Invoking Google Maps with SAS® to Determine Driving Distances

Michael A. Raithel, Westat, Inc.

ABSTRACT

SAS programmers are sometimes asked to determine the distance between addresses. Perhaps we are trying to establish which study subject persons are within a specific distance of an interviewer. Maybe we want to identify existing establishments such as doctors offices, hospitals, clinics, or pharmacies that are a reasonable distance from study subjects. Such distances are often measured in absolute terms; such as *all of the doctor offices within 50 miles* of a subject. But, straight-line, as-the-crow-flies, distances do not account for geographical and infrastructure features such as mountains, rivers, lakes, and existing highways and surface roads. Consequently, driving distances may be more reliable than absolute distances.

When many of us want to get from *Here* to *There*, we invoke Google Maps, plug in the address for *Here*, plug in the address for *There*, and hit ENTER. Google Maps then returns some routes to our destination as well as the driving distances for each route. So, Google Maps provides an existing engine for determining driving distances between two points. And by their very nature, driving routes must take geography and transportation infrastructure into account.

This paper presents a SAS macro program that invokes Google Maps to calculate driving distances between pairs of addresses. Start and End addresses are input via an Excel spreadsheet, so any practical number of driving distances can be computed in a single run of the program. SAS programmers can copy the macro program from **Appendix C** of this paper and begin using it right away.

INTRODUCTION

There are several SAS language constructs that can help you determine the distance between two points:

- The GEODIST function allows a programmer to determine the distance between two sets of Latitudes and Longitudes. If you do not have lattitudes and longitudes, you can calculate them with SAS:
    - PROC GEOCODE can be used to compute the Latitude and Longitude of a street address
- The ZIPCITYDISTANCE function can be run to find the distance between two Zip codes

Programs that calculate the straight-line distance between pairs of zip codes, lattitutes and longitudes, or street addresses generally do not account for geographic features and transportation infrastructure. But, geographic features such as mountains, bays, rivers, and lakes can have a significant influence on establishments people actually willing to access. Available transportation infrastructure such as highways, bridges, and surface roads can have a similar influence on the destinations people will really travel to. For example, consider Figure 1.

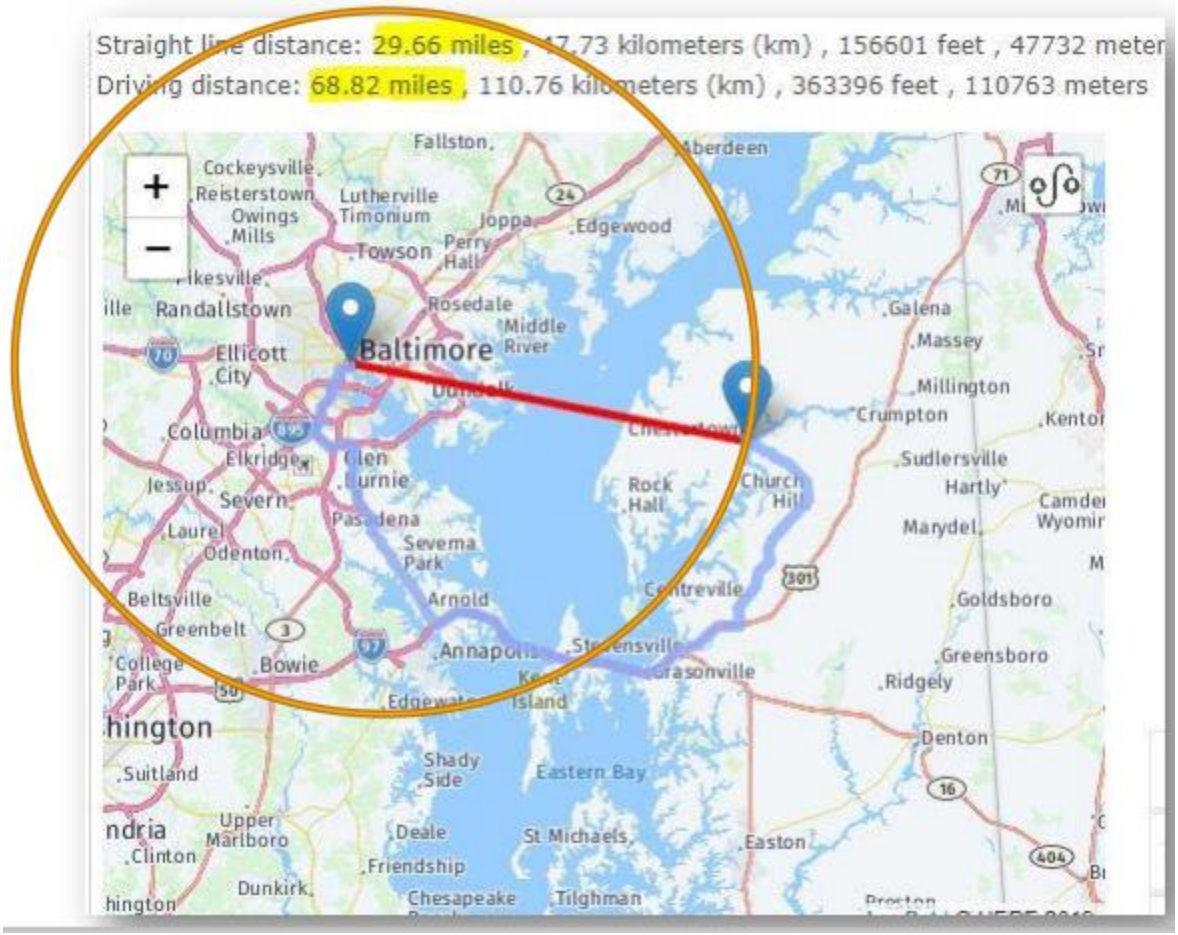


Figure 1 – Addresses within a 30-mile radius of Baltimore.

