Package 'arcgeocoder'

June 3, 2026

Title Geocoding with the 'ArcGIS' REST API Service

Version 0.4.1

Description Lightweight interface for converting addresses into geographic coordinates and coordinates into addresses 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 locations can be converted into addresses. No API key is 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 (>= 4.1.0)

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

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

VignetteBuilder quarto

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

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

X-schema.org-applicationCategory cartography

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

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-06-03 09:30:02 UTC

Contents

`arc_categories` *ArcGIS REST API category data*

Description

Dataset of available categories used to filter results provided by `arc_geo()`, `arc_geo_multi()` and `arc_geo_categories()` in tibble format.

Format

A tibble with 383 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 application development because developers do not need to know what types of places their users are searching for. 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` argument. The `category` argument limits the types of places for which the service searches, thus eliminating false-positive matches and potentially speeding up the search process.

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

Note

Data extracted on **15 January 2026**.

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

# Use categories.

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

dplyr::glimpse(sea_1)

# Categories can disambiguate the result.

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

dplyr::glimpse(sea_2)

# 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 *Geocode addresses with the ArcGIS REST API*

Description

Geocodes addresses supplied as character values and returns the tibble associated with each query.

This function uses the `SingleLine` approach detailed in the ArcGIS REST docs. For multi-field queries (that is, using specific address components), use `arc_geo_multi()`.

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`	Single-line address text (e.g. `"1600 Pennsylvania Ave NW, Washington"`) or a vector of addresses (e.g. `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. Each query has a hard API limit of 50 results.
`full_results`	Logical. If `TRUE`, return all available API fields via `outFields=*`. Default is `FALSE`.
`return_addresses`	Logical. If `TRUE`, keep the input query in the output.

verbose	Logical. If TRUE, output process messages to the console.
progressbar	Logical. If TRUE, show a progress bar for multiple points.
outsr	The spatial reference of the x and y coordinates returned by a geocode request. By default, it is NULL (that is, the argument will not be used in the query). See **Details** and arc_spatial_references.
langcode	Sets the language in which reverse-geocoded addresses are returned.
sourcecountry	Country filter using ISO codes (e.g. "USA"). Multiple values can be supplied as a comma-separated string.
category	Place or address type used as a filter. Multiple values are accepted (e.g. c("Cinema", "Museum")). See arc_categories.
custom_query	Additional API parameters as named list values.

Details

See the ArcGIS REST docs for more information and valid values.

Value

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

outsr

The spatial reference can be specified as a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, that is, 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 the search to the 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 by category in a given area*

Description

This function extracts places with a given category or list of categories near or within a given location or area. It wraps `arc_geo()` and is vectorized over `category`.

See arc_categories for a detailed explanation and available values.

Note: To obtain results, provide either a pair of coordinates (`x` and `y` arguments) used as a reference for geocoding or a bounding box via the `bbox` argument defining a desired extent for results.

You can combine both approaches (providing `x`, `y` and `bbox` values) to improve 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 used to filter results. Several values can also be supplied as a vector (for example, `c("Cinema", "Museum")`), which performs 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 longitude and latitude values `c(minX, minY, maxX, maxY)` that restricts 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 per query. The ArcGIS REST API limits a single request to 50 results.
full_results	Logical. If `TRUE`, return all available API fields via `outFields=*`. Default is `FALSE`.
verbose	Logical. If `TRUE`, output process messages to the console.
custom_query	Additional API parameters as named list values.
...	Arguments passed on to `arc_geo` `sourcecountry` Country filter using ISO codes (e.g. `"USA"`). Multiple values can be supplied as a comma-separated string. `outsr` The spatial reference of the `x` and `y` coordinates returned by a geocode request. By default, it is `NULL` (that is, the argument will not 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 online tools, such as Bounding Box Tool.

For a full list of valid categories, see arc_categories. This function is vectorized over `category`, which means it performs 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"`), which is treated internally as `c("Cinema", "Museum")`.

Value

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

`outsr`

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

# Example 1: Search near Carabanchel (not restricted).
ex1 <- arc_geo_categories("Gas Station",
  # Location.
  x = carab$lon, y = carab$lat,
  limit = 50, full_results = TRUE
)

