Package 'arcgeocoder'

March 21, 2024

Title Geocoding with the 'ArcGIS' REST API Service

Version 0.2.0

Description Lite interface for finding locations of addresses or businesses around the world using the 'ArcGIS' REST API service <https://developers.arcgis.com/rest/geocode/api-reference/overview-world-geocoding-service.htm>. Address text can be converted to location candidates and a location can be converted into an address. No API key required.

License MIT + file LICENSE

URL https://dieghernan.github.io/arcgeocoder/, https://github.com/dieghernan/arcgeocoder

BugReports https://github.com/dieghernan/arcgeocoder/issues

Depends R (>= 3.6.0)

Imports dplyr (>= 1.0.0), jsonlite (>= 1.7.0)

Suggests ggplot2, knitr, rmarkdown, sf, testthat (>= 3.0.0), tibble, tidygeocoder

VignetteBuilder knitr

Config/Needs/website dieghernan/gitdevr, lwgeom, leaflet, terra, tidyterra, maptiles, styler, mapSpain, remotes, reactable, crosstalk

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

X-schema.org-applicationCategory cartography

X-schema.org-keywords r, geocoding, arcgis, address, reverse-geocoding, rstats, r-package, api-wrapper

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph]
(<https://orcid.org/0000-0001-8457-4658>)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2024-03-21 13:10:02 UTC

R topics documented:

`arc_categories` *ArcGIS REST API category data base*

Description

Database of available categories that can be used for filtering results provided by `arc_geo()`, `arc_geo_multi()` and `arc_geo_categories()` in `tibble` format.

Format

A `tibble` with 376 rows and fields:

level_1 Top-level category

level_2 Second-level category

level_3 Child-level category

Details

See ArcGIS REST Category filtering for details and examples.

The geocoding service allows users to search for and geocode many types of addresses and places around the world. This simplifies the application building process, as developers don't need to know what types of places their users are searching for, because the service can decipher that. However, due to this flexibility, it is possible for ambiguous searches to match to many different places, and users may sometimes receive unexpected results. For example, a search for a city may match to a street name, or a search for an airport code may match to a country abbreviation.

For such cases, the service provides the ability to filter out unwanted geocode results with the `category` parameter. The `category` parameter limits the types of places for which the service searches, thus eliminating false positive matches and potentially speeding up the search process.

The results shows a list of categories with three different hierarchy levels (`level_1`, `level_2`, `level_3`). If a `level_1` category is requested (i.e. `POI`) the child categories may be included also in the results.

Note

Data extracted on **10 January 2023**.

Source

ArcGIS REST Category filtering.

See Also

`arc_geo_categories()`, `arc_geo()`, `arc_geo_multi()`

Other datasets: `arc_spatial_references`

Examples

```
# Get all possible values
data("arc_categories")
arc_categories

# Using categories

sea_1 <- arc_geo("sea",
  custom_query = list(outFields = c("LongLabel", "Type")),
  limit = 2
)


dplyr::glimpse(sea_1)

# An airport, but if we use categories...

sea_2 <- arc_geo("sea",
  custom_query = list(outFields = c("LongLabel", "Type")),
  limit = 2, category = "Food"
)

dplyr::glimpse(sea_2)

# We can use a list of categories
sea_3 <- arc_geo("sea",
  custom_query = list(outFields = c("LongLabel", "Type")),
  sourcecountry = "UK", limit = 5,
  category = c("Amusement Park", "Aquarium")
)

dplyr::glimpse(sea_3)
```

arc_geo *Geocoding using the ArcGIS REST API*

Description

Geocodes addresses given as character values. This function returns the `tibble` object associated with the query.

This function uses the `SingleLine` approach detailed in the ArcGIS REST docs. For multi-field queries (i.e. using specific address parameters) use `arc_geo_multi()` function.

