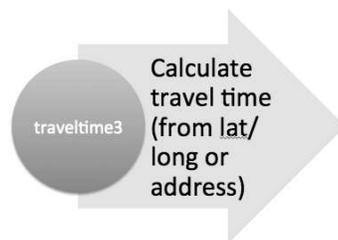


Geocoding in Stata

1. Geocode (`geocode3`)
 - Geocode (address -> coordinates)
 - Reverse Geocode (coordinates -> address)
2. Generate a and view a map file
 - a. Generate a map file (`kmlwriter`)
 - b. View a Map
 - Google Maps
 - Google Earth
3. Generate Travel time (`traveltime3`)



geocode3

Purpose

Retrieves coordinates via Google Geocoding API V3 (geocoding)

or

Retrieves addresses via Google Geocoding API V3 (reverse geocoding)

Syntax

geocode3, address(varname) fulladdress quality zip state number street ad1 ad2 ad3 sub |
reverse coord(varname)

- Specify address(varname) to retrieve coordinates from addresses provided in address(varname), or
- specify coord(varname) to retrieve addresses from coordinates provided in coord(varname). The other options are available for both normal and reverse geocoding.

e.g.

geocode3, address(varname) zip state

geocode3, coord(varname) zip state number street ad2

Install

- ssc install geocode3
- geocode3 requires insheetjson, which in turn requires libjson. Install both from ssc:
 - ssc install insheetjson
 - ssc install libjson

Limits

- Google Maps api has a daily query limit of 2500 per IP-address. The status code "OVER_QUERY_LIMIT" is returned and the program stops when the limit is reached. Restart geocode on the next day and it will automatically resume where it stopped. Or get a new IP.
- There is a 500ms delay between queries so that Google's servers don't reject them for overflowing, so take your time while geocoding.
- The Geocoding API may only be used in conjunction with a Google map; geocoding results without displaying them on a map is prohibited. For complete details on allowed usage, consult the Maps API Terms of Service License Restrictions.

Required for Geocoding (calculate lat/long from address)

- address(varname)
- The address has to be coded in a single string variable similar to this:

 number+street+zip+town+state
- Use your own regional conventional address format, Google is quite clever:
 - 1600+Amphitheatre+Parkway,+Mountain+View,+CA
 - 1+Friedrich-Schmidt-Platz+1010+Wien+Austria
 - Bahnhofstrasse+1+12555+Berlin+Deutschland
 - 2+Rue+de+Viarmes+75001+Paris+France

Required for reverse Geocoding (calculate address from lat/long)

coord(varname)

The coords need to be in a single string variable with the format:

lat,lon

E.g.:

47.9948972,16.9287615

For more information:

- <https://developers.google.com/maps/documentation/distancematrix/>
- in Stata: help geocode3

writekml

Purpose

writekml takes latitude and longitude information from a data set in memory and writes a KML file (Keyhole Markup Language) in the current working directory.

Syntax

```
writekml, filename(string.kml) plcategory(varname string) [pname(varname string)
pldesc(varname string)]
```

Options

filename(string.kml) is the name of the KML file to be written. If it already exists, it will be replaced.

plcategory(varname string) is the name of the variable that groups the places being mapped. Each group is mapped with paddles (flags) of the same color. The Google Maps API makes available eight colors. Trying to map more will return an error message.

pname(varname string) is the name of the variable that stores the names of the places being mapped. These names will show up in bold black letters on the list next to the Google map that the KML file renders.

pldesc(varname string) is the name of the variable that stores descriptions of the places being mapped. The descriptions will show up in smaller, lighter-colored type below the place names in the list.

View

- Make sure in your current working directory and the extension of the kml file is .kml
- View
 - In Google Earth, File Open and browse to your .kml file
or
 - Copy the .kml file to Google Drive and double click on it to open in Google Maps (or upload it to a web site and paste the URL into Google maps search window)

Remarks

writekml expects latitude and longitude to be present in the data set as numeric variables with the same names. If place names or descriptions are missing writekml will fill in the category name instead. The easiest way to associate latitude and longitude information with a data set of physical addresses is to (install and) run geocode3 before running writekml.