Computing all medical practices within a 30-mile radius of Baltimore, Maryland would net those located in Chestertown, which is 29.66 straight-line miles away. However, residents of Baltimore may not be willing to drive the actual 68.82 miles to get to such facilities. They are more likely to visit medical practices that fall within a more reasonable driving distance.

The Google Maps Driving Distance Search Macro program invokes the Google Maps API to capture the driving distance between two addresses. The macro can be used to compute the distance between multiple pairs of addresses. The output is written to an Excel file that analysts can then use to determine the optimal driving distances between two points.

The next section of this paper provides an overview of the Google Maps Driving Distance Search macro. Subsequent sections describe what the SAS code in each of the five segments of the macro program is doing. The paper then provides information on how you can operationalize the macro in your own organization. It ends with a summarization of the main points, and then several appendix. Note that **Appendix C** contains the entire SAS Data Set Characterization Macro program.

OVERVIEW OF THE GOOGLE MAPS DRIVING DISTANCE SEARCH MACRO

The Google Maps Driving Distance Search Macro requires two parameters:

- **EXCELFIL** - The full path to the input Excel file containing Start/End addresses
- **OUTDIR** - The directory where the output Excel results file will be written

The Excel input file specified by EXCELFIL should have the following eight columns:

- **STARTNAME** - The name of the originating person or establishment--e.g. Jim Smith
- **STARTADDRESS** - The street address of the originating person or establishment
- **STARTCITY** - The name of the originating city
- **STARTSTATE** - The originating state abbreviation--e.g. MD
- **ENDNAME** - The name of the destination person or establishment--e.g. City Medical Center
- **ENDADDRESS** - The street address of the destination person or establishment
- **ENDCITY** - The name of the destination city
- **ENDSTATE** - The destination state abbreviation--e.g. MD

The columns must be in the same order as above, have the column names as above, and each column must have a valid address value in it. **Appendix A**, *The Google Maps Driving Distance Search Macro Input Excel File*, has a screen capture of an example Excel input file.

The macro program produces an Excel report file with up to four tabs:

- **Shortest Drives** - The shortest driving distances between the Start/End addresses
- **Longer Drives** - The longer driving distances between the Start/End addresses
- **Bad Addresses** - Incomplete Start or End addresses that were in the input Excel file that would not work with Google Maps
- **Google Maps Couldn't Find** – Google Maps could not compute the distance for some reason. It could be that it could not find either the Start or End address or some other reason; such as attempting to drive from one Hawaiian island to another.

Appendix B, *Example of a Report Generated by the Google Maps Driving Distance Search Macro*, holds an example of the output of the macro.

The next five sections of this paper discuss the SAS code found in the five parts of the macro program.

PART 1 – INPUT THE EXCEL FILE AND WEED OUT ENTRIES THAT HAVE MISSING COLUMNS

This section begins by allocating the input Excel file with a LIBNAME statement using SAS/Access Interface to PC File Formats software. Next, a DATA _NULL_ step captures the name of the Excel file's worksheet and stores it in a macro variable named SHEETNAME for future use. Then, the name of the Excel file is stored in macro variable EXCELNAME. This is done because the output report file's name will include the actual name of the input Excel file. *See Figure 2.*

```
LIBNAME exfile pcfiles path="&EXCELFIL" access=readonly;


/* Get the Excel sheet name and store in a macro variable */
data _null_;
set  sashelp.vtable(where=(libname= "EXFILE"));

call symput("SHEETNAME",strip(memname));

run;


/* Obtain the name of the actual Excel file to use as part of the output report file name */
%LET EXCELNAME=%SCAN(&EXCELFIL,-2,.\);
```

Figure 2 – Part 1 – Screenshot 1 of 3.

Part 1 continues with a DATA step that first weeds out invalid addresses. Address pairs that are missing any one of the required eight columns are written to the BadAddresses SAS data set and the counter variable BadCount is incremented. *See Figure 3.*

```
/* Import the Excel file and put a "+" between spaces for future calls to Google Maps */
data AddressFile
     BadAddresses(drop=FromAddress ToAddress);

set  EXFILE."&SHEETNAME"n end=eof;

retain GoodCount BadCount 0;

drop GoodCount BadCount;

/* Check for invalid addresses that will not work with Google Maps */
if missing(StartAddress) or
   missing(StartCity)    or
   missing(StartState)   or
   missing(EndAddress)   or
   missing(EndCity)      or
   missing(EndState)     then do;

    BadCount + 1;
    output BadAddresses;

end;
    /* Only valid addresses pass this point*/
   else do;
```

Figure 3–Part 1 – Screenshot 2 of 3.

Address pairs with non-missing column values are transformed into two variables—FromAddress and ToAddress—by concatenating the relevant address, city, and state variables. During the concatenations, the TRANWRD function is used to have a "+" sign replace the blank spaces within the start/end address and city variables. The DATA step ends by creating the aptly-named GoodAddressCount and BadAddressCount macro variables. Those variables will be instrumental in future processing. *See Figure 4.*

```
    /* Only valid addresses pass this point*/
   else do;

        FromAddress = tranwrd(strip(StartAddress)," ","+")||","||
                      tranwrd(strip(StartCity)," ","+")||","||
                      strip(StartState);

        ToAddress = tranwrd(strip(EndAddress)," ","+")||","||
                    tranwrd(strip(EndCity)," ","+")||","||
                    strip(EndState);

        Goodcount + 1;

        output AddressFile;

    end;

/* Store the number of Good/Bad addresses in Macro variables for future use */
if eof then do;
    call symput("GoodAddressCount",GoodCount);
    call symput("BadAddressCount",BadCount);
end;
run;
```

Figure 4–Part 1 – Screenshot 3 of 3.

