Package ‘PytrendsLongitudinalR’

September 17, 2024

Type Package

Title Create Longitudinal Google Trends Data

Version 0.1.4

Description 'Google Trends' provides cross-sectional and time-series data on searches, but lacks readily available longitudinal data. Researchers, who want to create longitudinal 'Google Trends' on their own, face practical challenges, such as normalized counts that make it difficult to combine cross-sectional and time-series data and limitations in data formats and timelines that limit data granularity over extended time periods.
This package addresses these issues and enables researchers to generate longitudinal 'Google Trends' data.
This package is built on 'pytrends', a Python library that acts as the unofficial 'Google Trends API' to collect 'Google Trends' data. As long as the 'Google Trends API', 'pytrends' and all their dependencies are working, this package will work.
During testing, we noticed that for the same input (keyword, topic, data_format, timeline), the output index can vary from time to time. Besides, if the keyword is not very popular, then the resulting dataset will contain a lot of zeros, which will greatly affect the final result. While this package has no control over the accuracy or quality of 'Google Trends' data, once the data is created, this package coverts it to longitudinal data.
In addition, the user may encounter a 429 Too Many Requests error when using cross_section() and time_series() to collect 'Google Trends' data. This error indicates that the user has exceeded the rate limits set by the 'Google Trends API'. For more information about the 'Google Trends API' - 'pytrends', visit <https://pypi.org/project/pytrends/>.

Encoding UTF-8

Imports lubridate, jsonlite, reticulate, utils

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

License MIT + file LICENSE

SystemRequirements Python (>= 3.7), pip (for installing Python packages), pytrends, pandas, requests

Config/reticulate list(virtualenv = ``pytrends-in-r-new", packages = list( list(package = ``pandas"), list(package = ``requests"), list(package = ``pytrends")))

NeedsCompilation no

Author Taeyong Park [cre, cph, aut], Malika Dixit [aut]

Maintainer Taeyong Park <taeyongp@andrew.cmu.edu>

Repository CRAN

Date/Publication 2024-09-17 09:30:11 UTC

Contents

concat_time_series 2
convert_cross_section 4
cross_section 6
initialize_request_trends 7
install_pytrendslongitudinalr 9
time_series 9

Index 11

`concat_time_series` *Concatenate Multiple Time-Series Data Sets*

Description

This function concatenates the time-series data collected by the 'time_series()' function.

Usage

```
concat_time_series(params, reference_geo_code = "US", zero_replace = 0.1)
```

Arguments

`params`	A list containing parameters including keyword, topic, folder_name, start_date, end_date, and data_format.
`reference_geo_code`	Google Trends Geo code for the user-selected reference region. For example, UK's Geo is 'GB', Central Denmark Region's Geo is 'DK-82, and US DMA Philadelphia PA's Geo is '504'. The default is 'US'.
`zero_replace`	When re-scaling data from different time periods for concatenation, the last/first data point of a time period may be zero. Then the calculation will throw an error, or every single data point will be zero. To avoid this, the user can adjust the zero to an insignificant number to continue the calculation. The default is 0.1.

Details

This method concatenates the reference time-series data collected by the 'time_series()' function when the function has produced more than one data file. Because the time series data of each time period is normalized, the multiple time-series data sets are not on the same scale and must be re-scaled. The re-scaled reference time-series data will be used in the next step to re-scale the cross-section data. If the given period is less than 269 days/weeks/months, and the 'time_series()' function produced only one data file, concatenation is unnecessary, and thus no concatenated file will be created in this step. The user can move to the 'convert_cross_section()' function without any problems.

Value

No return value, called for side effects. The function concatenates the time-series data and saves it as a CSV file.

Examples

```
# Please note that this example may take a few minutes to run
# Create a temporary folder for the example

# Ensure the temporary folder is cleaned up after the example
if (reticulate::py_module_available("pytrends")) {
  params <- initialize_request_trends(
    keyword = "Coronavirus disease 2019",
    topic = "/g/11j2cc_qll",
    folder_name = file.path(tempdir(), "test_folder"),
    start_date = "2017-12-31",
    end_date = "2024-05-19",
    data_format = "weekly"
  )
  result <- TRUE

  # Run the time_series function and handle TooManyRequestsError
  tryCatch({
    time_series(params, reference_geo_code = "US-CA")
  }, error = function(e) {
    message("An error occurred: ", conditionMessage(e))
    result <- FALSE # Indicate failure only on error
  })

  # Check if at least one file is present in the expected directory
  data_dir <- file.path("test_folder", "weekly", "over_time", "US-CA")
  if (result && length(list.files(data_dir)) > 0) {
    concat_time_series(params, reference_geo_code = "US-CA")
  } else {
    if (result) {
    message("Skipping concat_time_series because no files were found in the expected directory.")
    } else {
      message("Skipping concat_time_series because time_series failed.")
    }
    result <- FALSE
  }

  # Clean up temporary directory
  on.exit(unlink("test_folder", recursive = TRUE))
} else {
  message("The 'pytrends' module is not available.
  Please install it by running install_pytrendslongitudinalr()")
}
```

convert_cross_section *Convert the Cross-Section data for Re-scaling.*

Description

This function uses the single or concatenated reference time-series data to re-scale the cross-section data collected by the cross_section() function.

