The Stata Journal (2011)
11, Number 1, pp. 106–119

Stata utilities for geocoding and generating travel time and travel distance information

Adam Ozimek
Econsult Corporation
Philadelphia, PA
ozimek@econsult.com

Daniel Miles
Econsult Corporation
Philadelphia, PA
miles@econsult.com

Abstract. This article describes `geocode` and `traveltime`, two commands that use Google Maps to provide spatial information for data. The `geocode` command allows users to generate latitude and longitude for various types of locations, including addresses. The `traveltime` command takes latitude and longitude information and finds travel distances between points, as well as the time it would take to travel that distance by either driving, walking, or using public transportation.

Keywords: dm0053, geocode, traveltime, Google Maps, geocoding, ArcGIS

1 Introduction

The location and spatial interaction of data has long been important in many scientific fields, from the social sciences to environmental and natural sciences. The increased use and availability of geographic information systems (GIS) software has allowed researchers in a growing range of disciplines to incorporate space in their models. In addition, businesses, governments, and the nonprofit sector have found spatial information useful in their analysis.

A crucial first step in incorporating space into analysis is to identify the spatial location of the data. For data that represent points on roads, one way to do this is to "geocode" observations. Geocoding is the process of converting addresses or locations (such as "3600 Market Street, Philadelphia, PA" or simply "Philadelphia, PA") into geographic coordinates (such as [39.95581, −75.19466] or [39.95324, −75.16739]). Once the geographic coordinates are known, the data can be mapped and spatial relationships between data points can be incorporated into analyses.

Geocoding can easily be accomplished using ArcGIS or similar high-end mapping software. However, for those users who are unfamiliar with GIS, this process requires a substantial investment of time to learn the basics of the software and its geocoding capabilities, and often the software's expense is prohibitive.

Alternatively, one can geocode using Google's free mapping website, Google Maps.[1] However, the website is designed to be used for one or two addresses at a time, and using it would be an extremely time-consuming way to geocode and find distances between more than a handful of points. The commands discussed in this article combine the convenience of a high-end software package, like ArcGIS, with the free services of Google Maps.

The `geocode` command automates the geocoding service that is included in the Google Geocoding API (application programming interface) to easily and quickly batch geocode a small set of addresses. The `traveltime` command uses Google Maps to calculate the travel time between a set of geographic coordinates (latitude and longitude) using a variety of transportation modes. The use of these commands in tandem will allow researchers who are unfamiliar with the use of GIS or those without access to GIS the ability to quickly and easily incorporate spatial interactions into their research.

2 The geocode command

2.1 Syntax

`geocode,` [`address(`*varname*`)` `city(`*varname*`)` `state(`*varname*`)` `zip(`*varname*`)` `fulladdr(`*varname*`)`]

See *Remarks* for details on specifying options.

2.2 Options

`address(`*varname*`)` specifies the variable containing a street address. *varname* must be a string. Cleaned address names will provide better results, but the program performs some basic cleaning.

`city(`*varname*`)` specifies the variable containing the name or common abbreviation for the city, town, county, metropolitan statistical area (MSA),[2] or equivalent. *varname* must be a string.

`state(`*varname*`)` specifies the variable containing the name or the two-letter abbreviation of the state of the observation. An example of such an abbreviation is Pa for Pennsylvania. *varname* must be a string.

1. http://maps.google.com
2. MSA refers to a geographic entity defined by the United States Office of Management and Budget for use by federal statistical agencies in collecting, tabulating, and publishing federal statistics. An MSA contains a core urban area of 50,000 or more population. Each MSA consists of one or more counties that contain the core urban area as well as any adjacent counties that have a high degree of social and economic integration with the urban core.

`zip(`*varname*`)` specifies the variable containing the standard United States Postal Service 5-digit postal zip code[3] or zip code +4. If zip code +4 is specified, it should be in the form 12345-6789. *varname* must be a string.

`fulladdr(`*varname*`)` allows users to specify all or some of the above options in a single string. *varname* must be a string and should be in a format that would be used to enter an address using http://maps.google.com. Standard formats are listed in section 2.3.

2.3 Remarks

When geocoding within the United States, one or all of the options `address(`*varname*`)`, `city(`*varname*`)`, `state(`*varname*`)`, and `zip(`*varname*`)` may be specified, with more information allowing for a higher degree of geocoding detail. This allows for the geocoding of zip codes, counties, cities, or other geographic areas. In general, when a specific street address is not specified, a latitude and longitude will be provided for a central location within the specified city, state, or zip code. The same option for specifying geographic detail applies using `fulladdr()`.

