type of time series-column from data_score, takes either *obs*, *sad*, or *trd*. Defaults to *"obs"*.

Value

Barplot of cross-sectional data as `ggplot2` object.

Examples

```
## Not run:
data <- export_score(keyword = "amazon")
plot_bar(data, type = "obs")

data <- export_score(keyword = "amazon")
data <- get_abnorm_hist(data, train_win = 12, train_break = 0, type = "obs")
plot_bar(data)

## End(Not run)
```

`plot_box` *Create boxplot for time series of globaltrends data*

Description

The function creates boxplots for time series globaltrends data. It uses the output of `export_...` or `get_abnorm_hist` to prepare boxplots for up to nine keywords.

Usage

```
plot_box(data, ...)

## S3 method for class 'exp_score'
plot_box(data, type = c("obs", "sad", "trd"), ...)

## S3 method for class 'abnorm_score'
plot_box(data, ci = 0.95, ...)

## S3 method for class 'exp_voi'
plot_box(data, type = c("obs", "sad", "trd"), ...)

## S3 method for class 'abnorm_voi'
plot_box(data, ci = 0.95, ...)
```

```
## S3 method for class 'exp_doi'
plot_box(
  data,
  type = c("obs", "sad", "trd"),
  measure = c("gini", "hhi", "entropy"),
  locations = "countries",
  ...
)

## S3 method for class 'abnorm_doi'
plot_box(
  data,
  type = c("obs", "sad", "trd"),
  locations = "countries",
  ci = 0.95,
  ...
)
```

Arguments

`data`	Data exported from `export_...` or `compute_abnorm` functions.
`...`	Further arguments passed to or from other methods.
`type`	Object of type `character` indicating the type of time series-column from data_score, takes either *obs*, *sad*, or *trd*. Defaults to *"obs"*.
`ci`	Confidence interval within which changes are assumed to be normal. Object of type `double, 0 < ci < 1`. Defaults to *0.95*.
`measure`	Object of type `character` indicating the DOI measure, takes either *gini*, *hhi*, or *entropy*. Defaults to *"gini"*.
`locations`	Object of type `character` indicating for which set of locations should be filtered. Defaults to *"countries"*.

Details

For output of `export_score`, only data for a single location is shown. When date for more than one location is provided, the function selects only the first location. For output of `get_abnorm_hist`, users can specify confidence intervals to highlight abnormal changes in the data.

Value

Boxplot of time series as `ggplot2` object. For plots for output from `get_abnorm_hist` the provided confidence interval is indicated by red dots.

Examples

```
## Not run:
data <- export_score(keyword = "amazon")
plot_box(data, type = "obs")
```

```
data <- export_voi(keyword = "amazon")
data <- get_abnorm_hist(data, train_win = 12, train_break = 0, type = "obs")
plot_box(data)

data <- export_doi(keyword = "amazon")
data <- get_abnorm_hist(data, train_win = 12, train_break = 0, measure = "gini")
plot_box(data, ci = 0.9)

## End(Not run)
```

`plot_ts` *Plot time series of globaltrends data*

Description

The function creates line plots for time series globaltrends data. It uses the output of `export_...` or `get_abnorm_hist` to prepare line plots for up to nine keywords.

Usage

```
plot_ts(data, ...)

## S3 method for class 'exp_score'
plot_ts(data, type = c("obs", "sad", "trd"), smooth = TRUE, ...)

## S3 method for class 'abnorm_score'
plot_ts(data, ci = 0.95, ...)

## S3 method for class 'exp_voi'
plot_ts(data, type = c("obs", "sad", "trd"), smooth = TRUE, ...)

## S3 method for class 'abnorm_voi'
plot_ts(data, ci = 0.95, ...)

## S3 method for class 'exp_doi'
plot_ts(
  data,
  type = c("obs", "sad", "trd"),
  measure = c("gini", "hhi", "entropy"),
  locations = "countries",
  smooth = TRUE,
  ...
)

## S3 method for class 'abnorm_doi'
plot_ts(
  data,
  type = c("obs", "sad", "trd"),
  locations = "countries",
  ci = 0.95,
  ...
)
```