# Reduce labels to the 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 for 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: Search within a bounding box.
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 bounding box"
  )
```

`arc_geo_multi` *Geocode addresses with a multi-field ArcGIS REST API query*

Description

Geocodes addresses from specific address components and returns the tibble associated with each query.

For geocoding with a single text string, use `arc_geo()`.

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	Additional address components. 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. Each query has a hard API limit of 50 results.
full_results	Logical. If TRUE, return all available API fields via outFields=*. Default is FALSE.
return_addresses	Logical. If TRUE, keep the input query in the output.
verbose	Logical. If TRUE, output process messages to the console.
progressbar	Logical. If TRUE, show a progress bar for multiple points.
outsr	The spatial reference of the x and y coordinates returned by a geocode request. By default, it is NULL (that is, the argument will not be used in the query). See **Details** and arc_spatial_references.
langcode	Sets the language in which reverse-geocoded addresses are returned.
category	Place or address type used as a filter. Multiple values are accepted (e.g. c("Cinema", "Museum")). See arc_categories.
custom_query	Additional API parameters as named list values.

Details

See the ArcGIS REST docs for more information and valid values.

Value

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

The output also includes the input arguments as columns prefixed with q_ to help track the results.

Address components

This function performs structured queries by different components of an address. At least one field should be different from `NA` or `NULL`.

A vector of values can be provided for each argument for multiple geocoding. When using vectors in different arguments, their lengths must be the same.

The following list provides a brief description of each argument:

- `address`: A string that represents the first line of a street address. In most cases, it is the **street name and house number** input, but it can also be used to input a 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 subunit**.
- `address3`: A string that represents the third line of a street address. This can include **street name/house number, building name, place name or subunit**.
- `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 subregion 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 a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, that is, 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 geocode coordinates with the ArcGIS REST API*

Description

Generates an address from a longitude and latitude. Latitudes must be in the range $[-90, 90]$ and longitudes in the range $[-180, 180]$. This function returns the tibble associated with each 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	Output address column name (default "address").
full_results	Logical. If TRUE, return all available API fields. FALSE (default) returns latitude, longitude and address only.
return_coords	Logical. If TRUE, return input coordinates with the results.
verbose	Logical. If TRUE, output process messages to the console.
progressbar	Logical. If TRUE, show a progress bar for multiple points.
outsr	The spatial reference of the x and y coordinates returned by a geocode request. By default, it is NULL (that is, the argument will not be used in the query). See **Details** and arc_spatial_references.
langcode	Sets the language in which reverse-geocoded addresses are returned.
featuretypes	This argument limits the possible match types returned. By default, it is NULL (that is, the argument will not 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 (that is, not using the argument in the query), "rooftop" and "street".
custom_query	API-specific arguments to be used, passed as a named list.

Details

See the ArcGIS REST docs for more information and valid values.

Value

A tibble with the corresponding results. The x and y values returned by the API are named lon and lat. Note that these coordinates correspond to the geocoded feature and may differ from the x and 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 a well-known ID (WKID). If not specified, the spatial reference of the output locations is the same as that of the service (WGS84, that is, WKID = 4326).

See arc_spatial_references for values and examples.

`featuretypes`

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

This argument may be used to filter the type of feature returned when geocoding. Possible values are `"StreetInt"`, `"DistanceMarker"`, `"StreetAddress"`, `"StreetName"`, `"POI"`, `"Subaddress"`, `"PointAddress"`, `"Postal"` and `"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: use additional arguments.
sev <- arc_reverse_geo(
  x = c(-73.98586, -3.188375),
  y = c(40.75728, 55.95335),
  # Restrict to these features.
  featuretypes = "POI,StreetInt",
  # Return results in this WKID.
  outsr = 102100,
  verbose = TRUE, full_results = TRUE
)

dplyr::glimpse(sev)
```

`arc_spatial_references` *Esri (ArcGIS) spatial reference data*

Description

Dataset of available spatial references (CRS) in tibble format.

Format

A tibble with 9,608 rows and fields:

projtype Projection type (`"ProjectedCoordinateSystems"`, `"GeographicCoordinateSystems"` or `"VerticalCoordinateSystems"`).

wkid Well-Known ID.

latestWkid Most recent `wkid`, if `wkid` is deprecated.

authority `wkid` authority (Esri or EPSG).

deprecated Logical indicating whether `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 dataset is useful when using the `outsr` argument.

Some projection IDs have changed over time. For example, Web Mercator `wkid = 102100` is deprecated and is currently `wkid = 3857`. However, both values work and return similar results.

Note

Data extracted on **15 January 2026**.

Source

Esri Projection Engine factory

See Also

`sf::st_crs()`

Other datasets: `arc_categories`

Examples

```
# Get all possible values.
data("arc_spatial_references")
arc_spatial_references

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

glimpse(wkid)

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

# Note values for `lat`, `lon` and `wkid`. `latestWkid` gives the current
# valid `wkid`.
add |>
  select(lat, lon, wkid, latestWkid) |>
  glimpse()

# Try with `sf`.

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

# Try the latest WKID.
try(sf::st_crs(wkid$latestWkid))

# Or use WKT.
try(sf::st_crs(wkid$wkt))
```

Index