When geocoding outside the United States, the `fulladdr()` option must be used and the country must be specified. When inputting data using `fulladdr()`, any string that would be usable with http://maps.google.com is in an acceptable format. Acceptable examples for `fulladdr()` in the United States include but are not limited to these formats:

"street address, city, state, zip code"
"street address, city, state"
"city, state, zip code"
"city, state"
"state, zip code"
"state"

Acceptable examples for `fulladdr()` outside the United States include but are not limited to these formats:

"street address, city, state, country"
"street address, city, country"
"city, state, country"
"city, country"

Country should be specified using the full country name. State can be whatever regional entity exists below the country level—for instance, Canadian provinces or Japanese prefectures. Again, format acceptability may be gauged using the Google Maps website.

The `geocode` command queries Google Maps, which allows for a fair degree of tolerance in how addresses can be entered and still be geocoded correctly. The inputs are

3. Zip codes are postal codes used by the United States Postal Service, the independent government agency that provides postal service in the United States.

not case sensitive and are robust to a wide range of abbreviations and spelling errors. For instance, each of the following would be an acceptable way to enter the same street address:

"123 Fake Street"
"123 Fake St."
"123 fake st"

Common abbreviations for cities, states, towns, counties, and other relevant geographies are also often acceptable. For instance, it is fine to use "Phila" for "Philadelphia", "PA" for "Pennsylvania", "NYC" for "New York City", and "UK" for "United Kingdom". The program is also fairly robust to spelling errors; it is capable of accepting "Allantown, PA" for "Allentown, PA". It is recommended that addresses be as accurate as possible to avoid geocoding errors, but the program is as flexible as Google Maps.

The `geocode` command generates four new variables: `geocode`, `geoscore`, `latitude`, and `longitude`. `latitude` and `longitude` contain the geocoded coordinates for each observation in decimal degrees. The `geocode` variable contains a numerical indicator of geocoding success or type of failure, and `geoscore` provides a measure of accuracy. These values and their definitions are provided by Google Maps. For more information, see http://code.google.com/apis/maps/documentation/geocoding/.

`geocode` error definitions:

200 = no errors
400 = incorrectly specified address
500 = unknown failure reason
601 = no address specified
602 = unknown address
603 = address legally or contractually ungeocodable
620 = Google Maps query limit reached

`geoscore` accuracy level:

0 = unknown accuracy
1 = country-level accuracy
2 = region-level (state, province, etc.) accuracy
3 = subregion-level (county, municipality, etc.) accuracy
4 = town-level (city, village, etc.) accuracy
5 = postal code–level (zip code) accuracy
6 = street-level accuracy
7 = intersection-level accuracy
8 = address-level accuracy
9 = premise-level (building name, property name, store name, etc.) accuracy

Google Maps appears to limit the number of queries allowed from a single Internet protocol (IP) address within a 24-hour period. This exact limit is not known, but it is not recommended that more than 10,000 to 15,000[4] observations be geocoded at any one time from a single IP address.

Data acquired using `geocode` are subject to Google's terms of service, specified here: http://code.google.com/apis/maps/terms.html.

2.4 Standard geocoding example

Start with, for example, a dataset of survey respondent addresses for which an analyst wishes to retrieve latitude and longitude coordinates. The data are as follows:

id	resp_street	resp_city	resp_st	resp_zp
1	1500 Market St	Philadelphia	PA	19102
2	2124 Fairmount Ave	Philadelphia	PA	19130
3	2600 Benjamin Franklin Pkwy	Philadelphia	PA	19130
4	1219 S 9th St	Philadelphia	PA	19147
5	420 Chestnut St	Philadelphia	PA	19106
6	8500 Essington Ave	Philadelphia	PA	19153
7	3600 Market St	Philadelphia	PA	19104
8	1455 Franklin Mills Circle	Philadelphia	PA	19154
9	1901 Vine Street	Philadelphia	PA	19103
10	1801 N Broad St	Philadelphia	PA	19122

The analyst can geocode these data using the following command:

```
. geocode, address(resp_street) city(resp_city) state(resp_st) zip(resp_zp)
Geocoding 1 of 10
Geocoding 2 of 10
Geocoding 3 of 10
Geocoding 4 of 10
Geocoding 5 of 10
Geocoding 6 of 10
Geocoding 7 of 10
Geocoding 8 of 10
Geocoding 9 of 10
Geocoding 10 of 10
```