Usage

```
arc_geo(
  address,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  progressbar = TRUE,
  outsr = NULL,
  langcode = NULL,
  sourcecountry = NULL,
  category = NULL,
  custom_query = list()
)
```

Arguments

`address`	character with single line address (`"1600 Pennsylvania Ave NW, Washington"`) or a vector of addresses (`c("Madrid", "Barcelona")`).
`lat`	latitude column name in the output data (default `"lat"`).
`long`	longitude column name in the output data (default `"lon"`).
`limit`	maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
`full_results`	returns all available data from the API service. This is a shorthand of `outFields=*`. See **References**. If `FALSE` (default) only the default values of the API would be returned. See also `return_addresses` argument.
`return_addresses`	return input addresses with results if `TRUE`.
`verbose`	if `TRUE` then detailed logs are output to the console.

progressbar	Logical. If TRUE displays a progress bar to indicate the progress of the function.
outsr	The spatial reference of the x,y coordinates returned by a geocode request. By default is NULL (i.e. the parameter won't be used in the query). See **Details** and arc_spatial_references.
langcode	Sets the language in which reverse-geocoded addresses are returned.
sourcecountry	Limits the candidates returned to the specified country or countries. Acceptable values include the three-character country code. You can specify multiple country codes to limit results to more than one country.
category	A place or address type that can be used to filter results. Several values can be used as well as a vector (i.e. c("Cinema", "Museum")). See arc_categories for details.
custom_query	API-specific parameters to be used, passed as a named list.

Details

More info and valid values in the ArcGIS REST docs.

Value

A tibble object with the results. See the details of the output in ArcGIS REST API Service output.

outsr

The spatial reference can be specified as either a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, i.e. WKID = 4326)).

See arc_spatial_references for values and examples.

References

ArcGIS REST findAddressCandidates.

See Also

tidygeocoder::geo()

Other functions for geocoding: arc_geo_categories(), arc_geo_multi(), arc_reverse_geo()

Examples

```
arc_geo("Madrid, Spain")

library(dplyr)

# Several addresses with additional output fields
with_params <- arc_geo(c("Madrid", "Barcelona"),
  custom_query = list(outFields = c("LongLabel", "CntryName"))
)
```

```
with_params %>%
  select(lat, lon, CntryName, LongLabel)

# With options: restrict search to USA
with_params_usa <- arc_geo(c("Madrid", "Barcelona"),
  sourcecountry = "USA",
  custom_query = list(outFields = c("LongLabel", "CntryName"))
)

with_params_usa %>%
  select(lat, lon, CntryName, LongLabel)
```

`arc_geo_categories` *Geocode places on a given area by category*

Description

This function is useful for extracting places with a given category (or list of categories) near or within a given location or area. This is a wrapper of `arc_geo()`, but it is vectorized over `category`.

See arc_categories for a detailed explanation and available values.

Note that for obtaining results it is needed:

- Either to provide a pair of coordinates (`x`,`y` parameters) that would be used as a reference for geocoding.
- Or a viewbox (aka bounding box) on the `bbox` parameter defining an desired extent of the results.

It is possible to combine the two approaches (i.e. providing `x`,`y`,`bbox` values) in order to boost the geocoding process. See **Examples**.

Usage

```
arc_geo_categories(
  category,
  x = NULL,
  y = NULL,
  bbox = NULL,
  name = NULL,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  verbose = FALSE,
  custom_query = list(),
  ...
)
```

Arguments

`category`	A place or address type that can be used to filter results. Several values can be used as well as a vector (i.e. `c("Cinema", "Museum")`), performing one call for each value. See **Details**.
`x`	longitude values in numeric format. Must be in the range $[-180, 180]$.
`y`	latitude values in numeric format. Must be in the range $[-90, 90]$.
`bbox`	A numeric vector of latitude and longitude `c(minX, minY, maxX, maxY)` that restrict the search area. See **Details**.
`name`	Optionally, a string indicating the name or address of the desired results.
`lat`	latitude column name in the output data (default `"lat"`).
`long`	longitude column name in the output data (default `"lon"`).
`limit`	maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
`full_results`	returns all available data from the API service. This is a shorthand of `outFields=*`. See **References**. If `FALSE` (default) only the default values of the API would be returned. See also `return_addresses` argument.
`verbose`	if `TRUE` then detailed logs are output to the console.
`custom_query`	API-specific parameters to be used, passed as a named list.
`...`	Arguments passed on to `arc_geo` `sourcecountry` Limits the candidates returned to the specified country or countries. Acceptable values include the three-character country code. You can specify multiple country codes to limit results to more than one country. `outsr` The spatial reference of the `x,y` coordinates returned by a geocode request. By default is `NULL` (i.e. the parameter won't be used in the query). See **Details** and arc_spatial_references. `langcode` Sets the language in which reverse-geocoded addresses are returned.

Details

Bounding boxes can be located using different online tools, as Bounding Box Tool.

