Opencagegeo: Stata Module for Forward and Reserve Geocoding

Lars Zeigermann
Düsseldorf Insitute for Competition Economics (DICE)
Germany
zeigermann@dice.hhu.de

Abstract. This article describes `opencagegeo` and its (simplified) immediate version `opencagegeoi`, which allow the user to obtain latitudes and longitudes for addresses (forward geocoding) and retrieve addresses from latitude longitude pairs (reverse geocoding). `opencagegeo` uses OpenCage Data's geocoder which has very flexible terms of use. Contrary to existing geocoders in Stata (using Google Maps, MapQuest or Here Maps), OpenCage Data does not restrict the use of geocodes and explicitly allows data storage.

Keywords: st0001, opencagegeo, opencagegeoi, geocoding, forward geocoding, reverse geocoding, geocoder, OpenCage Data

1 Introduction

Spatial information is extensively used by researchers and practitioners in numerous fields and (forward and reverse) geocoding has become a common exercise. Forward geocoding is the process of converting an address into geographic coordinates. Reverse geocoding uses geographic coordinates to retrieve the postal address.

To facilitate geocoding for Stata users, a number of user-written geocoding routines have been made available. They use so-called APIs (application programming interfaces) to access online geocoding services. These are (the outdated) `geocode` (Ozimek and Miles, 2011), its (no longer available) successor `geocode3` (Bernhard, 2013) and the immediate command `gcode` (Ansari, 2015) using Google Maps' API, `geocodeopen` (Anderson, 2013) using MapQuest's API and `geocodehere` (Hess, 2015) using Here Maps' API. Although these routines work very well, they share a common disadvantage. The terms of use of Google Maps, MapQuest and Here Maps are very restrictive and prohibit data storage which makes them inappropriate for most purposes.

Here, `opencagegeo`'s major advantage comes into play; the flexible terms of use of OpenCage Data's geocoding API it is using. All geocoded data is jointly made available under the ODbL and CC-BY-SA licenses[1] and OpenCage Data does not restrict the use of geocodes.[2]

[1]The ODbL and CC-BY-SA licences are available at `http://opendatacommons.org/licenses/odbl\/summary/` and `https://creativecommons.org/licenses/by-sa/2.0/`.

[2]OpenCage Data's terms of use are available at `http://geocoder.opencagedata.com/faq.html#legal`.

The syntax of `opencagegeo` and its immediate version `opencagegeoi` is described in Section 2 and Section 3. Two short examples are provided in Section 4.

2 Opencagegeo

2.1 Syntax

`opencagegeo` [*if*] [*in*] [, `key(`*string*`)` `number(`*varname*`)` `street(`*varname*`)` `postcode(`*varname*`)` `city(`*varname*`)` `county(`*varname*`)` `state(`*varname*`)` `country(`*varname*`)` `fulladdress(`*varname*`)` `latitude(`*varname*`)` `longitude(`*varname*`)` `coordinates(`*varname*`)` `countrycode(`*varname* `or` *string*`)` `language(`*varname* `or` *string*`)` `replace` `resume`]

2.2 Options

General

`key` (*string*) is required and specifies the OpenCage Data API key.

`countrycode` (*varname* `or` *string*) specifies the country code of the locations. `countrycode()` allows either a string variable or a string as input.

`language` (*varname* `or` *string*) specifies the language in which the results are returned. `language()` allows either a string variable or a string as input. The default is *en* for English.

`resume` specifies that the process is continued after the query limit was exceeded the day before. `resume` may not be combined with `replace`.

Forward Geocoding

`number` (*varname*) specifies the variable containing the house number.

`street` (*varname*) specifies the variable containing the street. *varname* must be string.

`postcode` (*varname*) specifies the variable containing the postal code.

`city` (*varname*) specifies the variable containing the city, town or village. *varname* must be string.

`county` (*varname*) specifies the variable containing the county. *varname* must be string.

`state` (*varname*) specifies the variable containing the state. *varname* must be string.

`country` (*varname*) specifies the variable containing the country. *varname* must be string.

`fulladdress` (*varname*) specifies a single variable containing some or all of the above options. *varname* must be string. `fulladdress()` may not be combined with any of the above options.

Reverse Geocoding

`latitude`(*varname*) specifies the variable containing the latitude. Values must lie between -90 and 90.

`longitude`(*varname*) specifies the variable containing the longitude. Values must lie between -180 and 180.

`coordinates`(*varname*) specifies the variable containing latitude longitude pairs. Latitudes must lie between -90 and 90 and be stated first. Longitudes must lie between -180 and 180. Both values must be separated by a comma. *varname* must be string. `coordinates()` may not be combined with `latitude()` and `longitude()`.