Upon completion, the geocoded data will include four extra variables, as follows:

4. We have successfully geocoded over 15,000 observations on several occasions, but to be conservative, we do not recommend attempting to geocode more than 15,000 at any one time.

id	resp_street	resp_city	resp_st	resp_zp
1	1500 Market St	Philadelphia	PA	19102
2	2124 Fairmount Ave	Philadelphia	PA	19130
3	2600 Benjamin Franklin Pkwy	Philadelphia	PA	19130
4	1219 S 9th St	Philadelphia	PA	19147
5	420 Chestnut St	Philadelphia	PA	19106
6	8500 Essington Ave	Philadelphia	PA	19153
7	3600 Market St	Philadelphia	PA	19104
8	1455 Franklin Mills Circle	Philadelphia	PA	19154
9	1901 Vine Street	Philadelphia	PA	19103
10	1801 N Broad St	Philadelphia	PA	19122

geocode	geoscore	latitude	longitude
200	8	39.95239	-75.16619
200	8	39.96706	-75.17309
200	8	39.96561	-75.18099
200	8	39.93365	-75.15895
200	8	39.94889	-75.14804
200	8	39.89513	-75.22896
200	8	39.95581	-75.19466
200	8	40.09012	-74.95904
200	8	39.9593	-75.1711
200	8	39.98027	-75.15705

The `geocode` score of 200 indicates no errors in geocoding, and the `geoscore` score of 8 indicates that the geocoding was completed at address-level accuracy.

2.5 Using the fulladdr() option

The previous example showed how to geocode when the address, city, state, and zip were each specified in separate string variables. Alternatively, the full address could appear as a single string variable, as shown in the following output:

id	resp_addr
1	1500 Market St, Philadelphia, PA 19102
2	2124 Fairmount Ave, Philadelphia, PA 19130
3	2600 Benjamin Franklin Pkwy, Philadelphia, PA 19130
4	1219 S 9th St, Philadelphia, PA 19147
5	420 Chestnut St, Philadelphia, PA 19106
6	8500 Essington Ave, Philadelphia, PA 19153
7	3600 Market St, Philadelphia, PA 19104
8	1455 Franklin Mills Circle, Philadelphia, PA 19154
9	1901 Vine Street, Philadelphia, PA 19103
10	1801 N Broad St, Philadelphia, PA 19122

These data, which refer to the exact same locations as those in section 2.4, can be geocoded using the following command:

```
geocode, fulladdr(resp_addr)
```

The coordinates, `geocode`s, and `geoscore`s that are produced are identical to those produced in the previous example.

2.6 Geocoding larger geographical areas

Rather than being concerned with specific street addresses, an analyst might be concerned with the geographic location of particular zip codes, cities, counties, or even states. `geocode` can calculate the latitude and longitude of these geographic areas using Google Maps. Google does not explicitly state how the "centers" of these locations are determined, though in general, it appears that central downtown areas are used for cities and towns, and geographic centroids are used for zip codes, states, and other larger regions. If, for example, the analyst needed to know the latitudes and longitudes corresponding to the centers of the zip codes for the 10 previously used observations, the following command could be issued:

```
. geocode, state(resp_st) zip(resp_zp)
Geocoding 1 of 10
Geocoding 2 of 10
Geocoding 3 of 10
Geocoding 4 of 10
Geocoding 5 of 10
Geocoding 6 of 10
Geocoding 7 of 10
Geocoding 8 of 10
Geocoding 9 of 10
Geocoding 10 of 10
```

This command produces the following dataset of coordinates, `geocode`s, and `geoscore`s:

id	resp_street	resp_city	resp_st	resp_zp
1	1500 Market St	Philadelphia	PA	19102
2	2124 Fairmount Ave	Philadelphia	PA	19130
3	2600 Benjamin Franklin Pkwy	Philadelphia	PA	19130
4	1219 S 9th St	Philadelphia	PA	19147
5	420 Chestnut St	Philadelphia	PA	19106
6	8500 Essington Ave	Philadelphia	PA	19153
7	3600 Market St	Philadelphia	PA	19104
8	1455 Franklin Mills Circle	Philadelphia	PA	19154
9	1901 Vine Street	Philadelphia	PA	19103
10	1801 N Broad St	Philadelphia	PA	19122

geocode	geoscore	latitude	longitude
200	5	39.9548	-75.1656
200	5	39.96883	-75.17586
200	5	39.96883	-75.17586
200	5	39.93567	-75.15173
200	5	39.9493	-75.14471
200	5	39.89152	-75.22866
200	5	39.96157	-75.19677
200	5	40.09333	-74.98056
200	5	40.03131	-75.1698
200	5	39.97274	-75.1246