For a full list of valid categories see arc_categories.

This function is vectorized over `category`, that means that it would perform one independent call to `arc_geo()` for each `category` value.

`arc_geo_categories()` also understands a single string of categories separated by commas (`"Cinema,Museum"`), that would be internally treated as `c("Cinema", "Museum")`.

Value

A `tibble` object with the results. See the details of the output in ArcGIS REST API Service output.

`outsr`

The spatial reference can be specified as either a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, i.e. WKID = 4326)).

See arc_spatial_references for values and examples.

See Also

ArcGIS REST Category filtering.

arc_categories

Other functions for geocoding: arc_geo(), arc_geo_multi(), arc_reverse_geo()

Examples

```
# Full workflow: Gas Stations near Carabanchel, Madrid

# Get Carabanchel
carab <- arc_geo("Carabanchel, Madrid, Spain")

# CRS
carab_crs <- unique(carab$latestWkid)


library(ggplot2)

base_map <- ggplot(carab) +
  geom_point(aes(lon, lat), size = 5, color = "red") +
  geom_rect(aes(xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax),
    fill = NA,
    color = "blue"
  ) +
  coord_sf(crs = carab_crs)


# Ex1: Search near Carabanchel (not restricted)
ex1 <- arc_geo_categories("Gas Station",
  # Location
  x = carab$lon, y = carab$lat,
  limit = 50, full_results = TRUE
)


# Reduce number of labels to most common ones
library(dplyr)

labs <- ex1 %>%
  count(ShortLabel) %>%
  slice_max(n = 7, order_by = n) %>%
  pull(ShortLabel)

base_map +
  geom_point(data = ex1, aes(lon, lat, color = ShortLabel)) +
  scale_color_discrete(breaks = labs) +
  labs(
    title = "Example 1",
    subtitle = "Search near (points may be far away)"
```

```
  )

# Example 2: Include part of the name, different results
ex2 <- arc_geo_categories("Gas Station",
  # Name
  name = "Repsol",
  # Location
  x = carab$lon, y = carab$lat,
  limit = 50, full_results = TRUE
)

base_map +
  geom_point(data = ex2, aes(lon, lat, color = ShortLabel)) +
  labs(
    title = "Example 2",
    subtitle = "Search near with name"
  )

# Example 3: Near within a extent
ex3 <- arc_geo_categories("Gas Station",
  name = "Repsol",
  bbox = c(carab$xmin, carab$ymin, carab$xmax, carab$ymax),
  limit = 50, full_results = TRUE
)

base_map +
  geom_point(data = ex3, aes(lon, lat, color = ShortLabel)) +
  labs(
    title = "Example 3",
    subtitle = "Search near with name and bbox"
  )
```

`arc_geo_multi` *Geocoding using the ArcGIS REST API with multi-field query*

Description

Geocodes addresses given specific address components.This function returns the `tibble` associated with the query.

For geocoding using a single text string use `arc_geo()` function.

Usage

```
arc_geo_multi(
  address = NULL,
  address2 = NULL,
  address3 = NULL,
```

```
  neighborhood = NULL,
  city = NULL,
  subregion = NULL,
  region = NULL,
  postal = NULL,
  postalext = NULL,
  countrycode = NULL,
  lat = "lat",
  long = "lon",
  limit = 1,
  full_results = FALSE,
  return_addresses = TRUE,
  verbose = FALSE,
  progressbar = TRUE,
  outsr = NULL,
  langcode = NULL,
  category = NULL,
  custom_query = list()
)
```

Arguments

`address`, `address2`, `address3`, `neighborhood`, `city`, `subregion`	Address components (See **Details**).
`region`, `postal`, `postalext`, `countrycode`	More address components, see (See **Details**).
`lat`	latitude column name in the output data (default `"lat"`).
`long`	longitude column name in the output data (default `"lon"`).
`limit`	maximum number of results to return per input address. Note that each query returns a maximum of 50 results.
`full_results`	returns all available data from the API service. This is a shorthand of `outFields=*`. See **References**. If `FALSE` (default) only the default values of the API would be returned. See also `return_addresses` argument.
`return_addresses`	return input addresses with results if `TRUE`.
`verbose`	if `TRUE` then detailed logs are output to the console.
`progressbar`	Logical. If `TRUE` displays a progress bar to indicate the progress of the function.
`outsr`	The spatial reference of the `x,y` coordinates returned by a geocode request. By default is `NULL` (i.e. the parameter won't be used in the query). See **Details** and arc_spatial_references.
`langcode`	Sets the language in which reverse-geocoded addresses are returned.
`category`	A place or address type that can be used to filter results. Several values can be used as well as a vector (i.e. `c("Cinema", "Museum")`). See arc_categories for details.
`custom_query`	API-specific parameters to be used, passed as a named list.