At this point, we potentially have two SAS data sets: AddressFile which contains entries from the Excel file that have values in the eight columns; and BadAddresses, which contains entries that were missing values in one or more of the eight columns. We will use the observations in AddressFile in our calls to Google Maps, while observations in BadAddresses will simply be written to the report file.

PART 2 - MAIN LOOP TO PROCESS EACH START/END (FROMADDRESS/TOADDRESS) ADDRESS PAIR WITH GOOGLE MAPS

This part is the main loop that processes each Start/End (FromAddress/ToAddress) address pair found in the AddressFile SAS data set with Google Maps. It is composed of two DATA steps with a FILENAME statement sandwiched between them. The macro loop is executed once for each unique Start/End address pair found in the AddressFile SAS data set.

This section starts with a DATA _NULL_ step that creates four macro variables: FromAddress, ToAddress, StartName, and ENDNAME. The first two will be used in the subsequent FILENAME statement which will invoke Google Maps. Note that the POINT=

option is used on the SET statement so that we can process each individual observervation of the AddressFile SAS data set in each subsequent iteration of the main macro loop that counts from 1 to %GoodAddressCount. *See Figure 5.*

```
/* BEGINNING of Loop to read each FromAddress/ToAddress pair and run them against Google Maps */
%DO I = 1 %TO &GoodAddressCount;

/* Get each subsequent FromAddress/ToAddress pair and load them into macro variables */
data _null_;

whichobs = &I;

set AddressFile point=whichobs;

call symput("FromAddress", strip(FromAddress));
call symput("ToAddress", strip(ToAddress));

call symput("StartName", StartName);
call symput("EndName", EndName);

stop;

run;


/* Allocate Google Maps with the To/From address pair */
filename google url "http://maps.google.com/maps?daddr=&ToAddress.%nrstr(&saddr)=&FromAddress";
```

Figure 5 – Part 2 – Screenshot 1 of 3

The second DATA step in Part 2 begins with some housekeeping. Lengths, labels, and formats are specified for the important variables. Next, an INFILE statement invokes the previouly declared FILENAME statement to use the URL method and execute Google Maps over the Internet. The subsequent IF statement limits incoming records to only those which contain the text string "miles."

Next, we find the exact position of the word "miles"; and then subtract one from that to get the space before the word "miles." Armed with that information, we scan the substring to get the number that is right before the word "miles." That number is the distance between the two addresses that was returned by Google Maps. *See Figure 6*.

IMPORTANT NOTE: This DATA step can take up to 30-seconds to execute per invocation of Google Maps for a given Start/End address pair over the Internet. So, expect expanded program execution times for runs where you are computing driving distances for large numbers of addresses.

```
/* Execute the Google Maps FILENAME to get the distance between Start/End addresses */
data distance(drop=BigLine BeforeMiles);

length StartAddress EndAddress StartName EndName $100.;

label StartName       = "Start Name"
      StartAddress    = "Start Address"
      EndName         = "End Name"
      EndAddress      = "End Address"
      DrivingDistance = "Driving Distance"
      ;

format DrivingDistance comma12.;

* Invoke the Google Maps URL and read the resulting html input strings;
infile google recfm=f lrecl=10000;
input BigLine $10000.;

* Search html strings for the word miles, keep only those entries;
if find(BigLine,'miles');

* Set pointer variable to column just before the word 'miles';
BeforeMiles= find(BigLine,'miles') - 1;

* Scan the substring and keep the numeric value found right before 'miles';
DrivingDistance=input(compress(scan(substr(BigLine,1,BeforeMiles),1,' "','bc'),','),best12.);

* Create the other variables for the output file;
StartAddress = translate("&FromAddress"," ","+");
EndAddress   = translate("&ToAddress"," ","+");
StartName    = strip("&StartName");
EndName      = strip("&EndName");

run;
```

Figure 6 – Part 2 – Screenshot 2 of 3

Now it is time to determine whether Google Maps was able to find the addresses and consequently determine the distance between them. This is done by checking whether we successfully created one or more records in the DISTANCE SAS data set. We check SASHELP.VTABLE and set the macro variable AnyGoogleHits equal to the number of distance records now stored as observations. If AnyGoogleHits is greater than zero, we concatenate the distance observations in DISTANCE to the AllDistances SAS data set. If AnyGoogleHits is *not* greater than zero, Google did not return a distance. In that case, we construct the two address and two name variables, and then append an observation to the NoGoogleHits SAS data set. *See Figure 7*.

```
/* Determine if the Google Map search got results */
data _null_;
set  sashelp.vtable(where=(libname="WORK" and memname="DISTANCE"));

    call symput("AnyGoogleHits",nobs);

run;

/* Addresses that Google Maps finds are processed here */
%IF &AnyGoogleHits > 0 %then %do;

    * Append all distances to file for future reporting;
    proc append base=AllDistances data=distance;
    run;

%END;
    /* Addresses that Google Maps could not find are processed here */
    %ELSE %DO;

        data nohits;

            length StartAddress EndAddress StartName EndName $100.;

            label StartName       = "Start Name"
                  StartAddress    = "Start Address"
                  EndName         = "End Name"
                  EndAddress      = "End Address"
                  ;

            StartAddress = translate("&FromAddress"," ","+");
            EndAddress   = translate("&ToAddress"," ","+");
            StartName    = strip("&StartName");
            EndName      = strip("&EndName");

        run;

        * Append addresses that did not get a hit on Google Maps to file for future reporting;
        proc append base=NoGoogleHits data=nohits;
        run;

    %END;
/* END of Loop to read each Start/End address pair and run them against Google Maps */
%END;
```

Figure 7 – Part 2 – Screenshot 3 of 3

We are now done with Part 2 of the program, and can go onto Part 3 which puts the observations in the AllDistances SAS data set in reporting order.

