Package 'streetscape'

December 19, 2024

Type Package

Title Collect And Investigate Street Views For Urban Science

Version 1.0.4

Description A collection of functions to search and download street view imagery ('Mapilary' <https://www.mapillary.com/developer/api-documentation>) and to extract, quantify, and visualize visual features. Moreover, there are functions provided to generate Qualtrics survey in TXT format using the collection of street views for various research purposes.

License GPL-3

Depends R (>= 4.1)

RoxygenNote 7.3.1

Language en-US

Encoding UTF-8

LazyData true

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Imports rlang, cli, methods, dplyr, sf, sp, httr, reticulate, osmdata, quickPWCR, mapview, SuperpixelImageSegmentation, OpenImageR, pbmcapply, parallelly

Config/reticulate list(packages = list(list(package = ``mapbox_vector_tile", pip = TRUE)))

NeedsCompilation no

Author Xiaohao Yang [aut, cre, cph], Derek Van Berkel [aut], Mark Lindquist [aut]

Maintainer Xiaohao Yang <xiaohaoy@umich.edu>

Repository CRAN

Date/Publication 2024-12-19 22:30:02 UTC

Contents

`available-field-and-filter` *available-field-and-filter*

Description

available_field provides a list of available fields.

available_filter provides a list of available filters.

Usage

```
available_field()

available_filter()
```

Details

Field and Filter List

Value

dataframe, including field names and their descriptions.

dataframe, including filter names and their descriptions.

Note

More information about fields and filter at https://www.mapillary.com/developer/api-documentation

decode_detections *decode_detections*

Description

convert Mapillary object detection into sf polygons

Usage

```
decode_detections(detections_string)
```

Arguments

detections_string	character, an endcoded string of semantic segmentation, for example, "Gmt4AgoGbXB5L=="

Value

sf polygon

Examples

```
detection <- readLines(system.file('detection.txt', package = 'streetscape'))
streetscape::decode_detections(detection)
```

install_mvt *install_mvt*

Description

install_mvt is a wrapped function of py_install in the reticulate package for installing the python package mapbox_vector_tile, which will be installed in a virtual environment - "r-mvt".

Usage

```
install_mvt(envname = "r-mvt", method = "auto")
```

Arguments

envname	The name, or full path, of the environment in which Python packages are to be installed.
method	character, indicating installation method.

Value

None

scdataframe *streetscape dataframe*

Description

streetscape dataframe

Usage

```
data(scdataframe)
```

Format

An object of class "StreetscapeDataFrame"; see see_streetscape_class().

Examples

```
data(scdataframe)
```

see_streetscape_class *see_streetscape_class*

Description

A function to call out help page of StreetscapeDataFrame

Usage

```
see_streetscape_class()
```

Details

see_streetscape_class

Value

No return value, called for side effects

Note

User can also directly use ?StreetscapeDataFrame

Examples

```
see_streetscape_class()
```

Streetscape-class *Streetscape-Class*

Description

The output of strview_search family functions is constructed in this data format - A specialized data frame for streetscape package for initializing the object with streetscape data and extracting and decoding segmentation information of streetscape dataframe.

Fields

`data` A data frame containing metadata of Mapillary street view images

`epsg` A numeric epsg code

Methods

`decodeDetection()` Regenerate a dataframe with decoded segmentation. 'detections' column will be updated and a new column 'segmentation' will be added.

`download_data(path, items)` Download street view images (and segmentations in sf format if applicable)

`get_mask(index)` Convert the semantic segmentation of a street view image from the Streetscape-DataFrame into sf polygons

`gvi(level)` Calculate green view index (GVI) for each collected image by segmenting green pixels and quantifing the percentage in street view images. This method adds a new column of greeness percetage to the dataframe

`mapPreview(maptype = "meta", fields = c())` Plot data points in an ineractive map view

Class Methods

Method list:

- `StreetscapeDataFrame$decodeDetection()`
- `StreetscapeDataFrame$gvi()`
- `StreetscapeDataFrame$get_mask()`
- `StreetscapeDataFrame$mapPreview()`
- `StreetscapeDataFrame$download_data()`