Details

More info and valid values in the ArcGIS REST docs.

Value

A `tibble` object with the results. See the details of the output in ArcGIS REST API Service output.

The resulting output would include also the input parameters (columns with prefix `q_`) for better tracking the results.

Address components

This function allows to perform structured queries by different components of an address. At least one field should be different than `NA` or `NULL`.

A vector of values can be provided for each parameter for multiple geocoding. When using vectors on different parameters, their lengths should be the same.

The following list provides a brief description of each parameter:

- `address`: A string that represents the first line of a street address. In most cases it will be the **street name and house number** input, but it can also be used to input building name or place-name.
- `address2`: A string that represents the second line of a street address. This can include **street name/house number, building name, place-name, or sub unit**.
- `address3`: A string that represents the third line of a street address. This can include **street name/house number, building name, place-name, or sub unit**.
- `neighborhood`: The smallest administrative division associated with an address, typically, a **neighborhood** or a section of a larger populated place.
- `city`: The next largest administrative division associated with an address, typically, a **city or municipality**.
- `subregion`: The next largest administrative division associated with an address. Depending on the country, a sub region can represent a **county, state, or province**.
- `region`: The largest administrative division associated with an address, typically, a **state or province**.
- `postal`: The **standard postal code** for an address, typically, a three– to six-digit alphanumeric code.
- `postalext`: A **postal code extension**, such as the United States Postal Service ZIP+4 code.
- `countrycode`: A value representing the **country**. Providing this value **increases geocoding speed**. Acceptable values include the full country name in English or the official language of the country, the two-character country code, or the three-character country code.

`outsr`

The spatial reference can be specified as either a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, i.e. WKID = 4326)).

See arc_spatial_references for values and examples.

References

ArcGIS REST findAddressCandidates

See Also

tidygeocoder::geo()

Other functions for geocoding: arc_geo(), arc_geo_categories(), arc_reverse_geo()

Examples

```
simple <- arc_geo_multi(
  address = "Plaza Mayor", limit = 10,
  custom_query = list(outFields = c("LongLabel", "CntryName", "Region"))
)

library(dplyr)

simple %>%
  select(lat, lon, CntryName, Region, LongLabel) %>%
  slice_head(n = 10)

# Restrict search to Spain
simple2 <- arc_geo_multi(
  address = "Plaza Mayor", countrycode = "ESP",
  limit = 10,
  custom_query = list(outFields = c("LongLabel", "CntryName", "Region"))
)

simple2 %>%
  select(lat, lon, CntryName, Region, LongLabel) %>%
  slice_head(n = 10)

# Restrict to a region
simple3 <- arc_geo_multi(
  address = "Plaza Mayor", region = "Segovia",
  countrycode = "ESP",
  limit = 10,
  custom_query = list(outFields = c("LongLabel", "CntryName", "Region"))
)

simple3 %>%
  select(lat, lon, CntryName, Region, LongLabel) %>%
  slice_head(n = 10)
```

arc_reverse_geo *Reverse Geocoding using the ArcGIS REST API*

Description

Generates an address from a latitude and longitude. Latitudes must be in the range $[-90, 90]$ and longitudes in the range $[-180, 180]$. This function returns the `tibble` associated with the query.

Usage

```
arc_reverse_geo(
  x,
  y,
  address = "address",
  full_results = FALSE,
  return_coords = TRUE,
  verbose = FALSE,
  progressbar = TRUE,
  outsr = NULL,
  langcode = NULL,
  featuretypes = NULL,
  locationtype = NULL,
  custom_query = list()
)
```