The `geocode` of 200 indicates that there were no errors and the geocoding was performed correctly. The `geoscore` of 5 indicates that the observations were geocoded at zip code–level accuracy, as expected. Notice that the second and third observations are in the same zip code, and so their latitudes and longitudes are identical.

3 The traveltime command

The geographic distance between data points is often important information in applied research. When geographic coordinates are known, estimating straight-line distance between points can be done using either simple Euclidean distance or a more complex formula, such as the Haversine formula, that takes the curvature of the earth into consideration. However, accurately estimating the driving distance (rather than the straight line or as-the-crow-flies distance) is complicated. Some of the factors that must be taken into consideration include the available network of streets, one-way streets, and the shortest choice among alternative routes.

Even more complicated to estimate than driving distance is driving time. An accurate measure of this includes traffic congestion, speed limits, turning time, and stop signs and traffic lights. The problem is exponentially magnified when mode choice—that is, traveling by car versus taking public transportation—is taken into consideration.

These difficulties explain why straight-line distance is an often-used shortcut. However, real driving time or travel time can be integral in many applications, and a straight-line method often introduces errors that can be correlated with other important variables. For instance, in some urban areas, road and traffic congestion density may be inversely correlated with resident income because lower income people are more likely to live in more densely populated areas. Therefore, if an estimate of the impact of income on an individual's willingness to travel used straight-line distance, that estimate would be biased downward. Moreover, drivers have to travel more road miles and travel for more time to drive a straight-line mile in a city than in a suburb or rural area.

A market study that used straight-line distance to estimate which zip codes lie within the catchment area of a particular store location would overestimate the number of urban zip codes and underestimate the number of suburban or rural zip codes. This is particularly problematic because urban, rural, and suburban zip codes may have differences in average demographics, which would bias estimates in ways that are critically altering to the market study.

3.1 Syntax

The `traveltime` command is designed to work in tandem with the `geocode` command discussed above. The `traveltime` command uses the following syntax:

`traveltime, start_x(`*varname*`) start_y(`*varname*`) end_x(`*varname*`)`
`end_y(`*varname*`)` [`mode(`*varname*`) km`]

3.2 Options

`start_x(`*varname*`)` specifies the variable containing the geocoded x coordinate of the starting point. `start_x()` is required.

`start_y(`*varname*`)` specifies the variable containing the geocoded y coordinate of the starting point. `start_y()` is required.

`end_x(`*varname*`)` specifies the variable containing the geocoded x coordinate of the destination. `end_x()` is required.

`end_y(`*varname*`)` specifies the variable containing the geocoded y coordinate of the destination. `end_y()` is required.

`mode(`*varname*`)` specifies the mode choice of the trip. The values are set to `1` for car, `2` for public transportation, and `3` for walking. The default mode is car.

`km` specifies that `traveltime_dist` be reported in kilometers rather than in miles (the default).

3.3 Remarks

Both starting coordinates (`start_x()`, `start_y()`) and ending coordinates (`end_x()`, `end_y()`) are required inputs. `traveltime` queries Google Maps, which requires that the coordinates be in decimal degrees. We suggest using the `geocode` command to geocode both the start and the end points of the trip before proceeding with the `traveltime` command. Beginning with `geocode` will ensure that the coordinates are in the correct format for Google Maps and will reduce the possibility for errors. However, the use of the `geocode` command is not necessary if the start and end coordinates are known and are reported in decimal degrees.

The `mode()` option is optional. The availability of this option is limited by Google Maps, which does not provide the multiple mode choices for all geographic areas. This especially pertains to the public transportation option, which is currently available in only a limited number of places, mainly in the United States.

The `traveltime` command generates four new variables: `days`, `hours`, `mins`, and `traveltime_dist`. The combination of the `days`, `hours`, and `mins` variables contains the days, hours, and minutes portion of the time that it takes to travel between the origin and the destination. The `traveltime_dist` variable contains the distance between the starting and ending points.