Method `decodeDetection()`:

Usage: `scdataframe$decodeDetection()`

Method `gvi()`:

Usage: `scdataframe$gvi(level = 1)`

Arguments:

`level` numeric, indicating the resolution level of images for calculating the green view index. 1 - the 256px wide thumbnail; 2 - the 1024px wide thumbnail; 3 - the 2048px wide thumbnail; 4 - the original wide thumbnail. The default is level = 1

Method `get_mask()`**:**

Usage: `scdataframe$get_mask(index = 1)`

Arguments:

`index` numeric, the row index of the dataframe of StreetscapeDataFrame class

Method `mapPreview()`**:**

Usage: `scdataframe$mapPreview(maptype = 'meta')`

Arguments:

`maptype` character or character, specifying what type of information to be mapped: 'meta' - image meta, 'seg' - segmentation proportion, and 'gvi' - GVI".

`fields` vector (optional), a vector of fields indicates the information of images to be included for the 'meta' map. The fields of 'id', 'is_pano', 'height', 'width', 'lon', and 'lat' are already included

Method `download_data()`**:**

Usage: `scdataframe$download_data(path = 'path/to/download', items = c('image', 'mask'))`

Arguments:

`path` character, directory for downloading street view images or segmentation masks or both

`items` character or vector, specifying what to download: 'image' - 'original street view image; 'mask' - semantic segmentation (sf objects in .geojson format)"

`StreetscapeDataFrame-class`

Class `"StreetscapeDataFrame"`

Description

"The output of strview_search family functions is constructed in this data format - A specialized data frame for streetscape package for initializing the object with streetscape data and extracting and decoding segmentation information of streetscape dataframe."

Extends

All reference classes extend and inherit methods from `"envRefClass"`.

Fields

`data`**:** Object of class `data.frame` ~~

`epsg`**:** Object of class `numeric` ~~

Methods

`download_data(path, items): ~~`

`get_mask(index): ~~`

`gvi(): ~~`

`decodeDetection(): ~~`

`mapPreview(maptype, fields): ~~`

`initialize(data, epsg): ~~`

Author(s)

"Xiaohao Yang"

Examples

```
showClass("StreetscapeDataFrame")
```

---

strview2qualtrics *strview2qualtrics*

Description

strview2rate: pack street views as a Qualtrics survey file that can be imported to Qualtrics platform

strview2pwc: pack street views as a Qualtrics survey file for pair-wised comparison

Usage

```
strview2rate(df, header, questions, choices, file)

strview2pwc(df, k, header, questions, file)
```

Arguments

`df`	StreetscapeDataFrame
`header`	character, indicating the task for a question. For example, "Please review the following picture:"
`questions`	vector, a list of questions (see details)
`choices`	list, a list of choices (see details) (this is only for strview2rate)
`file`	character, indicating the directory and file name (without extension) for saving the Qualtrics survey file
`k`	numeric, indicating how many street views each street view will be paired with for pair-wised comparison (this is only for strview2pwc)

Details

strview2qualtrics

For strview2rate(), the lengths of questions and choices must match. For example, when questions = c('1. To what existence you can feel pleasant if you were in this environment', '2. To what existence you can feel safe if you were in this environment'), choices could be list(c('Unpleasant', 'Less pleasant', 'More pleasant', 'Pleasant'), c('Unsafe', 'Less safe', 'Safer', 'Safe')) For strview2pwc, the choices are always c('left', 'right') for the coparison purposes.

Value

character if argument 'file' is not specified

character if argument 'file' is not specified

Examples

```
data('scdataframe')
header <- "Please review the following picture(s):"
questions <- c('1. To what extent you feel pleasant if you were in this environment',
               '2. To what extent you feel safe if you were in this environment')
choices <- list(c('Unpleasant','Less pleasant', 'Pleasant', 'More pleasant'),
                c('Unsafe', 'Less safe','Safe', 'Safer'))
txt <- streetscape::strview2rate(scdataframe, header, questions, choices)

data('scdataframe')
header <- "Please review the following picture(s):"
questions <- 'which one is more beautiful?'
txt <- streetscape::strview2pwc(scdataframe, k=1, header, questions)
```

