Package 'onemapsgapi'

February 6, 2020

Type Package

Title R Wrapper for the 'OneMap.Sg API'

Version 1.0.0

Maintainer Jolene Lim <jyl2187@columbia.edu>

Description An R wrapper for the 'OneMap.Sg' API <https://docs.onemap.sg/>.
Functions help users query data from the API and return raw JSON data
in ``tidy'' formats. Support is also available for users to retrieve data from multiple API calls
and integrate results into single dataframes, without
needing to clean and merge the data themselves. This package is best suited for users who would
like to perform analyses with Singapore's spatial data without having to perform excessive data cleaning.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports rlang, httr, dplyr, purrr, stringr, tidyr, future, furrr

RoxygenNote 7.0.2

Suggests knitr, rmarkdown, sf, rgdal, sp

VignetteBuilder knitr

NeedsCompilation no

Author Jolene Lim [aut, cre]

Repository CRAN

Date/Publication 2020-02-06 11:00:02 UTC

R topics documented:

`get_planning_areas` *Get Planning Areas (All)*

Description

This function is a wrapper for the Planning Area Polygons API. It returns the data either in raw format or a combined sf or sp object.

Usage

```
get_planning_areas(token, year = NULL, read = NULL)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`year`	Optional, check documentation for valid options. Invalid requests will are ignored by the API.
`read`	Optional, which package to use to read geojson object. For "sf" objects, specify `read = "sf"` and for "sp" objects use `read = "rgdal"`. Note that if used, any missing geojson objects will be dropped (this affects the "Others" planning area returned by the API).

Value

If the parameter `read` is not specified, the function returns a raw JSON object with planning names and geojson string vectors.

If `read = "sf"`, the function returns a single "sf" dataframe with 2 columns: "name" (name of planning area) and "geometry", which contains the simple features.

If `read = "rgdal"`, the function returns a SpatialPolygonsDataFrame of "sp" class. The names of each planning area is recorded in the "name" column of the dataframe.

If an error occurs, the function returns NULL and a warning message is printed.

Note

The `read = "rgdal"` option will take a longer time to return an output than the `read = "sf"` option.

If `read` is specified, any missing geojson objects will be dropped (this affects the "Others" planning area returned by the API). The returned outputs are NOT projected.

If the user specifies a `read` method but does not have the corresponding package installed, the function will return the raw JSON and print a warning message.

Examples

```
# returns raw JSON object
## Not run: get_planning_areas(token)
## Not run: get_planning_areas(token, 2008)

# returns dataframe of class "sf"
## Not run: get_planning_areas(token, read = "sf")

# returns SpatialPolygonsDataFrame ("sp" object)
## Not run: get_planning_areas(token, read = "rgdal")

# error: output is NULL, warning message shows status code
## Not run: get_planning_areas("invalid_token")
```

`get_planning_names` *Get Planning Area Names*

Description

This function is a wrapper for the Names of Planning Area API. It returns the data as a tibble.

Usage

```
get_planning_names(token, year = NULL)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`year`	Optional, check documentation for valid options. Invalid requests will are ignored by the API.

Value

A tibble with 2 columns:

id Planning area id

pln_area_n Planning area name

Examples

```
# returns tibble
## Not run: get_planning_names(token)
## Not run: get_planning_names(token, 2008)

# error: output is NULL, warning message shows status code
## Not run: get_planning_names("invalid_token")
```

`get_planning_polygon` *Get Planning Polygon for a Specific Point*

Description

This function is a wrapper for the Planning Area Query API. It returns the spatial polygon data matching the specified location point, either in raw format, as an sf or sp object.

Usage

```
get_planning_polygon(token, lat, lon, year = NULL, read = NULL)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`lat`	Latitude of location point
`lon`	Longitude of location point
`year`	Optional, check documentation for valid options. Invalid requests will are ignored by the API.
`read`	Optional, which package to use to read geojson object. For "sf" objects, specify `read = "sf"` and for "sp" objects use `read = "rgdal"`.

Value

If the parameter `read` is not specified, the function returns a raw JSON object a list containing the planning area name and a geojson string representing the polygon.

If `read = "sf"`, the function returns a 1 x 2 "sf" dataframe: "name" (name of planning area) and "geometry", which contains the simple feature.

If `read = "rgdal"`, the function returns a SpatialPolygonsDataFrame of "sp" class. The names of the planning area is recorded in the "name" column of the dataframe.

If an error occurs, the function returns NULL and a warning message is printed.

Note

If the user specifies a `read` method but does not have the corresponding package installed, the function will return the raw JSON and print a warning message.

Examples