3.4 Standard traveltime example

Suppose you need to calculate the travel time between Philadelphia, PA, and other major cities in Pennsylvania. After using the `geocode` command, the data might look like this:

start_city	end_city	start_~g	start_lat	end_long	end_lat
Philadelphia, PA	Allentown, PA	39.95234	-75.16379	40.60843	-75.49018
Philadelphia, PA	York, PA	39.95234	-75.16379	39.9626	-76.72775
Philadelphia, PA	Lancaster, PA	39.95234	-75.16379	40.03788	-76.30551
Philadelphia, PA	Harrisburg, PA	39.95234	-75.16379	40.2737	-76.88441
Philadelphia, PA	Pittsburgh, PA	39.95234	-75.16379	40.44062	-79.99589
Philadelphia, PA	Erie, PA	39.95234	-75.16379	42.12922	-80.08506
Philadelphia, PA	Scranton, PA	39.95234	-75.16379	41.40897	-75.66241
Philadelphia, PA	Wilkes-Barre, PA	39.95234	-75.16379	41.24591	-75.88131
Philadelphia, PA	Johnstown, PA	39.95234	-75.16379	40.32674	-78.92197
Philadelphia, PA	Reading, PA	39.95234	-75.16379	40.33565	-75.92687

You can calculate the travel time between the start and end points for each observation by using the following command:

```
. traveltime, start_x(start_lat) start_y(start_long) end_x(end_lat)
> end_y(end_long)
Processed 1 of 10
Processed 2 of 10
Processed 3 of 10
Processed 4 of 10
Processed 5 of 10
Processed 6 of 10
Processed 7 of 10
Processed 8 of 10
Processed 9 of 10
Processed 10 of 10
```

Upon completion of the `traveltime` command, the data look as follows:

start_city	end_city	start_~g	start_lat	end_long
Philadelphia, PA	Allentown, PA	39.95234	-75.16379	40.60843
Philadelphia, PA	York, PA	39.95234	-75.16379	39.9626
Philadelphia, PA	Lancaster, PA	39.95234	-75.16379	40.03788
Philadelphia, PA	Harrisburg, PA	39.95234	-75.16379	40.2737
Philadelphia, PA	Pittsburgh, PA	39.95234	-75.16379	40.44062
Philadelphia, PA	Erie, PA	39.95234	-75.16379	42.12922
Philadelphia, PA	Scranton, PA	39.95234	-75.16379	41.40897
Philadelphia, PA	Wilkes-Barre, PA	39.95234	-75.16379	41.24591
Philadelphia, PA	Johnstown, PA	39.95234	-75.16379	40.32674
Philadelphia, PA	Reading, PA	39.95234	-75.16379	40.33565

end_lat	days	hours	mins	travel~t
-75.49018	0	1	9	61.5
-76.72775	0	1	56	102
-76.30551	0	1	32	73
-76.88441	0	1	56	107
-79.99589	0	5	15	305
-80.08506	0	6	43	420
-75.66241	0	2	13	125
-75.88131	0	2	2	113
-78.92197	0	4	13	239
-75.92687	0	1	10	57.6

As illustrated in the above table, it takes one hour and nine minutes to drive the 61.5 miles from Philadelphia, PA, to Allentown, PA; it takes five hours and fifteen minutes from Philadelphia, PA, to Pittsburgh, PA; and it takes six hours and forty-three minutes to drive from Philadelphia, PA, to Erie, PA.

The data in the above example did not have a travel mode specified, so by default, Google Maps calculated the travel time for an automobile trip between the start and end points. If you had data on the transportation mode[5] used for each trip, your data might look like this:

start_city	end_city	start_~g	start_lat	end_long	end_lat	mode
Philadelphia, PA	Allentown, PA	39.95234	-75.16379	40.60843	-75.49018	1
Philadelphia, PA	York, PA	39.95234	-75.16379	39.9626	-76.72775	3
Philadelphia, PA	Lancaster, PA	39.95234	-75.16379	40.03788	-76.30551	3
Philadelphia, PA	Harrisburg, PA	39.95234	-75.16379	40.2737	-76.88441	1
Philadelphia, PA	Pittsburgh, PA	39.95234	-75.16379	40.44062	-79.99589	1
Philadelphia, PA	Erie, PA	39.95234	-75.16379	42.12922	-80.08506	3
Philadelphia, PA	Scranton, PA	39.95234	-75.16379	41.40897	-75.66241	3
Philadelphia, PA	Wilkes-Barre, PA	39.95234	-75.16379	41.24591	-75.88131	3
Philadelphia, PA	Johnstown, PA	39.95234	-75.16379	40.32674	-78.92197	1
Philadelphia, PA	Reading, PA	39.95234	-75.16379	40.33565	-75.92687	1