2.3 Remarks

General

`opencagegeo` requires an OpenCage Data API key which may be requested free of charge by signing up at `http://geocoder.opencagedata.com`. The free key allows 2.500 queries per day.[3] If the number of observations to be geocoded exceeds the query limit of 2.500, the `resume` option allows the user to continue geocoding at the beginning of the following day (i.e. 00:00:00 Coordinated Universal Time (UTC)).[4] To continue geocoding, the user has to re-run the same specification of `opencagegeo` but include `resume` until all selected observations are completed. The user is strongly advised not to make any changes to the data until geocoding of all observations is completed.[5]

If the variables produced by `opencagegeo`, see below, already exist, the user can specify `replace` to overwrite existing observations. If either `in` or `if` is specified, only the selected observations will be replaced. `resume` and `replace` may not be combined.

The `countrycode()` option allows the user to specify the country of the locations to be geocoded. Providing a country code will restrict the results to the given country. If all locations lie in the same country, the user may enter the country code directly

[3] For users in need of higher volumes per day, different customer plans are available.

[4] When the query limit is hit, `opencagegeo` displays the remaining time in hh:mm:ss till the limit is reset. For computing the time remaining, `opencagegeo` reuses a code segment from the `elap` command of the `egenmore` package (Cox, 2014).

[5] The `resume` option evaluates the output variable `g_quality` to select observations which have not yet been geocoded.

into `countrycode()`. Alternatively, if the locations are in more than just one country, a variable containing the respective country codes needs to be specified. `opencagegeo` takes two character country codes as defined by the ISO 3166-1 Alpha 2 standard.[6] If any country code is invalid, `opencagegeo` will issue an error message and exit.

The `language()` option allows the user to specify the language in which the results are returned from the OpenCage Geocoder. As with `countrycode()`, either a string or a string variable can be specified. The language must be entered in IETF format language code.[7] The default is English. If the language is set to *native*, the results will be returned in the native language of the location - provided the underlying OpenStreetMap(OSM) data is available in that language. Users of Stata 13 or older are advised to use the `language()` option carefully as many languages other than English contain special characters which cannot be displayed properly.

`opencagegeo` requires two user-written Stata libraries, `insheetjson` and `libjson` (Lindsley, 2012a,b), which are available at Statistical Software Components.

Forward geocoding

Generally, there are two different ways of feeding addresses into `opencagegeo`. Firstly, by components using the `street()`, `number()`, `postcode()`, `city()`, `county()`, `state()` and `country()` options. This is recommended if the location address data is provided in separate variables. Not all options need to be specified at the same time; they can be combined in any meaningful way. To obtain geocodes of, let's say, cities, you may use `city()`, `state()` and `country()`.

Secondly, if the address is contained in a single string variable, the `fulladdress()` option may be employed. The address should follow the country specific conventions, although the OpenCage Geocoder allows for some flexibility. A well-formatted address might take the following formats:

"number street, city postal code, county, state, country"
"street number, postal code city, county, state, country"

Generally, the OpenCage Geocoder is not case sensitive and can deal with commonly used abbreviations. Running `opencagegeo` in Stata 14 (or newer) allows the address variables to be in Unicode (UTF-8). That is address names may contain accented characters, symbols and non-latin characters. For older releases, the input variables must however not contain any special characters.[8] Otherwise, `opencagegeo` will issue an error message and exit. `opencagegeo` is sensitive to spelling mistakes and will return empty strings for misspelled location addresses.

[6] A comprehensive list of all ISO 3166-1 Alpha 2 codes is available at `http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm`.

[7] A comprehensive list of all IETF language codes is available at `http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry`.

[8] Only ASCII printable characters, i.e. character codes 32-127, may be used in the input variables.

Reverse geocoding

As with forward geocoding, two options are available for reverse geocoding. If longitudes and latitudes are contained in two separate variables, the `latitude()` and `longitude()` options should be used.

Alternatively, latitude longitude pairs may be fed into the `coordinates()` option. Latitudes need to be stated first and both values must be separated by a comma.

Latitudes must take values between -90 and 90 and longitudes must be between -180 and 180. Otherwise, `opencagegeo` will issue an error message and exit.

2.4 Output variables

For both forward and reverse geocoding, `opencagegeo` generates a set of 12 variables, all having `g_` prefixes.

`g_latitude` and `g_longitude` contain the latitudes and longitudes retrieved. The variables `g_number`, `g_street`, `g_postcode`, `g_city`, `g_county`, `g_state` and `g_country` contain the respective information returned. If the information is missing (e.g. the house number is not known) or not requested (`number()` was not specified or no information on the house number was contained in address variable fed into the `fulladdress()` option), empty string for the respective variable will be returned. `g_formatted` contains a well-formatted place name generated by the OpenCage Geocoder. In addition to the postal address, it might have information such as the name or the type of the building, institution, shop etc. The JSON paths used to extract data from returned results of the OpenCage Geocoder into the respective variable are given in the appendix.