PART 3 – GET THE OBSERVATIONS IN THE RIGHT ORDER AND SPLIT THEM INTO TWO DATA SETS FOR REPORTS

Part 3 is only executed if Google Maps returned actionable records—records that are now stored in AllDistances. We first sort AllDistances by five variables (StartName, StartAddress, EndName, EndAddress, and DrivingDistance) to sort the driving distances from low to high for each name/address pair. The variable DrivingDistance is not in the BY statement of the second sort. Consequently, the first (shortest distance) observation for each unique name/address pair is written to ShortDistances; while subsequent "duplicate" records for

the given name/address pair are written to the LongerDistances SAS data set. Thus, at the end of this section, we have segregated the shortest distances into data set ShortDistances, and the longer distances in data set LongerDistances. *See Figure 8*.

```
/*********************************************************************************************/
/* Part 3                                                                                    */
/*********************************************************************************************/
/* Get the observations in the right order and split them into two data sets for reports*/
/*********************************************************************************************/

/* Only execute if there were successful Google Searches */
%IF &AnyGoogleHits > 0 %THEN %DO;

/* Keep observation with the shortest distance on "top" */
proc sort data=AllDistances;
    by StartName StartAddress EndName EndAddress DrivingDistance;
run;

/* Separate short distances from longer distances */
proc sort nodupkey data=AllDistances out=ShortDistances dupout=LongerDistances;
    by StartName StartAddress EndName EndAddress ;
run;

%END;
```

Figure 8 – Part 3 – Screenshot 1 of 1

PART 4 - CREATE THE EXCEL REPORT FILE WITH VARIOUS TABS

Part 4 starts by allocating the Excel report file in the directory specified by the &OUTDIR macro parameter. Note that the report file's name will include the name of the input Excel file. This is done to keep report files distinct in cases where the macro program is run against multiple Excel input files.

If the AnyGoogleHits macro variable is greater than zero, we create a worksheet named "Shortest Drives" and PROC PRINT the observations from ShortDistances into the worksheet. We also create a worksheet named "Longer Drives" and PROC PRINT the observations from LongerDistances into that worksheet. *See Figure 9*.

```
/*************************************************************************************************/
/* Part 4                                                                                        */
/*************************************************************************************************/
/* Create the Excel report file with various tabs.                                               */
/*************************************************************************************************/

/* Create the Excel Report File */

ODS EXCEL FILE="&OUTDIR.\Driving Distances Report for &EXCELNAME..xlsx";

/* Only execute if there were successful Google Searches */
%IF &AnyGoogleHits > 0 %THEN %DO;

        /* Create the tab with shortest drive distances between the addresses */
        ODS EXCEL options(sheet_name="Shortest Drives");

        proc print noobs data=ShortDistances label;
        var StartName StartAddress EndName EndAddress DrivingDistance;
        title1 "Short Driving Distances Computed by Google Maps";
        run;

        /* Create the tab with other, longer drive distances between the addresses */
        ODS EXCEL options(sheet_name="Longer Drives");

        proc print noobs data=LongerDistances label;
        var StartName StartAddress EndName EndAddress DrivingDistance;
        title1 "Longer Driving Distances Computed by Google Maps";
        run;

%END;
```

Figure 9 – Part 4 – Screenshot 1 of 2

We next turn our attention to any bad addresses that may have been entered in the input spreadsheet—the ones weeded out in Part 1. If the &BadAddressCount macro variable is greater than zero, then we have Addresses in the BadAddress SAS data set. When that is the case, we create a worksheet named "Bad Addresses" and write the observations in the BadAddress SAS data set to the worksheet.

If Google Maps couldn't determine a particular distance, we will have a NoGoogleHits SAS data set. So, we check for the existance of said data set. If it exists, we create a worksheet named "Google Maps Couldn't Find It" and dump the observations into that worksheet.

The final act of Part 4 is to close the Excel worksheet. *See Figure 10*.

```
/* This tab has any bad addresses found in the input Excel file */
%IF &BadAddressCount > 0 %THEN %DO;

    ODS EXCEL options(sheet_name="Bad Addresses");

    proc print noobs data=BadAddresses label;
    title1 "Problematic Addresses that Won't Work with Google Maps";
    run;

%END;

/* This tab has any addresses that did not return a hit using Google Maps */
%IF %SYSFUNC(EXIST(work.NoGoogleHits)) %THEN %DO;

    ODS EXCEL options(sheet_name="Google Maps Couldn't Find");

    proc print noobs data=NoGoogleHits label;
    title1 "Addresses Google Maps Could Not Find";
    run;

%END;

ODS EXCEL close;
```

Figure 10 – Part 4 – Screenshot 2 of 2

Now, we can move on to Part 5.

PART 5 – CLEAN UP WORK LIBRARY FOR POSSIBLE MULTIPLE SUBMISSIONS OF THIS MACRO

This part of the program performs some housekeeping that is only really necessary if we are going to run the macro against multiple input Excel files. It uses the KILL option of PROC DATASETS to clear out all files in the WORK library. Then, it clears all LIBNAMES. *See Figure 11*.

```
/*************************************************************************************************/
/* Part 5                                                                                        */
/*************************************************************************************************/
/* Clean up WORK library for possible multiple submissions of this macro. Clear all              */
/* LIBNAMEs.                                                                                     */
/*************************************************************************************************/

proc datasets library=work kill noprint;
run;
quit;

libname _all_ clear;

%MEND GETDIST;
```

Figure 11 – Part 5 – Screenshot 1 of 1

OPERATIONALIZING THE GOOGLE MAPS DRIVING DISTANCE SEARCH MACRO

Here are some simple steps you can consider implementing to operationalize the Google Maps Driving Distance Search Macro SAS program in your own environment.