```
# returns raw JSON object
## Not run: get_planning_polygon(token)
## Not run: get_planning_polygon(token, 2008)

# returns dataframe of class "sf"
## Not run: get_planning_polygon(token, read = "sf")

# returns SpatialPolygonsDataFrame ("sp" object)
## Not run: get_planning_polygon(token, read = "rgdal")

# error: output is NULL, warning message shows status code
## Not run: get_planning_polygon("invalid_token")
## Not run: get_planning_polygon(token, "invalidlat", "invalidlon")
```

`get_pop_queries` *Get Population Data (Multiple)*

Description

This function is a wrapper for the Population Query API. It allows for querying of multiple Pop-query data types for multiple towns and years.

Usage

```
get_pop_queries(
  token,
  data_types,
  planning_areas,
  years,
  gender = NULL,
  parallel = FALSE
)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`data_types`	Type of data to be retrieved, should correspond to one of the API endpoints. E.g. to get economic status data, `data_type = "getEconomicStatus"`. The API endpoints can be found on the documentation page.
`planning_areas`	Town for which the data should be retrieved.
`years`	Year for which the data should be retrieved.
`gender`	Optional, if specified only records for that gender will be returned. This parameter is only valid for the `"getEconomicStatus"`, `"getEthnicGroup"`, `"getMaritalStatus"` and `"getPopulationAgeGroup"` endpoints. If specified for other endpoints, the parameter will be dropped.

`parallel` Default = `FALSE`. Whether to run API calls in parallel or sequentially (default). Enabling parallel iterations is highly recommended for when querying multiple data types/years/towns.

Value

A tibble with each row representing a town in a particular year for a particular gender, and columns with the variables returned by the API endpoint. If any API call returns no data, the values will be NA but the row will be returned. However, if all data_types do not return data for that town and year, no row will be returned for it.

Examples

```
# output with no NA
## Not run: get_pop_queries(token, c("getOccupation", "getLanguageLiterate"),
    c("Bedok", "Yishun"), "2010")
## End(Not run)
## Not run: get_pop_queries(token, c("getEconomicStatus", "getEthnicGroup"),
    "Yishun", "2010", "female")
## End(Not run)

## note behaviour if data types is a mix of those that accept gender params
### only total will have all records
## Not run: get_pop_queries(token, c("getEconomicStatus", "getOccupation", "getLanguageLiterate"),
    "Bedok", "2010")
## End(Not run)
### data type that does not accept gender params will be in gender = Total
## Not run: get_pop_queries(token, c("getEconomicStatus", "getOccupation", "getLanguageLiterate"),
    "Bedok", "2010", gender = "female")
## End(Not run)

# output with some town-year queries without record due to no data
# warning message will show data_type/town/year/gender for which an error occurred
## Not run: get_pop_queries(token, c("getEconomicStatus", "getOccupation"),
    "Bedok", c("2010", "2012"))
## End(Not run) # no records for 2012
```

`get_pop_query` *Get Population Data*

Description

This function is a wrapper for the Population Query API. It only allows for querying of one data type (i.e. one of the API endpoints) for a particular town and year.

Usage

```
get_pop_query(token, data_type, planning_area, year, gender = NULL)
```

Arguments

token	User's API token. This can be retrieved using get_token
data_type	Type of data to be retrieved, should correspond to one of the API endpoints. E.g. to get economic status data, data_type = "getEconomicStatus". The API endpoints can be found on the documentation page.
planning_area	Town for which the data should be retrieved.
year	Year for which the data should be retrieved.
gender	Optional, if specified only records for that gender will be returned. This parameter is only valid for the "getEconomicStatus", "getEthnicGroup", "getMaritalStatus" and "getPopulationAgeGroup" endpoints. If specified for other endpoints, the parameter will be dropped.

Value

A tibble with 1 row and values for all the corresponding variables returned by the API endpoint. If gender is not specified for endpoints with a gender parameter, records for total, male and female will be returned. The notable exception to this is for the "getEthnicGroup" endpoint, which only returns the total record if gender is not specified. This is because by default, this is the only API endpoint with a gender parameter that does not return gender breakdown by default. If an error occurs, the function will return a NULL value

---

get_route *Get Route Information*

Description

This function is a wrapper for the Route Service API. It returns the full route data in a tibble format, or a list of 2 tibbles with results and status information if desired.