Finally, `g_confidence` and `g_quality` are generated. The former provides a measure of precision of the match and is directly returned from the OpenCage Geocoder. The confidence value is calculated as the distance in kilometres between the South-East and the North-West corners of the bounding box. Confidence levels are defined as follows:

0 = unable to determine bounding box
1 = 25 km or more
2 = less than 25 km
3 = less than 20 km
4 = less than 15 km
5 = less than 10 km
6 = less than 7.5 km
7 = less than 5 km
8 = less than 1 km
9 = less than 0.5 km
10 = less than 0.25 km

The variable `g_quality` contains the accuracy level of the returned results and is defined as follows:

0 = location not found
1 = country
2 = state
3 = county
4 = city
5 = postcode
6 = street
7 = number

If, for example, the returned results for an address contains information down to the postal code level, but the street and the house number were not found, the quality level will be *postcode*. The highest quality level to be reached is hence determined by the inputs; if the `number()` is not specified or no information on the house number is contained in the variable fed into `fulladdress()`, the highest quality level to be achieved is *street*.

3 Opencagegeoi

3.1 Syntax

```
opencagegeoi #location
```

3.2 Remarks

The immediate `opencagegeoi` takes its inputs from what is typed as arguments rather than from data stored in variables. Instead of creating output variables it immediately displays the results in the output window.

Again, an OpenCage Geocoder API key is required. To save the user the trouble of having to enter the key every time, `opencagegeoi` requires the API key to be stored in a global macro *mykey*. The location to be geocoded is then directly typed after `opencagegeoi`. For forward geocoding, the input should be well-formatted as described above. For reverse geocoding, `opencagegeoi` accepts a latitude longitude pair. The latitude is entered first and both values must be separated by a comma.

The user may also define a global macro *language* containing the IETF code of the language in which the result shall be returned. The default is English. If the user sets the language to *native*, the results are returned in the native language of the location. Due to special characters in many languages, the user is advised to use this option carefully, see above.

3.3 Saved results

In addition to displaying the well-formatted address returned from the OpenCage Geocoder and the latitude and longitude values, `opencagegeoi` saves the following results to `r()`.

Macros

`r(input)`	address as typed by user	`r(formatted)`	well-formatted address returned from OpenCage Geocoder
`r(lat)`	latitude	`r(lon)`	longitude
`r(conf)`	confidence level of result		

4 Examples

4.1 Opencagegeo

Forward Geocoding

As an example consider the addresses of five Goethe Institutes in Europe, Mexico and the United States.[9] The data is displayed below.

```
. list
```

	CITY	POSTCODE	STREET	NUMBER	COUNTRY
1.	London	SW7 2PH	Exhibition Road	50	UK
2.	Mexico City	06700	Tonala	43	Mexico
3.	New York	10003	Irving Place	30	USA
4.	Paris	75116	Avenue d´Iena	17	France
5.	Rome	00198	Via Savoia	15	Italy

To forward geocode the five locations given above, we specify `opencagegeo` as follows:

```
. opencagegeo, key(YOUR-KEY-HERE) street(STREET) number(NUMBER) postcode(POSTCODE)
> country(COUNTRY)
OpenCage geocoded 1 of 5
  (output omitted)
OpenCage geocoded 5 of 5
```

g_quality	Freq.	Percent	Cum.
street	1	20.00	20.00
number	4	80.00	100.00
Total	5	100.00	

```
Data generated is jointly licensed under the ODbL and CC-BY-SA licenses.
```

The output table produced by `opencagegeo` illustrates that four out of five addresses were identified at the highest quality level, i.e. *number*. For one location, the exact house number was not found and the resulting quality level is *street*.

[9] The Goethe Institute is a non-profit German cultural association promoting the German language, the addresses are obtained from `https://www.goethe.de/de/wwt.html`.

Reverse Geocoding

Now, we want to reverse geocode the latitudes and longitudes generated above. First, we rename the variables and then specify `opencagegeo` using `latitude()`, `longitude()` and the `replace` option.

```
. rename g_lat LATITUDE
. rename g_lon LONGITUDE
. opencagegeo, key(YOUR-KEY_HERE) latitude(LATITUDE) longitude(LONGITUDE) replace
OpenCage geocoded 1 of 5
  (output omitted)
OpenCage geocoded 5 of 5
```

g_quality	Freq.	Percent	Cum.
street	1	20.00	20.00
number	4	80.00	100.00
Total	5	100.00	

```
Data generated is jointly licensed under the ODbL and CC-BY-SA licenses.
```

Not surprisingly, we find that four latitude longitude pairs are identified at the *number* and one at the *street* level.