1. Copy the macro from Appendix C into a SAS program on your computer. You can save it in your autocall macro library or save it in a directory of your choosing. If you put the macro in your autocall library, you can code the macro call directly in your programs.

2. If you do not choose to put the program in one of your organization's macro libraries, take a look at Appendix D, which contains the Execute Google Maps Driving Distance Search Macro SAS program. This is a driver program for the Google Maps Driving Distance Search Macro SAS program. Simply copy the SAS code from Appendix D to a SAS program in your environment. Then, update the %INCLUDE statement in the driver program to specify the full path to where you placed the utility macro.

3. Once you have "downloaded" the macro program and the driver program, simply specify the two parameters in the macro call:

   - **EXCELFIL** - The full path to the input Excel file containing Start/End addresses
   - **OUTDIR** - The directory where the output Excel results file will be written.

   …and execute the macro to create your consolidated SAS data set report. Note that you should not put quotes around the value of either parameter; the program will do that work for you.

Either of these two methods—macro autocall library, or %INCLUDE—should make the Google Maps Driving Distance Search Macro convenient to use.

CONCLUSION

This paper introduced the *Google Maps Driving Distance Search Macro* program; which can be used to determine the distance between a set of Start/End addresses. The program requires that you specify an Excel file holding the Start/End address pairs and specify the

directory where you want the report file to be written. The program creates an Excel output file with up to four reports: Shortest Drives, Longer Drives, Bad Addresses, and Google Maps Couldn't Find.

The *Google Maps Driving Distance Search Macro* program is ideal for creating a report of driving distances between pairs of addresses. Consider how this program can be of help when you need to determine practical driving distances for a project in your own organization.

REFERENCES

Raithel, Michael A. 2017. Did You Know That? Essential Hacks for Clever SAS Programmers: Over 100 Essential Hacks to Make Your Programs Leaner, Cleaner, and More Competitive. Bethesda, Maryland: Michael A. Raithel
Available: http://www.amazon.com/Michael-A.-Raithel/e/B001K8GG90/ref=ntt_dp_epwbk_0

Smith, Laurie (2018). Macro Method to use Google Maps™ and SAS® to Geocode a Location by Name or Address.
Proceedings of the SAS Global Forum 2018 Conference.
Available: https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2684-2018.pdf

Bekkerman, Anton (2013). Going the Distance: Google Maps Capabilities in a Friendly SAS Environment. Proceedings of the Western Users of SAS Software 2013 Conference.
Available: https://www.lexjansen.com/wuss/2013/100_Paper.pdf

Zdeb, Mike (2010). Driving Distances and Times Using SAS® and Google Maps.
Proceedings of the SAS Global Forum 2010 Conference.
Available: http://support.sas.com/resources/papers/proceedings10/050-2010.pdf

ACKNOWLEDGMENTS

Thanks and recognition go to Laurie Smith, Anton Bekkerman, and Mike Zdeb for their solid pioneering work on harnessing the power of Google Maps with SAS. Thanks are also due to Westat Senior Systems Analyst Raghav Adimulam for helping the author develop and test the macro. Finally, the author would like to thank Westat management for supporting his participation in SAS user group meetings and conferences.

RECOMMENDED READING

- The GEODIST Function: SAS Help Center: GEODIST Function
- PROC GEOCODE: SAS Help Center: Overview: PROC GEOCODE
- The ZIPCITYDISTANCE Function: SAS Help Center: ZIPCITYDISTANCE Function

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

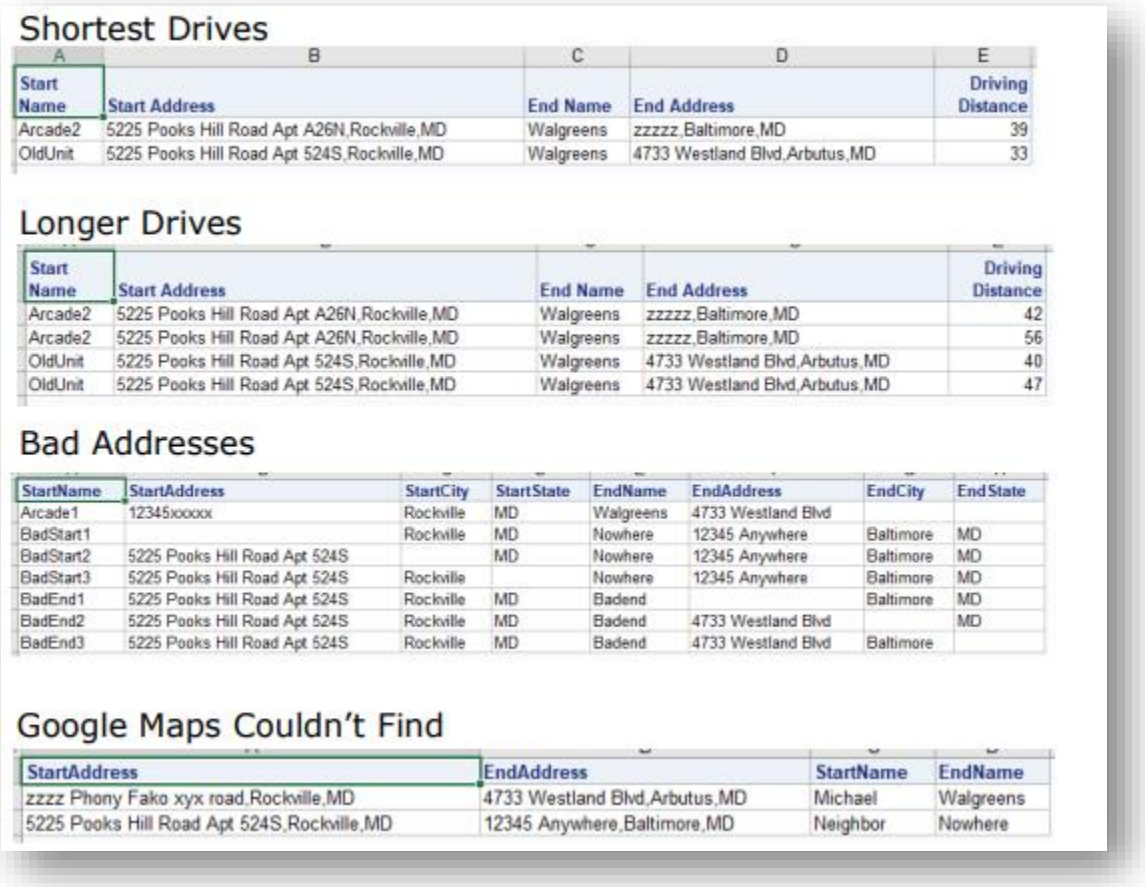
Michael A. Raithel
michaelraithel@westat.com

