Package 'happign'

April 26, 2022

Title R Interface to 'IGN' Web Services

Version 0.1.4

Maintainer Paul Carteron <carteronpaul@gmail.com>

Description Interface to easily access the National Institute of Geographic and Forestry Information open-source data for any area of interest in France via WFS (shapefile) and WMS (raster) web services <https://geoservices.ign.fr/services-web-experts>.

License GPL (>= 3)

URL https://github.com/paul-carteron, https://paul-carteron.github.io/happign/

BugReports https://github.com/paul-carteron/happign/issues

Imports checkmate, curl, dplyr, httr, httr2, magrittr, sf, stars, tidyr, xml2

Suggests covr, DT, knitr, rmarkdown, testthat (>= 3.0.0), tmap, vcr (>= 0.6.0), webmockr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.1.2

Depends R (>= 2.10)

Author Paul Carteron [aut, cre] (<https://orcid.org/0000-0002-6942-6662>)

Repository CRAN

Date/Publication 2022-04-25 22:10:03 UTC

R topics documented:

`code_insee` *code INSEE*

Description

A dataset containing insee code of commune

Usage

```
code_insee
```

Format

A vector

Source

https://www.insee.fr/fr/information/2115000

`get_apicarto_cadastre` *Apicarto Cadastre*

Description

Implementation of the cadastre module of the IGN's apicarto

Usage

```
## S3 method for class 'sf'
get_apicarto_cadastre(
  x,
  section = NULL,
  numero = NULL,
  code_abs = NULL,
  source_ign = "PCI"
)

## S3 method for class 'sfc'
get_apicarto_cadastre(
  x,
  section = NULL,
  numero = NULL,
  code_abs = NULL,
  source_ign = "PCI"
)

## S3 method for class 'character'
get_apicarto_cadastre(
  x,
  section = NULL,
  numero = NULL,
  code_abs = NULL,
  source_ign = "PCI"
)
```

Arguments

x	It can be a shape or multiple insee code : • Shape : all the cadastral parcels contained in it are downloaded. It should be an object of class `sf` or `sfc`. • Code insee : filter the response on the INSEE code entered (must be a `character` or a vector of `character`)
section	A `character` or a vector of `character` to filter the response on the cadastral section code entered (on 2 characters)
numero	A `character` or a vector of `character` to filter the answers on the entered parcel number (on 4 characters)
code_abs	A `character` or a vector of `character` to filter the answers on the code of absorbed commune. This prefix is useful to differentiate between communes that have merged
source_ign	Can be "BDP" for BD Parcellaire or "PCI" for Parcellaire express. The BD Parcellaire is a discontinued product. Its use is no longer recommended because it is no longer updated. The use of PCI Express is strongly recommended and will become mandatory. More information on the comparison of this two products can be found here

Details

#' @usage get_apicarto_cadastre(x, section = NULL, numero = NULL, code_abs = NULL, source_ign = "PCI")

Value

`get_apicarto_cadastre`return an object of class `sf`

Examples

```
## Not run:
library(sf)
library(tmap)

#' # shape from the best town in France
shape <- st_polygon(list(matrix(c(-4.373937, 47.79859,
                                  -4.375615, 47.79738,
                                  -4.375147, 47.79683,
                                  -4.373898, 47.79790,
                                  -4.373937, 47.79859),
                                  ncol = 2, byrow = TRUE)))
shape <- st_sfc(shape, crs = st_crs(4326))

PCI_shape <- get_apicarto_cadastre(shape, section = c("AX", "BR"))
BDP_Code <- get_apicarto_cadastre("29158", section = c("AX", "BR"), source_ign = "BDP")

tm_shape(PCI_shape)+
   tm_borders()+
tm_shape(shape)+
   tm_borders(col = "red")

tm_shape(BDP_Code)+
   tm_polygons(col = "section", border.col = "black")


## End(Not run)
```

---

get_apicarto_plu *Apicarto module Geoportail de l'urbanisme*

Description

Apicarto module Geoportail de l'urbanisme

Usage

```
get_apicarto_plu (x,
                  ressource = "zone-urba",
                  partition = NULL)
```

Arguments

x	An object of clas sf or sfc. If NULL, partition must be filled by partition of PLU.
ressource	A character from this list : "document", "zone-urba", "secteur-cc", "prescription-surf", "prescription-lin", "prescription-pct", "info-surf", "info-lin", "info-pct". See detail for more info.
partition	A character corresponding to PLU partition (can be retrieve using get_apicarto_plu(x,"document",pa = NULL)). If partition is explicitely set, all PLU features are returned and geom is override

Details

- "document' :
- "zone-urba" :
- "secteur-cc" :
- "prescription-surf" :
- "prescription-lin" :
- "prescription-pct" :
- "info-surf" :
- "info-lin" :
- "info-pct" :