4.2 Opencagegeoi

Forward Geocoding

If we intend to geocode a single location only and want the results to be displayed in the output window rather than saved as variables, we should use the immediate version `opencagegeoi`. Note, however, that we need to define a global macro *mykey* first.[10]

```
. global mykey YOUR-KEY-HERE
```

To obtain the geographic coordinates of the Goethe Institute London, we specify `opencagegeoi` as following:

```
. opencagegeoi 50 Exhibition Road, London SW7 2PH, UK

********************************
*** OpenCage Geocoder Results ***
********************************
Formatted address: Goethe Institute, 50-51 Exhibition Road, London SW7 1BF, UK
Latitude: 51.4994811
Longitude: -0.174013268370617
```

Reverse Geocoding

To retrieve the postal address from geographic coordinates - in our example of the Goethe Institute in London - we type:

[10] The global macro needs to be specified once when a new Stata window is opened.

```
. opencagegeoi 51.4994811,-0.174013268370617

********************************
*** OpenCage Geocoder Results ***
********************************
Formatted address: Goethe Institute, 50-51 Exhibition Road, London SW7 1BF, UK
Latitude: 51.4994811
Longitude: -0.174013268370617
```

5 Appendix

The OpenCage Geocoder API returns its results in JSON (JavaScript Object Notation) format. The respective information is then extracted and stored in the variables described above (or displayed immediately in the case of `opencagegeoi`). As the underlying OpenStreetMap (OSM) data is provided by many contributors, keys are not uniquely defined. E.g. the key leading to the street name is not always *street*, but may be *street_name*, *road* etc. instead. Unfortunately, there exists no exhaustive list of all these aliases, however `opencagegeo` uses the most common ones to generate its output variables.[11] The so-called JSON paths which are used for extracting the data are reported in the table below.

Variable	JSON path (aliases)
g_latitude	results:1:geometry:lat
g_longitude	results:1:geometry:lng
g_street	results:1:components:street (street_name, road, residential, footway, pedestrian)
g_number	results:1:components:house_number
g_postcode	results:1:components:postcode
g_city	results:1:components:city (town, village, hamlet)
g_county	results:1:components:county
g_state	results:1:components:state
g_country	results:1:components:country
g_formatted	results:1:components:formatted
g_confidence	results:1:components:confidence

The JSON paths used by `opencagegeoi` are the same as for the respective variables generated by `opencagegeo`.

[11] If none of the keys used by `opencagegeo` lead to the requested information, an empty string in the respective variable will be returned. However, the information will be part of the well-formatted address in `g_formatted`.

6 Acknowledgements

I would like to thank Ed Freyfogle of OpenCage Data for his support and Achim Ahrens for testing `opencagegeo` and helping improve it. I also thank the authors of the existing geocoding routines for Stata, especially Adam Ozimek and Daniel Miles (`geocode`). Parts of `opencagegeo` build upon their code. The immediate version was added at Kit Baum's suggestion. All remaining errors are my own. Comments and suggestions are highly appreciated.

7 References

Anderson, M. 2013. GEOCODEOPEN: Stata module to geocode addresses using MapQuest Open Geocoding Services and Open Street Maps `https://ideas.repec.org/c/boc/bocode/s457733.html`.

Ansari, M. R. 2015. GCODE: Stata module to download Google geocode data `https://ideas.repec.org/c/boc/bocode/s457969.html`.

Bernhard, S. 2013. GEOCODE3: Stata module to retrieve coordinates or addresses from Google Geocoding API Version 3 .

Cox, N. J. 2014. EGENMORE: Stata modules to extend the generate function `https://ideas.repec.org/c/boc/bocode/s386401.html`.

Hess, S. 2015. GEOCODEHERE: Stata module to provide geocoding relying on Nokias Here Maps API `https://ideas.repec.org/c/boc/bocode/s458048.html`.

Lindsley, E. 2012a. INSHEETJSON: Stata module for importing tabular data from JSON sources on the internet `https://ideas.repec.org/c/boc/bocode/s457407.html`.

———. 2012b. LIBJSON: Stata module to provide Mata class library for obtaining and parsing JSON strings into object trees `https://ideas.repec.org/c/boc/bocode/s457406.html`.

Ozimek, A., and D. Miles. 2011. Stata utilities for geocoding and generating travel time and travel distance information. *The Stata Journal* 11(1): pp. 106–119.

About the author

Lars Zeigermann is a Ph.D. candidate at the Düsseldorf Graduate School in Economics (DGSE) of the Düsseldorf Insitute for Competition Economics (DICE).