Usage

```
convert_cross_section(params, reference_geo_code = "US-CA", zero_replace = 0.1)
```

Arguments

`params`	A list containing parameters including keyword, topic, folder_name, start_date, end_date, and data_format.
`reference_geo_code`	Google Trends Geo code for the user-selected reference region. For example, UK's Geo is 'GB', Central Denmark Region's Geo is 'DK-82, and US DMA Philadelphia PA's Geo is '504'. The default is 'US'.
`zero_replace`	When re-scaling data from different time periods for concatenation, the last/first data point of a time period may be zero. Then the calculation will throw an error, or every single data point will be zero. To avoid this, the user can adjust the zero to an insignificant number to continue the calculation. The default is 0.1.

Details

This final method rescales the cross-section data based on the concatenated time series data to generate re-scaled accurate longitudinal Google Trends index.

Value

No return value, called for side effects.

Examples

```
# Please note that this example may take a few minutes to run
# Create a temporary folder for the example

# Ensure the temporary folder is cleaned up after the example
if (reticulate::py_module_available("pytrends")) {
  params <- initialize_request_trends(
    keyword = "Coronavirus disease 2019",
    topic = "/g/11j2cc_qll",
    folder_name = file.path(tempdir(), "test_folder"),
    start_date = "2024-05-01",
    end_date = "2024-05-03",
    data_format = "daily"
  )
  cross_section_success <- TRUE
  time_series_success <- TRUE

  # Run the cross_section function and handle potential errors
  tryCatch({
    cross_section(params, geo = "US", resolution = "REGION")
  }, pytrends.exceptions.TooManyRequestsError = function(e) {
    message("Too many requests error in cross_section: ", conditionMessage(e))
    cross_section_success <- FALSE # Indicate failure
  })

  # Run the time_series function and handle potential errors
  tryCatch({
    time_series(params, reference_geo_code = "US-CA")
  }, pytrends.exceptions.TooManyRequestsError = function(e) {
    message("Too many requests error in time_series: ", conditionMessage(e))
    time_series_success <- FALSE # Indicate failure
  })

  data_dir_time <- file.path("test_folder", "daily", "over_time", "US-CA")
  data_dir_region <- file.path("test_folder", "daily", "by_region")

  # Conditionally run convert_cross_section only if both functions succeeded
 if (cross_section_success && time_series_success && length(list.files(data_dir_time)) > 0
  && length(list.files(data_dir_region)) > 0) {
    convert_cross_section(params, reference_geo_code = "US-CA")
  } else {
    message("Skipping convert_cross_section due to previous errors.")
  }

  # Clean up temporary directory
  on.exit(unlink("test_folder", recursive = TRUE))
} else {
  message("The 'pytrends' module is not available.
  Please install it by running install_pytrendslongitudinalr()")
}
```

`cross_section` *Collect Cross-Section Google Trends Data*

Description

This function uses the 'pytrends.interest_by_region()' function available in 'pytrends' Python library to collect cross-section Google Trends data and automatically store it in the specified directory.

Usage

```
cross_section(params, geo = "", resolution = "COUNTRY")
```

Arguments

`params`	A list containing parameters including keyword, topic, folder_name, start_date, end_date, and data_format.
`geo`	Country/Region to collect data from. Defaults to Worldwide if empty.
`resolution`	Resolution is a sub-region of the region selected for 'geo' ('COUNTRY', 'REGION', 'CITY', 'DMA'). Defaults to 'COUNTRY'.

Details

This function collects Google Trends data based on the specified parameters and saves it in the following structure: `folder_name/data_format/by_region`. Each file contains data for a specific time period (day/week/month) and geographical region. The filenames include the start and end dates of the data period.

PS: This method may take a long time to complete due to Google Trends API rate limits.

Value

No return value, called for side effects.

Examples

```
# Please note that this example may take a few minutes to run
# Create a temporary folder for the example

# Ensure the temporary folder is cleaned up after the example

if (reticulate::py_module_available("pytrends")) {
  params <- initialize_request_trends(
    keyword = "Coronavirus disease 2019",
    topic = "/g/11j2cc_qll",
    folder_name = file.path(tempdir(), "test_folder"),
    start_date = "2024-05-01",
    end_date = "2024-05-03",
    data_format = "daily"
  )

  # Run the cross_section function with the parameters
  tryCatch({
    cross_section(params, geo = "US", resolution = "REGION")
  }, error = function(e) {
    message("An error occurred: ", e$message)
  })
  on.exit(unlink("test_folder", recursive = TRUE))
} else {
  message("The 'pytrends' module is not available.
  Please install it by running install_pytrendslongitudinalr()")
}
```

`initialize_request_trends`

Initialize Google Trends Request

Description

This function initializes a request to the Google Trends API using pytrends, creates necessary directories, and prepares parameters for data collection.