Value

A object of class sf

Examples

```
## Not run:
library(tmap)
library(sf)
point <- st_sfc(st_point(c(-0.4950188466302029, 45.428039987269926)), crs = 4326)

# If you know the partition (all PLU features are returned, geom is override)
partition <- "DU_17345"
poly <- get_apicarto_plu(x = NULL, ressource = "zone-urba", partition = partition)
qtm(poly)+qtm(point, symbols.col = "red", symbols.size = 2)

# If you don't know partition (only intersection between geom and PLU features is returned)
poly <- get_apicarto_plu(x = point, ressource = "zone-urba", partition = NULL)
qtm(poly)+qtm(point, symbols.col = "red", symbols.size = 2)

# If you wanna find partition
document <- get_apicarto_plu(point, ressource = "document", partition = NULL)
partition <- unique(document$partition)

# Get all prescription : /!\ prescription is different than zone-urba
```

```
partition <- "DU_17345"
ressources <- c("prescription-surf", "prescription-lin", "prescription-pct")

# I recommend to use purrr package for loop
library(purrr)
all_prescription <- map(.x = ressources,
                        .f = ~ get_apicarto_plu(point, .x, partition))

## End(Not run)
```

`get_apikeys` *List of all API keys from IGN*

Description

All API keys are manually extract from this table provided by IGN.

Usage

```
get_apikeys()
```

Value

character

Examples

```
## Not run:
# One API key
get_apikeys()[1]

# All API keys
get_apikeys()


## End(Not run)
```

get_iso *Create isochron or isodistance from IGN routing API*

Description

Create isochron or isodistance from IGN routing API

Usage

```
get_iso(point,
                time = NULL,
                distance = NULL,
                transport = "Voiture",
                reverse = FALSE,
                smoothing = TRUE,
                holes = TRUE,
                crs = "EPSG:4326")
```

Arguments

point	Object of class sf or sfc. Needs to be located in France.
time	Duration in minute to create isochron. If time and distance parameters are set, only time will be used.
distance	Distance in meter to create isodistance. If time and distance parameters are set, only time will be used.
transport	There is two type of transport : "Voiture" (car) or " Pieton" (pedestrian).
reverse	A boolean corresponding to the sens of travel. If FALSE, point corresponding to the start.
smoothing	A boolean determining whether the returned shape should be smoothed or not
holes	Some areas are inaccessible. If TRUE they are highlighted
crs	System of coordinate

Details

Native API parameter can be find at IGN documentation.

Value

get_iso() returned a polygon corresponding to all the points reachable in a defined time or distance

Examples

```
## Not run:
library(sf)
library(tmap); tmap_mode("view")

point <- st_sfc(st_point(c(-4.340329361590764, 47.81236909325107)),
                crs = 4326)
isodistance <- get_iso(point, distance = 1000, transport = "Pieton")

qtm(isodistance)+
qtm(point, dots.col = "red")


## End(Not run)
```

`get_layers_metadata` *Metadata for one couple of apikey and data_type*

Description

Metadata are retrieved using the IGN APIs. The execution time can be long depending on the size of the metadata associated with the API key and the overload of the IGN servers.

Usage

```
get_layers_metadata(apikey, data_type)

## S3 method for class 'character'
get_layers_metadata(apikey, data_type)

## S3 method for class 'wfs'
get_layers_metadata(apikey, data_type)

## S3 method for class 'wms'
get_layers_metadata(apikey, data_type)
```

Arguments

`apikey`	API key from `get_apikeys()` or directly from the IGN website
`data_type`	Should be `"wfs"` or `"wms"`. See details for more information about these two Webservice formats.

Details

- WFS is a standard protocol defined by the OGC (Open Geospatial Consortium) and recognized by an ISO standard. The reference document is available on the OGC website. The Geoportail WFS service implements version 2.0 of this protocol. The WFS service of Geoportail gives access to objects from different IGN databases: BD TOPO®, BD CARTO®, BD ADRESSE®, BD FORET® or PARCELLAIRE EXPRESS (PCI).

- WMS is a standard protocol defined by the OGC (Open Geospatial Consortium) and recognized by an ISO standard. The reference document is available on the OGC website.
- For further more detail, check IGN documentation page

Value

data.frame with name of layer, abstract and crs

See Also

get_apikeys()

Examples

```
## Not run:

apikey <- get_apikeys()[4]
metadata_table <- get_layers_metadata(apikey, "wms")
all_layer_name <- metadata_table$name
abstract_of_MNT <- metadata_table[1,3]
crs_of_MNT <- unlist(metadata_table[1,4])

# list with every wfs metadata
list_metadata = lapply(X = get_apikeys(),
                       FUN = get_layers_metadata,
                       data_type = "wfs")

# Convert list to one single data.frame
all_metadata = dplyr::bind_rows(list_metadata)

## End(Not run)
```

---

get_wfs *Download WFS layer*

Description

Directly download a shapefile layer from the French National Institute of Geographic and Forestry. To do that, it need a location giving by a shapefile, an apikey and the name of layer. You can find those information from IGN website

Usage

```
get_wfs(shape,
        apikey,
        layer_name,
        filename = NULL)
```

Arguments

shape	Object of class sf. Needs to be located in France.
apikey	API key from get_apikeys() or directly from IGN website
layer_name	Name of the layer from get_layers_metadata(apikey,"wfs") or directly from IGN website
filename	File name to save shape on disk as ".shp".