---

strview_search *strview_search*

Description

strview_searchByGeo: Search for and download the meta information of street view images via Mapillary API (See detials) based on coordinates of a spatial point with a given distance or a bounding box.

strview_search_nnb: Search for the nearest (within 10m buffer) available street view images and download meta information via Mapillary API (See detials) given coordinates of a spatial point.

strview_search_osm: Search for street view images by sampling locations along the OSM road lines and download meta information via Mapillary API (See detials) given a bounding box.

strview_search_multi: Search for and download the meta information of street view images via Mapillary API (See detials) based on multiple coordinates

Usage

```
strview_searchByGeo(
  x,
  y,
  r,
  epsg,
  bbox,
  token = "",
  limit = 10,
  fields = c(),
  ...
)

strview_search_nnb(x, y, epsg, token = "", fields = c(), ...)

strview_search_osm(bbox, epsg, token, fields = c(), size, ...)

strview_search_multi(viewpoints, epsg, token, fields = c(), ...)
```

Arguments

x	numeric, indicating Longtitude degree of the center point.
y	numeric, indicating latitude degree of the center point.
r	numeric, indicating search distance (meter or feet) for LiDAR data.
epsg	numeric, the EPSG code specifying the coordinate reference system.
bbox	vector, a bounding box defining the geographical area for downloading data.
token	character, API token of Mapillary.
limit	numeric, indicating the number of returns. The maximum is 2000.
fields	vector, a vector of fields indicates the information of images to be retrieved (See details). 'is_pano', 'thumb_256_url', 'height', 'width', 'computed_geometry', 'computed_altitude', and 'detections' are retrieved as a default setting.
...	indicating filters (see details)
size	numeric, (approximate) number of locations sampled on OSM spatial lines (this is for strview_search_osm only).
viewpoints	sf or matrix, indicating multiple degress-based coordinates for searching available street views (this is for strview_search_multi only).

Details

strview_search

To request an API token of Mapillary, please create your access token at https://mapillary.com/developer. For 'fields', one can review all available fields in this package by calling streetscape::field_list().

Value

For strview_searchByGeo(), a StreetscapeDataFrame returned combining a dataframe of the image information.

For strview_search_nnb(), a StreetscapeDataFrame with one-row dataframe will be returned if there is any available images near to the given point

For strview_search_osm(), a StreetscapeDataFrame that combines the information of street views from all sampled points along the OSM lines within the specified bounding box.

For strview_search_multi(), a StreetscapeDataFrame that combines the information of street views based on the coordinates of multiple spatial points

Note

If there is no street view images within the search area, the function only returns an integer 0.

See Also

available_field() available_filter() see_streetscape_class()

Examples

```
bbox <- c(-83.751812,42.272984,-83.741255,42.279716)
if (isTRUE(file.exists("streetscape_token.sysdata"))) {
 data <- streetscape::strview_searchByGeo(bbox = bbox,
                                          epsg = 2253,
                                          token = "",
                                          is_pano = TRUE)
 data <- streetscape::strview_searchByGeo(x = -83.741289,
                                          y = 42.270146,
                                          r = 100,
                                          epsg = 2253,
                                          token = "",
                                          is_pano = TRUE)
}



if (isTRUE(file.exists("streetscape_token.sysdata"))) {
 data <- streetscape::strview_search_nnb(
         x = -83.743460634278,
         y = 42.277848830294,
         epsg = 2253,
         token = '')
}


bbox <- c(-83.752041,42.274896,-83.740711,42.281945)
if (isTRUE(file.exists("streetscape_token.sysdata"))) {
 data <- streetscape::strview_search_osm(
         bbox = bbox,
         epsg = 2253,
```

```
          token = '',
          size = 100)
}


x <- c(-83.752041, -83.740711)
y <- c(42.274896, 42.281945)
viewpoints <- cbind(x, y)
if (isTRUE(file.exists("streetscape_token.sysdata"))) {
 data <- streetscape::strview_search_multi(
          viewpoints = viewpoints,
          epsg = 2253,
          token = '')
}
```

Index