Arguments

`data`	Data exported from `export_...` or `compute_abnorm` functions.
`...`	Further arguments passed to or from other methods.
`type`	Object of type `character` indicating the type of time series-column from data_score, takes either *obs*, *sad*, or *trd*. Defaults to *"obs"*.
`smooth`	Object of type `logical` indicating whether the `geom_smooth` function of `ggplot2` should be used. Defaults to `TRUE`.
`ci`	Confidence interval within which changes are assumed to be normal. Object of type `double, 0 < ci < 1`. Defaults to *0.95*.
`measure`	Object of type `character` indicating the DOI measure, takes either *gini*, *hhi*, or *entropy*. Defaults to *"gini"*.
`locations`	Object of type `character` indicating for which set of locations should be filtered. Defaults to *"countries"*.

Details

For output of `export_score`, only data for a single location is shown. When date for more than one location is provided, the function selects only the first location. For output of `get_abnorm_hist`, users can specify confidence intervals to highlight abnormal changes in the data.

Value

Line plot of time series as `ggplot2` object. For plots for output from `get_abnorm_hist` the provided confidence interval is indicated by red dots.

Examples

```
## Not run:
data <- export_score(keyword = "amazon")
plot_ts(data, type = "obs")

data <- export_voi(keyword = "amazon")
data <- get_abnorm_hist(data, train_win = 12, train_break = 0, type = "obs")
plot_ts(data)

data <- export_doi(keyword = "amazon")
data <- get_abnorm_hist(data, train_win = 12, train_break = 0, measure = "gini")
plot_ts(data, ci = 0.9)

## End(Not run)
```

plot_voi_doi — *Line plots of VOI and DOI time series*

Description

The function uses the outputs of `export_voi` and `export_doi` to prepare a parallel time series plot of volume and degree of internationalization values. When the output includes more than one keyword, only the first keyword is used.

Usage

```
plot_voi_doi(
  data_voi,
  data_doi,
  type = c("obs", "sad", "trd"),
  measure = c("gini", "hhi", "entropy"),
  locations = "countries",
  smooth = TRUE
)
```

Arguments

`data_voi`	Data exported from `export_voi` function.
`data_doi`	Data exported from `export_doi` function.
`type`	Object of type `character` indicating the type of time series-column from data_score, takes either *obs*, *sad*, or *trd*. Defaults to *"obs"*.
`measure`	Object of type `character` indicating the DOI measure, takes either *gini*, *hhi*, or *entropy*. Defaults to *"gini"*.
`locations`	Object of type `character` indicating for which set of locations should be filtered. Defaults to *"countries"*.
`smooth`	Object of type `logical` indicating whether the `geom_smooth` function of `ggplot2` should be used. Defaults to `TRUE`.

Value

Line plot of VOI and DOI time series as `ggplot2` object.

See Also

- `export_voi()`
- `export_doi()`
- `ggplot2::ggplot()`

Examples

```
## Not run:
data1 <- export_voi(keyword = "manchester united")
data2 <- export_doi(
  keyword = "manchester united",
  locations = "countries"
)
plot_voi_doi(
  data_voi = data1,
  data_doi = data2,
  type = "obs",
  measure = "gini",
  smooth = TRUE
)
plot_voi_doi(
  data_voi = data1,
  data_doi = data2,
  type = "sad",
  measure = "hhi",
  smooth = FALSE
)
plot_voi_doi(
  data_voi = data1,
  data_doi = data2,
  type = "trd",
  measure = "entropy",
  smooth = TRUE
)

## End(Not run)
```

`remove_data` *Remove data from database tables*

Description

The function removes data from database tables for control or object batches.