Other brand and product names are trademarks of their respective companies.

APPENDIX A – THE GOOGLE MAPS DRIVING DISTANCE SEARCH MACRO EXCEL INPUT FILE

StartName	StartAddress	StartCity	StartState	EndName	EndAddress	EndCity	EndState

APPENDIX B – EXAMPLE OF A REPORT FILE GENERATED BY THE GOOGLE MAPS DRIVING DISTANCE SEARCH MACRO



Shortest Drives

Start Name	Start Address	End Name	End Address	Driving Distance
Arcade2	5225 Pooks Hill Road Apt A26N,Rockville,MD	Walgreens	zzzzz,Baltimore,MD	39
OldUnit	5225 Pooks Hill Road Apt 524S,Rockville,MD	Walgreens	4733 Westland Blvd,Arbutus,MD	33

Longer Drives

Start Name	Start Address	End Name	End Address	Driving Distance
Arcade2	5225 Pooks Hill Road Apt A26N,Rockville,MD	Walgreens	zzzzz,Baltimore,MD	42
Arcade2	5225 Pooks Hill Road Apt A26N,Rockville,MD	Walgreens	zzzzz,Baltimore,MD	56
OldUnit	5225 Pooks Hill Road Apt 524S,Rockville,MD	Walgreens	4733 Westland Blvd,Arbutus,MD	40
OldUnit	5225 Pooks Hill Road Apt 524S,Rockville,MD	Walgreens	4733 Westland Blvd,Arbutus,MD	47

Bad Addresses

StartName	StartAddress	StartCity	StartState	EndName	EndAddress	EndCity	EndState
Arcade1	12345xxxxx	Rockville	MD	Walgreens	4733 Westland Blvd		
BadStart1		Rockville	MD	Nowhere	12345 Anywhere	Baltimore	MD
BadStart2	5225 Pooks Hill Road Apt 524S		MD	Nowhere	12345 Anywhere	Baltimore	MD
BadStart3	5225 Pooks Hill Road Apt 524S	Rockville		Nowhere	12345 Anywhere	Baltimore	MD
BadEnd1	5225 Pooks Hill Road Apt 524S	Rockville	MD	Badend		Baltimore	MD
BadEnd2	5225 Pooks Hill Road Apt 524S	Rockville	MD	Badend	4733 Westland Blvd		MD
BadEnd3	5225 Pooks Hill Road Apt 524S	Rockville	MD	Badend	4733 Westland Blvd	Baltimore	

Google Maps Couldn't Find

StartAddress	EndAddress	StartName	EndName
zzzz Phony Fako xyx road,Rockville,MD	4733 Westland Blvd,Arbutus,MD	Michael	Walgreens
5225 Pooks Hill Road Apt 524S,Rockville,MD	12345 Anywhere,Baltimore,MD	Neighbor	Nowhere