Usage

```
get_route(
  token,
  start,
  end,
  route,
  date = Sys.Date(),
  time = format(Sys.time(), format = "%T"),
  mode = NULL,
  max_dist = NULL,
  n_itineraries = 3,
  status_info = FALSE
)
```

Arguments

token	User's API token. This can be retrieved using get_token
start	Vector of c(lat, lon) coordinates for the route start point
end	Vector of c(lat, lon) coordinates for the route end point
route	Type of route. Accepted values are walk, drive, pt (public transport), or cycle
date	Default = current date. Date for which route is requested.
time	Default = current time. Time for which route is requested.
mode	Required if route = "pt". Accepted values are transit, bus or rail
max_dist	Optional if route = "pt". Maximum walking distance
n_itineraries	Optional if route = "pt". Default = 3. The number of potential routes to provide.
status_info	Default = FALSE. Whether to return output as a list including a list of status information and a tibble of output

Value

If no error occurs and status_info = TRUE:

status_info A list containing information about the query status. If route = "pt", the output contains lists request_params, debug_output and elevation. Else, the list contains the variables status and status_msg

result A tibble containing the data retrieved from the query. This is the only output if status_info = FALSE. Each row is an itinerary. Output dimensions vary between route = "pt" and other routes

If an error occurs, the output will be NULL, along with a warning message.

Examples

```
# returns output tibble
## Not run: get_route(token, c(1.319728, 103.8421), c(1.319728905, 103.8421581), "drive")
## Not run: get_route(token, c(1.319728, 103.8421), c(1.319728905, 103.8421581), "pt",
    mode = "bus", max_dist = 300, n_itineraries = 2)
## End(Not run)

# returns list of status list and output tibble
## Not run: get_route(token, c(1.319728, 103.8421), c(1.319728905, 103.8421581),
    "drive", status_info = TRUE)
## End(Not run)

# error: output is NULL, warning message shows status code
## Not run: get_route("invalid_token", c(1.319728, 103.8421), c(1.319728905, 103.8421581), "drive")

# error: output is NULL, warning message shows error message from request
## Not run: get_route(token, c(300, 300), c(400, 500), "cycle")
## Not run: get_route(token, c(1.319728, 103.8421), c(1.319728905, 103.8421581), "fly")
```

get_summ_route *Get Summary Route Information*

Description

This function is a wrapper for the Route Service API. It is similar to `get_route`, except it returns a tibble with only total time and total distance, and also optionally, the start coordinates and end coordinates. If `route = "pt"`, only the best route is chosen (i.e. `n_itineraries = 1`).

Usage

```
get_summ_route(
  token,
  start,
  end,
  route,
  date = Sys.Date(),
  time = format(Sys.time(), format = "%T"),
  mode = NULL,
  max_dist = NULL
)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`start`	Vector of c(lat, lon) coordinates for the route start point
`end`	Vector of c(lat, lon) coordinates for the route end point
`route`	Type of route. Accepted values are `walk`, `drive`, `pt` (public transport), or `cycle`
`date`	Default = current date. Date for which route is requested.
`time`	Default = current time. Time for which route is requested.
`mode`	Required if `route = "pt"`. Accepted values are `transit`, `bus` or `rail`
`max_dist`	Optional if `route = "pt"`. Maximum walking distance

Value

If no error occurs, a tibble of 1 x 2 with the variables:

total_time The total time taken for this route

total_dist The total distance travelled for this route

If an error occurs, the output will be `NA`, along with a warning message.

get_theme *Get Theme Data from OneMap.Sg*

Description

This function is a wrapper for the Retrieve Theme API. It returns the data as cleaned tibbles.

Usage