Usage

```
remove_data(table, control = NULL, object = NULL)

vacuum_data()
```

Arguments

`table`	Database table from which the batch should be removed. Object of type `character`.
`control`	Control batch for which the data is removed Object of type `numeric`.
`object`	Object batch for which the data is removed Object of type `numeric`.

Details

The function removes data "greedily": all data that builds on the deleted data is removed. For example, when data from *data_control* is removed data in *data_object* that maps to this control batch is also removed. The dependency structure works as follows: *batch_keyword* / *batch_time* -> *data_control* -> *data_object* -> *data_score* -> *data_doi*.

After using `remove_data`, run `vacuum_data` to free-up unused memory in the database file. Depending on the database size, `vacuum_data` might take some minutes for execution.

Value

Message that data has been removed successfully. Data is removed from database tables.

See Also

- `example_keywords()`
- `example_time()`
- `example_control()`
- `example_object()`
- `example_score()`
- `example_doi()`

Examples

```
## Not run:
remove_data(
  table = "batch_keywords",
  control = 1
)
remove_data(
  table = "data_score",
  control = 1,
  object = 1
)
vacuum_data()

## End(Not run)
```

`start_db` *Load globaltrends database and tables*

Description

The function connects to the database file *db/globaltrends_db.sqlite* in the working directory. After connecting to the database connections to the database tables (through `dplyr::tbl`) are created. Data from the tables *batch_keywords* and *batch_time* are exported to the `tibble` objects *keywords_control*, *keywords_object*, *time_control*, and *time_object*.

Usage

```
start_db()
```

Value

The function exports the following objects to the package environment `globaltrends_db`:

- globaltrends_db A DBIConnection object, as returned by `DBI::dbConnect()`, connecting to the SQLite database in the working directory
- tbl_doi A remote data source pointing to the table *data_doi* in the connected SQLite database
- tbl_control A remote data source pointing to the table *data_control* in the connected SQLite database
- tbl_mapping A remote data source pointing to the table *data_mapping* in the connected SQLite database
- tbl_object A remote data source pointing to the table *data_object* in the connected SQLite database
- tbl_score A remote data source pointing to the table *data_score* in the connected SQLite database
- countries A `character` vector containing ISO2 country codes of countries that add at least 0.1\
- us_states A `character` vector containing ISO2 regional codes of US states
- keywords_control A `tibble` containing keywords of control batches
- time_control A `tibble` containing times of control batches
- keywords_object A `tibble` containing keywords of object batches
- time_object A `tibble` containing times of control batches
- keyword_synonyms A `tibble` containing synonymous keywords

Warning

SQLite databases only allow one writer at any instant in time. To run parallel downloads use one database for each download client and merge them once all downloads are complete.

See Also

- `initialize_db()`
- `disconnect_db()`
- `dplyr::tbl()`

Examples

```
## Not run:
start_db()

## End(Not run)
```

`us_states` *us_states*

Description

A character vector that includes ISO2 codes for all US federal states and Washington DC. The data includes:

- US-AL
- US-AK
- US-AZ
- US-AR
- US-CA
- US-CO
- US-CT
- US-DE
- US-FL
- US-GA
- US-HI
- US-ID
- US-IL
- US-IN
- US-IA
- US-KS
- US-KY
- US-LA
- US-ME
- US-MD
- US-MA
- US-MI
- US-MN
- US-MS
- US-MO
- US-MT
- US-NE
- US-NV
- US-NH
- US-NJ

- US-NM
- US-NY
- US-NC
- US-ND
- US-OH
- US-OK
- US-OR
- US-PA
- US-RI
- US-SC
- US-SD
- US-TN
- US-TX
- US-UT
- US-VT
- US-VA
- US-WA
- US-WV
- US-WI
- US-WY
- US-DC

Usage

```
us_states
```

Format

An object of class `character` of length 51.

Index