The syntax for the `traveltime` command would then be

```
. traveltime, start_x(start_lat) start_y(start_long) end_x(end_lat)
> end_y(end_long) mode(mode)
Processed 1 of 10
Processed 2 of 10
Processed 3 of 10
Processed 4 of 10
Processed 5 of 10
Processed 6 of 10
Processed 7 of 10
Processed 8 of 10
Processed 9 of 10
Processed 10 of 10
```

This command produces the following dataset of travel times:

5. We use only the automobile transportation mode and walking. We did not include the public transportation mode in the example because, as mentioned before, Google Maps does not currently support travel times using public transportation in many areas.

start_city	end_city	start_~g	start_lat	end_long
Philadelphia, PA	Allentown, PA	39.95234	-75.16379	40.60843
Philadelphia, PA	York, PA	39.95234	-75.16379	39.9626
Philadelphia, PA	Lancaster, PA	39.95234	-75.16379	40.03788
Philadelphia, PA	Harrisburg, PA	39.95234	-75.16379	40.2737
Philadelphia, PA	Pittsburgh, PA	39.95234	-75.16379	40.44062
Philadelphia, PA	Erie, PA	39.95234	-75.16379	42.12922
Philadelphia, PA	Scranton, PA	39.95234	-75.16379	41.40897
Philadelphia, PA	Wilkes-Barre, PA	39.95234	-75.16379	41.24591
Philadelphia, PA	Johnstown, PA	39.95234	-75.16379	40.32674
Philadelphia, PA	Reading, PA	39.95234	-75.16379	40.33565

end_lat	mode	days	hours	mins	travel~t
-75.49018	1	0	1	9	61.5
-76.72775	3	1	5	0	87.2
-76.30551	3	0	21	13	63.3
-76.88441	1	0	1	56	107
-79.99589	1	0	5	15	305
-80.08506	3	5	0	0	362
-75.66241	3	1	16	0	119
-75.88131	3	1	14	0	112
-78.92197	1	0	4	13	239
-75.92687	1	0	1	10	57.6

When the travel mode between Philadelphia, PA, and York, PA, is switched from automobile to walking, the travel time increases from one hour and fifty-six minutes to one day and five hours, while the distance decreases from 102 miles to 87.2 miles. The difference in the distance is due to the fact that when travel mode changes, the travel route might also change. When the mode of travel between Philadelphia, PA, and Scranton, PA, is changed from automobile, the travel time increases from two hours and thirteen minutes to one day and sixteen hours, while the distance decreases from 125 miles to 119 miles.

`traveltime` faces the same queries limit from Google Maps as with the `geocode` command. Although it is not clear what the exact limit is, we do not recommend attempting to use the `traveltime` command with more than 10,000 to 15,000 latitude and longitude pairs at any one time.

4 Conclusions

Geographic information is often important in economic, epidemiological, and sociological research. The driving distance and time between locations and the geocoded addresses are useful in a wide variety of research. When dealing with a small set of addresses or latitude/longitude coordinate pairs, researchers can use Google Maps to find latitude and longitude or to find driving distance and drive time. Likewise, users with ArcGIS

or similar high-end mapping software can get the information for larger numbers of observations. The `geocode` and `traveltime` commands allow users to geocode and estimate travel time and travel distance for datasets within Stata by querying Google Maps. This approach provides users with the convenience of a high-end software package like ArcGIS and the free services of Google Maps.

5 Acknowledgments

We thank Graeme Blair for suggestions and editing, and Econsult Corporation for time and resources to work on this project. We would also like to acknowledge Franz Buscha and Lionel Page, for providing us with their `gmap` command, and Mai Nguyen, Shane Trahan, Patricia Nguyen, and Wafa Handley, whose work integrating Google Maps and SAS was helpful.

About the authors

Daniel Miles and Adam Ozimek are associates at Econsult Corporation, an economics consulting firm from Philadelphia, PA, that provides economic research in support of litigation, as well as economic consulting services.