```
get_theme(token, theme, extents = NULL, return_info = FALSE)
```

Arguments

token	User's API token. This can be retrieved using get_token
theme	OneMap theme in its QUERYNAME format. A tibble of available themes can be retrieved using search_themes
extents	Optional Location Extents for search. This should be in the format "Lat1,%20Lng1,Lat2,%20Lng2". For more information, consult the API Documentation.
return_info	Default = FALSE. If FALSE, function only returns a tibble for query results. If TRUE, function returns output as a list containing a tibble for query information and a tibble for query results.

Value

If no error occurs:

query_info A 1 x 7 tibble containing information about the query. The variables are FeatCount, Theme_Name, Category, Owner, DateTime.date, DateTime.timezone_type, DateTime.timezone

query_result Retuned if return_info = TRUE. A tibble containing the data retrieved from the query. The columns and rows vary depending on theme and user specification, however all tibbles will contain the variables: NAME, DESCRIPTION, ADDRESSPOSTALCODE, ADDRESSSTREETNAME, Lat, Lng, ICON_NAME

If an error occurs, the output will be NULL, along with a warning message. For non-error queries where 0 results are returned, the output will be query_info, along with a warning message.

Examples

```
# returns a tibble of output
## Not run: get_theme(token, "hotels")
## Not run: get_theme(token, "monuments",
    extents = "1.291789,%20103.7796402,1.3290461,%20103.8726032")
## End(Not run)

# returns a list of status tibble and output tibble
## Not run: get_theme(token, "lighting", return_info = TRUE)

# error: output is NULL, warning message shows status code
```

```
## Not run: get_theme("invalid_token", "hotels")

# error: output is NULL, warning message shows error message from request
## Not run: get_theme(token, "non-existent-theme")

# error: output is \code{query_info}, warning message query did not return any records
## Not run: get_theme(token, "ura_parking_lot", "1.291789,%20103.7796402,1.3290461,%20103.8726032")
```

`get_theme_info` *Get Theme Information*

Description

This function is a wrapper for the Get Theme Info API. It returns a named character vector of Theme Name and Query Name.

Usage

```
get_theme_info(token, theme)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`theme`	Query name of theme. Themes' query names can be retrieved using `search_themes`.

Value

A named character vector of Theme Name and Query Name. If an error occurred, the function returns `NULL` along with a warning message.

Examples

```
# returns named character vector
## Not run: get_theme_status(token, "kindergartens")

# returns NULL, warning message shows status code
## Not run: get_theme_status("invalid_token", "blood_bank")

# returns NULL, warning message shows error
## Not run: get_theme_status(token, "invalid_theme")
```

`get_theme_status` *Check Theme Status*

Description

This function is a wrapper for the Check Theme Status API. It returns a named logical indicating if the theme is updated at a specific date.

Usage

```
get_theme_status(
  token,
  theme,
  date = Sys.Date(),
  time = format(Sys.time(), format = "%T")
)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`
`theme`	Query name of theme. Themes' query names can be retrieved using `search_themes`.
`date`	Default = current date. Date to check for updates. Format YYYY-MM-DD
`time`	Default = current time. Time to check for updates. Format: HH:MM:SS:FFFZ

Value

A named logical indicating if the theme is updated at a specific date. If an error occurred, the function returns `NULL` along with a warning message.

Examples

```
# returns named logical
## Not run: get_theme_status(token, "kindergartens")
## Not run: get_theme_status(token, "2020-01-01", "12:00:00", "hotels")

# returns NULL, warning message shows status code
## Not run: get_theme_status("invalid_token", "blood_bank")

# returns NULL, warning message shows error
## Not run: get_theme_status(token, "invalid_theme")
```

get_token *Extract API token from OneMap.Sg*

Description

This function is a wrapper for the OneMap Authentication Service API. It allows users to generate a API token from OneMap.Sg. Using the API requires that users have a registered email address with Onemap.Sg. Users can register themselves using OneMap.Sg's form.

Usage

```
get_token(email, password, hide_message = FALSE)
```

Arguments

email	User's registered email address.
password	User's password.
hide_message	Default = FALSE. Whether to hide message telling user when the token expires.

Value

API token, or NULL if an error occurs. If error occurs, a warning message will be printed with the error code.

Examples

```
## Not run: get_token("user@example.com",  "password")
```

get_travel *Get Travel Time and Distance*

Description

This function is a wrapper for the Route Service API. It takes in a dataframe of start and end coordinates and returns the same dataframe with additional total time and total distance columns. The function also accepts multiple arguments for 'route' and 'pt_mode', allowing users to compare various route options.

Note that if 'as_wide = TRUE' is selected, any columns with identical names as the additional output columns will be overwritten. Also, if as_wide = TRUE, only unique pairs of start and end points should be used. Regardless, using only unique pairs and joining data back is also a generally recommended workflow to reduce computation time.

Usage