Arguments

x	longitude values in numeric format. Must be in the range $[-180, 180]$.
y	latitude values in numeric format. Must be in the range $[-90, 90]$.
address	address column name in the output data (default `"address"`).
full_results	returns all available data from the API service. If `FALSE` (default) only latitude, longitude and address columns are returned.
return_coords	return input coordinates with results if `TRUE`.
verbose	if `TRUE` then detailed logs are output to the console.
progressbar	Logical. If `TRUE` displays a progress bar to indicate the progress of the function.
outsr	The spatial reference of the `x,y` coordinates returned by a geocode request. By default is `NULL` (i.e. the parameter won't be used in the query). See **Details** and arc_spatial_references.
langcode	Sets the language in which reverse-geocoded addresses are returned.
featuretypes	This parameter limits the possible match types returned. By default is `NULL` (i.e. the parameter won't be used in the query). See **Details**.
locationtype	Specifies whether the output geometry of `featuretypes = "PointAddress"` or `featuretypes = "Subaddress"` matches should be the rooftop point or street entrance location. Valid values are `NULL` (i.e. not using the parameter in the query), `rooftop` and `street`.
custom_query	API-specific parameters to be used, passed as a named list.

Details

More info and valid values in the ArcGIS REST docs.

Value

A `tibble` with the corresponding results. The `x,y` values returned by the API would be named `lon,lat`. Note that these coordinates correspond to the geocoded feature, and may be different of the `x,y` values provided as inputs.

See the details of the output in ArcGIS REST API Service output.

`outsr`

The spatial reference can be specified as either a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, i.e. WKID = 4326)).

See arc_spatial_references for values and examples.

`featuretypes`

See `vignette("featuretypes", package = "arcgeocoder")` for a detailed explanation of this parameter.

This parameter may be used for filtering the type of feature to be returned when geocoding. Possible values are:

- `"StreetInt"`
- `"DistanceMarker"`
- `"StreetAddress"`
- `"StreetName"`
- `"POI"`
- `"Subaddress"`
- `"PointAddress"`
- `"Postal"`
- `"Locality"`

It is also possible to use several values as a vector (`featuretypes = c("PointAddress", "StreetAddress")`).

References

ArcGIS REST `reverseGeocode`.

See Also

`tidygeocoder::reverse_geo()`

Other functions for geocoding: `arc_geo()`, `arc_geo_categories()`, `arc_geo_multi()`

Examples

```
arc_reverse_geo(x = -73.98586, y = 40.75728)

# Several coordinates
arc_reverse_geo(x = c(-73.98586, -3.188375), y = c(40.75728, 55.95335))

# With options: using some additional parameters
sev <- arc_reverse_geo(
  x = c(-73.98586, -3.188375),
  y = c(40.75728, 55.95335),
  # Restrict to these feautures
  featuretypes = "POI,StreetInt",
  # Result on this WKID
  outsr = 102100,
  verbose = TRUE, full_results = TRUE
)

dplyr::glimpse(sev)
```

---

arc_spatial_references

ESRI (ArcGIS) Spatial Reference data base

Description

Database of available spatial references (CRS) in `tibble` format.

Format

A `tibble` with 9,364 rows and fields:

projtype Projection type (`"ProjectedCoordinateSystems", "GeographicCoordinateSystems","VerticalCoordinat`

wkid Well-Known ID

latestWkid Most recent `wkid`, in case that `wkid` is deprecated

authority `wkid` authority (Esri or EPSG)

deprecated Logical indicating if `wkid` is deprecated

description Human-readable description of the `wkid`

areaname Use area of the `wkid`

wkt Representation of `wkid` in Well-Known Text (WKT). Useful when working with **sf** or **terra**

Details

This data base is useful when using the `outsr` parameter of the functions.

Some projections ids have changed over time, for example Web Mercator is `wkid = 102100` is deprecated and currently is `wkid = 3857`. However, both values would work, and they would return similar results.

Note

Data extracted on **14 January 2023**.

Source

ESRI Projection Engine factory

See Also

`sf::st_crs()`

Other datasets: `arc_categories`

Examples

```
# Get all possible valuesdata("arc_spatial_references")
arc_spatial_references

# Request with deprecated Web Mercator
library(dplyr)
wkid <- arc_spatial_references %>%
  filter(latestWkid == 3857 & deprecated == TRUE) %>%
  slice(1)

glimpse(wkid)

add <- arc_geo("London, United Kingdom", outsr = wkid$wkid)

# Note values lat, lon and wkid. latestwkid give the current valid wkid
add %>%
  select(lat, lon, wkid, latestWkid) %>%
  glimpse()

# See with sf

try(sf::st_crs(wkid$wkid))

# But
try(sf::st_crs(wkid$latestWkid))

# or
try(sf::st_crs(wkid$wkt))
```

Index