APPENDIX C – THE GOOGLE MAPS DRIVING DISTANCE SEARCH MACRO.SAS

```
/*************************************************************************************************/
/*  Program: Google Maps Driving Distance Search Macro.sas                                       */
/*                                                                                               */
/*  Author: Michael A. Raithel and Raghav Adimulam                                               */
/*                                                                                               */
/*  Created: 12/07/2021                                                                          */
/*                                                                                               */
/*  Purpose: This SAS program reads an Excel file of Start/End street address pairs and uses Google   */
/*           Maps to find the driving distance between them.                                     */
/*                                                                                               */
/*  The Excel spreadsheet should have the following columns:                                     */
/*                                                                                               */
/*             StartName    - The name of the originating person or establishment--e.g. Jim Smith     */
/*             StartAddress - The street address of the originating person or establishment           */
/*             StartCity    - The name of the originating city                                     */
/*             StartState   - The originating state abbreviation--e.g. MD                           */
/*                                                                                               */
/*             EndName    - The name of the destination person or establishment--e.g. City Med Center */
/*             EndAddress - The street address of the destination person or establishment          */
/*             EndCity    - The name of the destination city                                     */
/*             EndState   - The destination state abbreviation--e.g. MD                          */
/*                                                                                               */
/*  Parameters: Users must specify the following two parameters:                                 */
/*                                                                                               */
/*             EXCELFIL - The full path to the input Excel file containg Start/End addresses     */
/*             OUTDIR   - The directory where the output Excel results file will be written      */
/*                                                                                               */
/*             NOTE: Do NOT enclose either of the two parameter inputs in quotes when invoking   */
/*                   the macro                                                                   */
/*                                                                                               */
/*  Outputs: This program creates an Excel spreadsheet named "Driving Distances Report for <<Name of  */
/*           Input Excel Spreadsheet>>".xlsx that can have up to four tabs:                      */
/*                                                                                               */
/*           Shortest Drives - The shortest driving distances between the Start/End addresses    */
/*           Longer Drives   - The longer driving distances between the Start/End addresses      */
/*           Bad Addresses   - Problematic Start or End Address that won't work with Google Maps */
/*           Google Maps Couldn't Find - Google Maps couldn't find either the Start or End address  */
/*                                                                                               */
/*           NOTE: Google Maps often returns 2 or 3 distances; that is why successful searches are  */
/*                 segmented into the Shortest Drives and the Longer Drives tabs.                */
/*                                                                                               */
/*  Change Log:                                                                                  */
/*                                                                                               */
/*************************************************************************************************/

%MACRO GETDIST(EXCELFIL=,OUTDIR=);

/*********************************************************************************/
/* Part 1                                                                        */
/*********************************************************************************/
/* Input the Excel file and weed out entries that have missing columns.          */
/*********************************************************************************/

LIBNAME exfile pcfiles path="&EXCELFIL"    access=readonly;


/* Get the Excel sheet name and store in a macro variable */
data _null_;
set  sashelp.vtable(where=(libname= "EXFILE"));

call symput("SHEETNAME",strip(memname));

run;


/* Obtain the name of the input Excel file to use as part of the output Excel report file name */
%LET EXCELNAME=%SCAN(&EXCELFIL,-2,.\);


/* Import the Excel file and put a "+" between spaces for future calls to Google Maps */
data AddressFile
          BadAddresses(drop=FromAddress ToAddress);

set  EXFILE."&SHEETNAME"n end=eof;

retain GoodCount BadCount 0;

drop GoodCount BadCount;
```

```
/* Check for invalid addresses that will not work with Google Maps */
if missing(StartAddress) or
   missing(StartCity)    or
   missing(StartState)   or
   missing(EndAddress)   or
   missing(EndCity)      or
   missing(EndState)     then do;

         BadCount + 1;
         output BadAddresses;

end;
         /* Only valid addresses pass this point*/
   else do;

                  FromAddress = tranwrd(strip(StartAddress)," ","+")||","||
                                    tranwrd(strip(StartCity)," ","+")||","||
                                    strip(StartState);

                  ToAddress = tranwrd(strip(EndAddress)," ","+")||","||
                                   tranwrd(strip(EndCity)," ","+")||","||
                                   strip(EndState);

                  Goodcount + 1;

                  output AddressFile;

         end;

/* Store the number of Good/Bad addresses in Macro variables for future use */
if eof then do;
         call symput("GoodAddressCount",GoodCount);
         call symput("BadAddressCount",BadCount);
end;
run;


/*********************************************************************************/
/* Part 2                                                                        */
/*********************************************************************************/
/* Main loop to process each FromAddress/ToAddress address pair with Google Maps. */
/*********************************************************************************/

/* BEGINNING of Loop to read each FromAddress/ToAddress pair and run them against Google Maps */
%DO I = 1 %TO &GoodAddressCount;

/* Get each subsequent FromAddress/ToAddress pair and load them into macro variables */
data _null_;

whichobs = &I;

set AddressFile point=whichobs;

call symput("FromAddress", strip(FromAddress));
call symput("ToAddress", strip(ToAddress));

call symput("StartName", StartName);
call symput("EndName", EndName);

stop;

run;


/* Allocate Google Maps with the To/From address pair */
filename google url "http://maps.google.com/maps?daddr=&ToAddress.%nrstr(&saddr)=&FromAddress";


/* Execute the Google Maps FILENAME to get the distance between Start/End addresses */
data distance(drop=BigLine BeforeMiles);

length StartAddress EndAddress StartName EndName $100.;

label StartName       = "Start Name"
          StartAddress    = "Start Address"
          EndName         = "End Name"
          EndAddress      = "End Address"
          DrivingDistance = "Driving Distance"
          ;

format DrivingDistance comma12.;
```

```
* Invoke the Google Maps URL and read the resulting html input strings;
infile google recfm=f lrecl=10000;
input BigLine $10000.;

* Search html strings for the word miles, keep only those entries;
if find(BigLine,'miles');

* Set pointer variable to column just before the word 'miles';
BeforeMiles= find(BigLine,'miles') - 1;

* Scan the substring and keep the numeric value found right before 'miles';
DrivingDistance=input(compress(scan(substr(BigLine,1,BeforeMiles),1,' "','bc'),','),best12.);

* Create the other variables for the output file;
StartAddress = translate("&FromAddress"," ","+");
EndAddress   = translate("&ToAddress"," ","+");
StartName    = strip("&StartName");
EndName      = strip("&EndName");

run;

/* Determine if the Google Map search got results */
data _null_;
set  sashelp.vtable(where=(libname="WORK" and memname="DISTANCE"));

        call symput("AnyGoogleHits",nobs);

run;

/* Addresses that Google Maps finds are processed here */
%IF &AnyGoogleHits > 0 %then %do;

        * Append all distances to file for future reporting;
        proc append base=AllDistances data=distance;
        run;

%END;
        /* Addresses that Google Maps could not find are processed here */
        %ELSE %DO;

                data nohits;

                        length StartAddress EndAddress StartName EndName $100.;

                        label StartName       = "Start Name"
                                  StartAddress    = "Start Address"
                                  EndName         = "End Name"
                                  EndAddress      = "End Address"
                                  ;

                        StartAddress = translate("&FromAddress"," ","+");
                        EndAddress   = translate("&ToAddress"," ","+");
                        StartName    = strip("&StartName");
                        EndName      = strip("&EndName");

                run;

                * Append addresses that did not get a hit on Google Maps to file for future reporting;
                proc append base=NoGoogleHits data=nohits;
                run;

        %END;
/* END of Loop to read each Start/End address pair and run them against Google Maps */
%END;


/*************************************************************************************/
/* Part 3                                                                            */
/*************************************************************************************/
/* Get the observations in the right order and split them into two data sets for reports*/
/*************************************************************************************/

/* Only execute if there were successful Google Searches */
%IF &AnyGoogleHits > 0 %THEN %DO;

        /* Keep observation with the shortest distance on "top" */
        proc sort data=AllDistances;
                by StartName StartAddress EndName EndAddress DrivingDistance;
        run;

        /* Separate short distances from longer distances */
        proc sort nodupkey data=AllDistances out=ShortDistances dupout=LongerDistances;
                by StartName StartAddress EndName EndAddress ;
```

```
        run;

%END;

/*****************************************************************************************/
/* Part 4                                                                                */
/*****************************************************************************************/
/* Create the Excel report file with various tabs.                                       */
/*****************************************************************************************/

/* Create the Excel Report File */

ODS EXCEL FILE="&OUTDIR.\Driving Distances Report for &EXCELNAME..xlsx";

/* Only execute if there were successful Google Searches */
%IF &AnyGoogleHits > 0 %THEN %DO;

        /* Create the tab with shortest drive distances between the addresses */
        ODS EXCEL options(sheet_name="Shortest Drives");

        proc print noobs data=ShortDistances label;
        var StartName StartAddress EndName EndAddress DrivingDistance;
        title1 "Short Driving Distances Computed by Google Maps";
        run;

        /* Create the tab with other, longer drive distances between the addresses */
        ODS EXCEL options(sheet_name="Longer Drives");

        proc print noobs data=LongerDistances label;
        var StartName StartAddress EndName EndAddress DrivingDistance;
        title1 "Longer Driving Distances Computed by Google Maps";
                run;

%END;

/* This tab has any bad addresses found in the input Excel file */
%IF &BadAddressCount > 0 %THEN %DO;

        ODS EXCEL options(sheet_name="Bad Addresses");

        proc print noobs data=BadAddresses label;
        title1 "Problematic Addresses that Won't Work with Google Maps";
        run;

%END;

/* This tab has any addresses that did not return a hit using Google Maps */
%IF %SYSFUNC(EXIST(work.NoGoogleHits)) %THEN %DO;

        ODS EXCEL options(sheet_name="Google Maps Couldn't Find");

        proc print noobs data=NoGoogleHits label;
        title1 "Addresses Google Maps Could Not Find";
        run;

%END;

ODS EXCEL close;

/*****************************************************************************************/
/* Part 5                                                                                */
/*****************************************************************************************/
/* Clean up WORK library for possible multiple submissions of this macro. Clear all      */
/* LIBNAMEs.                                                                             */
/*****************************************************************************************/

proc datasets library=work kill noprint;
run;
quit;

libname _all_ clear;

%MEND GETDIST;
```