Value

get_wfsreturn an object of class sf

See Also

get_apikeys(), get_layers_metadata()

Examples

```
## Not run:
library(sf)
library(tmap)

# Get the borders of best town in France --------------------

apikey <- get_apikeys()[1]
metadata_table <- get_layers_metadata(apikey, "wfs")
layer_name <- metadata_table[32,2]

# One point from the best town in France
shape <- st_point(c(-4.373937, 47.79859))
shape <- st_sfc(shape, crs = st_crs(4326))

# Download borders
borders <- get_wfs(shape, apikey, layer_name)

# Verif
tmap_mode("view") # easy interactive map
qtm(borders, fill = NULL, borders = "firebrick") # easy map

# Get forest_area of the best town in France ----------------
forest_area <- get_wfs(shape = borders,
                       apikey = get_apikeys()[9],
                       layer_name = get_layers_metadata(get_apikeys()[9], "wfs")[2,2])

# Verif
qtm(forest_area, fill = "essence")

# Get roads of the best town in France ----------------------
roads <- get_wfs(shape = borders,
                 apikey = "cartovecto",
                 layer_name = "BDCARTO_BDD_WLD_WGS84G:troncon_route")
```

```
# Verif
qtm(roads)

## End(Not run)
```

`get_wms_raster` *Download WMS raster layer*

Description

Directly download a raster layer from the French National Institute of Geographic and Forestry. To do that, it need a location giving by a shapefile, an apikey and the name of layer. You can find those information from IGN website

Usage

```
get_wms_raster(shape,
               apikey = "altimetrie",
               layer_name = "ELEVATION.ELEVATIONGRIDCOVERAGE",
               resolution = 10,
               filename = NULL,
               version = "1.3.0",
               format = "image/geotiff",
               styles = "",
               method = "auto",
               mode = "wb")
```

Arguments

`shape`	Object of class `sf`. Needs to be located in France.
`apikey`	API key from `get_apikeys()` or directly from IGN website
`layer_name`	Name of the layer from `get_layers_metadata(apikey,"wms")` or directly from IGN website
`resolution`	Cell size in meter. WMS are limited to 2048x2048 pixels so depending of the shape and the resolution, correct number and size of tiles is calculated. See detail for more information about resolution.
`filename`	Name of raster download to disk. The resolution is automatically added to the filename. If raster with same name already exist in the current directory, it is directly imported into R
`version`	The version of the service used. Set to latest version by default. See detail for more information about `version`.
`format`	The output format - type-mime - of the image file. Set to geotiff by default. See detail for more information about `format`.
`styles`	The rendering style of the layers. Set to "" by default. See detail for more information about `styles`.

method	Method to be used for downloading files. See download.file() for more detail.
mode	The mode with which to write the file. See download.file() for more detail.

Details

- Setting the resolution parameter higher than the base resolution of the layer multiplies the number of pixels without increasing the precision. For example, the download of the BD Alti layer from IGN will be optimal for a resolution of 25m.
- version, format and styles parameters are detailed on IGN documentation

Value

get_wms_raster return an object of class stars. Depending on the layer, this can be a simple raster (2 dimensions and 1 attribute) or an RGB raster (3 dimensions and 1 attribute).

See Also

get_apikeys(), get_layers_metadata(), download.file()

Examples

```
## Not run:
library(sf)
library(stars)
library(tmap)

apikey <- get_apikeys()[4]

metadata_table <- get_layers_metadata(apikey, "wms")
layer_name <- metadata_table[2,2]

# shape from the best town in France
shape <- st_polygon(list(matrix(c(-4.373937, 47.79859,
                                  -4.375615, 47.79738,
                                  -4.375147, 47.79683,
                                  -4.373898, 47.79790,
                                  -4.373937, 47.79859),
                                  ncol = 2, byrow = TRUE)))
shape <- st_sfc(shape, crs = st_crs(4326))

# Downloading digital elevation model from IGN
mnt <- get_wms_raster(shape, apikey, layer_name, resolution = 10, filename = "raster_name")
mnt[mnt < 0] <- NA # remove negative values in case of singularity
names(mnt) <- "Elevation [m]" # Rename raster ie the title legend

# Verif
qtm(mnt)+
qtm(shape, fill = NULL, borders.lwd = 3)

## End(Not run)
```

shp_to_geojson *Convert sf or sfc to geojson format*

Description

Convert sf or sfc to geojson format

Usage

```
shp_to_geojson(shape)
```

Arguments

shape	A shape of class sf or sfc. Could be a POLYGON, POINT or LINESTRING.

Value

Return a geojson string

Index