Usage

```
initialize_request_trends(
  keyword,
  topic = NULL,
  folder_name,
  start_date,
  end_date,
  data_format
)
```

Arguments

`keyword`	The keyword to be used for collecting Google Trends data.
`topic`	The topic associated with the keyword. For example, '/m/0ddwt' will give Google Trends data for Insomnia as topic of 'Disorder'. If identical to the keyword, data will reflect Google Trends search term data. NOTE: URL's have certain codes for special characters. For example, %20 = white space, %2F = / (forward slash) etc.
`folder_name`	Name of the parent folder where all data will be stored.
`start_date`	The start date from which to collect Google Trends data.
`end_date`	The end date until which to collect Google Trends data.
`data_format`	Time basis for the query. Can be one of 'daily', 'weekly', or 'monthly'.

Details

The initiation stage involves creating two folders automatically: - The main folder chosen by the user (‘folder_name‘). - A subfolder corresponding to the ‘data_format‘ (e.g., ’daily’, ’weekly’, ’monthly’) for storing data.

Value

A list containing initialized values and objects for further interaction with the package:

`logger`	A logging object for recording messages.
`keyword`	The keyword used for data collection.
`topic`	The topic associated with the keyword.
`folder_name`	Name of the parent folder for storing data.
`start_date`	Start date for data collection.
`end_date`	End date for data collection.
`data_format`	Time basis for the data query (’daily’, ’weekly’, or ’monthly’).
`num_of_days`	Number of days between `start_date` and `end_date`.
`pytrend`	Initialized pytrends request object.
`time_window`	Optional. Time window parameter, applicable for ’weekly’ data format.
`times`	Optional. Time periods determined for ’weekly’ or ’daily’ data formats.

Examples

```
# Create a temporary folder for the example

# Ensure the temporary folder is cleaned up after the example
if (reticulate::py_module_available("pytrends")) {
  # Run the function with the temporary folder
  params <- initialize_request_trends(
    keyword = "Coronavirus disease 2019",
    topic = "/g/11j2cc_qll",
    folder_name = file.path(tempdir(), "test_folder"),
    start_date = "2024-05-01",
    end_date = "2024-05-03",
    data_format = "daily"
  )
  on.exit(unlink("test_folder", recursive = TRUE))
} else {
  message("The 'pytrends' module is not available.
  Please install it by running install_pytrendslongitudinalr()")
}
```

`install_pytrendslongitudinalr` *Install and Set Up Python Environment for PytrendsLongitudinalR*

Description

This function sets up the Python virtual environment and installs required packages.

Usage

```
install_pytrendslongitudinalr(
  envname = "pytrends-in-r-new",
  new_env = identical(envname, "pytrends-in-r-new"),
  ...
)
```

Arguments

`envname`	Name of the virtual environment.
`new_env`	Checks if virtual environment already exists
`...`	Additional arguments passed to ‘py_install()‘.

Value

No return value, called for side effects. This function sets up the virtual environment and installs required Python packages.

`time_series` *Collect Time-Series Google Trends Data*

Description

This function uses the ’pytrends.interest_over_time()’ function available in ’pytrends’ Python library to collect time-series Google Trends data and automatically store it in the specified directory.

Usage

```
time_series(params, reference_geo_code = "")
```

Arguments

`params`	A list containing parameters including keyword, topic, folder_name, start_date, end_date, and data_format.
`reference_geo_code`	Google Trends Geo code for the user-selected reference region. For example, UK’s Geo is ’GB’, Central Denmark Region’s Geo is ’DK-82, and US DMA Philadelphia PA’s Geo is ’504’. Default is ’US’.

Details

This function collects Google Trends time-series data based on the specified parameters and saves it in the following structure: `folder_name/data_format/over_time/reference_geo_code`. Google Trends provides daily data if the time period between the start and end dates is less than 270 days, weekly data if the time period is between 270 days and 1890 days (270 weeks), and monthly data if it's equal to or greater than 270 weeks.

Value

No return value, called for side effects.

Examples

```
# Create a temporary folder for the example

# Ensure the temporary folder is cleaned up after the example


if (reticulate::py_module_available("pytrends")) {
  params <- initialize_request_trends(
    keyword = "Coronavirus disease 2019",
    topic = "/g/11j2cc_qll",
    folder_name = file.path(tempdir(), "test_folder"),
    start_date = "2024-05-01",
    end_date = "2024-05-03",
    data_format = "daily"
  )
  on.exit(unlink("test_folder", recursive = TRUE))

  # Run the time_series function with the parameters
  tryCatch({
    time_series(params, reference_geo_code = "US-CA")
  }, pytrends.exceptions.TooManyRequestsError = function(e) {
  message("Too many requests error: ", conditionMessage(e))
  })
} else {
  message("The 'pytrends' module is not available.
  Please install it by running install_pytrendslongitudinalr()")
}
```

Index

concat_time_series, 2
convert_cross_section, 4
cross_section, 6

initialize_request_trends, 7
install_pytrendslongitudinalr, 9

time_series, 9