Example

```
writekml, filename(kmlout.kml) pname(name) pldesc(description) plcategory(kind)
```

traveltime3

Purpose

traveltime3 retrieves travel time and road distance between two locations.

Install

- ssc install traveltime3
- traveltime3 requires insheetjson, which in turn requires libjson. Install both from ssc:
 - ssc install insheetjson
 - ssc install libjson

Syntax

traveltime3, start(string) end(string) mode(string) units(string) avoid(string)

Output

traveltime3 creates several variables and will overwrite existing ones. They have t_ prefixes, so you won't confuse them with previous variables.

traveltime3 creates the variables:

- t_status, which is "OK" if everything went as expected.
- t_origin, which contains the origin address interpreted from the input you specified.
- t_destination, which contains the destination address interpreted from your input.

Limits

- Google Maps api has a daily query limit of 2500 per IP-address. The status code "OVER_QUERY_LIMIT" is returned and the program stops when the limit is reached. Restart traveltime3 on the next day and it will automatically resume where it stopped. Or get a new IP.
- There is a 500ms delay between queries so that Google's servers don't reject them for overflowing, so take your time while geocoding.
- Be aware of Google's usage limits. Especially: Use of the Distance Matrix API must relate to the display of information on a Google Map; for example, to determine origin-destination pairs that fall within a specific driving time from one another, before requesting and displase destinations on a map. Use of the service in an application that doesn't display a Google map is prohibited.

Options

required:

start(string) end(string)

Can be either filled with addresses or coordinates.

With addresses:

- The address has to be coded in a single string variable similar to this:
 number+street+zip+town+state
- Use your regional conventional address format and you should be fine, Google is quite clever.
- You can also omit certain entries (e.g. only use zip+state).
- Make sure that there are no spaces or special characters that Stata cannot handle in your addresses (like %, ^, ,, fl).
- Spaces can be replaced with "+".
 - 1600+Amphitheatre+Parkway,+Mountain+View,+CA
 - 1+Friedrich-Schmidt-Platz+1010+Wien+Austria
 - Bahnhofstrasse+1+12555+Berlin+Deutschland
 - 2+Rue+de+Viarmes+75001+Paris+France
 - 4040+Austria

With coordinates:

The coordinates need to be in a single string variable with the format:

lat,lon

E.g.:

47.9948972,16.9287615

optional:

mode(string)

Can be driving, bicycling or walking; when not specified defaults to driving

units(string)

- Can be metric (km) or imperial (mi); when not specified defaults to metric.
- Time is always returned in minutes.

For more information:

- <https://developers.google.com/maps/documentation/distancematrix/>
- In Stata: `help traveltime3`

Other Stata mapping options

Using web resources

geocode3, traveltime3, and writekml, and the two resources below are all Geoweb resources based on api links to Google.

- Display .kml file in Google Maps or Google Earth
<http://chronicle.com/blogs/profhacker/easily-open-a-kml-file-in-google-maps/34835>
- Interactively zoom to one place to find a latitude and longitude
<http://maps.cga.harvard.edu/gpf/>
- Create a map from a list of addresses
 - <https://productforums.google.com/forum/#!topic/maps/baw9A5LuS3o>
 - http://google.about.com/od/googlemapsandmashups/qt/map_a_list.htm (third party)

Using legacy resources

The following are slightly more traditional tools for Stata that do not utilize Google.

- <http://www.williamdoerner.com/stata/making-maps>
- Maurizio Pisati presentation on overlaying data visually on a map ([link to PDF](#))
- <http://statadaily.ikonomiya.com/2011/03/20/fun-with-maps-in-stata/>
- <http://huebler.blogspot.com/2005/11/creating-maps-with-stata.html>