```
get_travel(
  token,
  df,
  origin_lat,
  origin_lon,
  destination_lat,
  destination_lon,
  routes,
  date = Sys.Date(),
  time = format(Sys.time(), format = "%T"),
  pt_mode = "TRANSIT",
  pt_max_dist = NULL,
  as_wide = TRUE,
  parallel = FALSE
)
```

Arguments

token	User's API token. This can be retrieved using get_token
df	The input dataframe of start and end coordinates (the dataframe can have additional variables)
origin_lat	Name of the dataframe column with the start point latitude.
origin_lon	Name of the dataframe column with the start point longitude.
destination_lat	Name of the dataframe column with the end point latitude.
destination_lon	Name of the dataframe column with the end point longitude.
routes	Vector of the types of routes desired. Accepted values are walk, drive, pt (public transport), or cycle
date	Default = current date. Date for which route is requested.
time	Default = current time. Time for which route is requested.
pt_mode	Vector of public transport modes required. Default = route = c("transit"). Accepted values are transit, bus or rail
pt_max_dist	Optional if route = "pt". Maximum walking distance
as_wide	Default = TRUE. Whether to return output as a list as a long tibble with each row a route, or a wide tibble with the same number of rows as the input tibble.
parallel	Default = FALSE. Whether to run API calls in parallel or sequentially (default).

Value

Original dataframe with total time and total distance for each route type.

If an error occurs, the output row will be have NAs for the additional variables, along with a warning message.

Examples

```
# sample dataframe
sample <- data.frame(start_lat = c(1.3746617, 1.3567797, 1.3361976, 500),
    start_lon = c(103.8366159, 103.9347695, 103.6957732, 501),
    end_lat = c(1.429443081, 1.380298287, 1.337586882, 601),
    end_lon = c(103.835005, 103.7452918, 103.6973215, 600),
    add_info = c("a", "b", "c", "d"))

# no error, wide format
## Not run: get_travel(token, sample[1:3, ],
    "start_lat", "start_lon", "end_lat", "end_lon",
    routes = c("cycle", "walk"))
## End(Not run)
## Not run: get_travel(token, sample[1:3, ],
    "start_lat", "start_lon", "end_lat", "end_lon",
    routes = c("drive", "pt"), pt_mode = c("bus", "transit"))
## End(Not run)

# no error, long format
## Not run: get_travel(token, sample[1:3, ],
    "start_lat", "start_lon", "end_lat", "end_lon",
    routes = c("walk", "pt"), pt_mode = c("bus", "transit"),
    as_wide = FALSE)
## End(Not run)

# with error
# warning message will show start/end/route/pt_mode for which an error occurred
## Not run: get_travel(token, sample,
    "start_lat", "start_lon", "end_lat", "end_lon",
    routes = c("cycle", "walk"))
## End(Not run)
```

`search_themes` *Search for Themes available on OneMap.Sg*

Description

This function is a wrapper for the Get All Themes Info API. It allows users to get a tibble of all available themes, and their details, in the OneMap.Sg API. It also provides an additional functionality where users can subset their results using search terms.

Usage

```
search_themes(token, ..., more_info = TRUE)
```

Arguments

`token`	User's API token. This can be retrieved using `get_token`

...	Optional Search terms to subset results; results with any of search terms will be returned. Search terms are not case-sensitive.
more_info	Whether more infomation should be queried, default = TRUE. If FALSE, output will contain Theme Name, Query Name and Icon information. If TRUE, output will additionally contain Category and Theme Owner information.

Value

If no error occurs, a tibble with the following variables:

THEMENAME Name of the Theme

QUERYNAME Query name of the Theme

ICON Name of image file used as Icon in OneMap Web Map

CATEGORY Returned only if more_info = TRUE. Topic that Theme relates to, e.g. Health, Sports, Environment, etc.

THEME_OWNER Returned only if more_info = TRUE. Government Agency who Owns the Dataset

If an error occurs, the function returns NULL, along with a warning message.

Examples

```
# valid
## Not run: search_themes(token)
## Not run: search_themes(token, "hdb", "parks")
## Not run: search_themes(token, more_info = FALSE)

# error
## Not run: search_themes("my_invalid_token")
```

Index