APPENDIX D – EXECUTE THE GOOGLE MAPS DRIVING DISTANCE SEARCH MACRO.SAS

```
/*************************************************************************************************/
/*  Program: Execute Google Maps Driving Distance Search Macro.sas                               */
/*                                                                                               */
/*  Author: Michael A. Raithel and Raghav Adimulam                                               */
/*                                                                                               */
/*  Created: 12/07/2021                                                                          */
/*                                                                                               */
/*  Purpose: This is the driver program for the Google Maps Driving Distance Search Macro program that*/
/*           reads an Excel file of Start/End street address pairs and uses Google Maps to find the   */
/*           driving distance between them.                                                      */
/*                                                                                               */
/*  The Excel spreadsheet should have the following columns:                                     */
/*                                                                                               */
/*             STARTNAME    - The name of the originating person or establishment--e.g. Jim Smith     */
/*             STARTADDRESS - The street address of the originating person or establishment           */
/*             STARTCITY    - The name of the originating city                                    */
/*             STARTSTATE   - The originating state abbreviation--e.g. MD                         */
/*                                                                                               */
/*             ENDNAME    - The name of the destination person or establishment--e.g. City Med Center */
/*             ENDADDRESS - The street address of the destination person or establishment         */
/*             ENDCITY    - The name of the destination city                                     */
/*             ENDSTATE   - The destination state abbreviation--e.g. MD                          */
/*                                                                                               */
/*  Parameters: Users must specify the following two parameters in this driver program:          */
/*                                                                                               */
/*             EXCELFIL - The full path to the input Excel file                                  */
/*             OUTDIR   - The directory where the output Excel results file will be written      */
/*                                                                                               */
/*             NOTE: Do NOT enclose either of the two parameter inputs in quotes when invoking   */
/*                   the macro.                                                                  */
/*                                                                                               */
/*  Outputs: This program creates an Excel spreadsheet named "Driving Distances Report for <<Name of  */
/*           Input Excel Spreadsheet>>".xlsx that can have up to four tabs:                      */
/*                                                                                               */
/*           Shortest Drives - The shortest drive distances between the Start/End addresses      */
/*           Longer Drives   - The longer drive distances between the Start/End addresses        */
/*           Bad Addresses   - Problematic Start or End Address that won't work with Google Maps   */
/*           Google Maps Couldn't Find - Google Maps couldn't find either the Start or End address  */
/*                                                                                               */
/*           NOTE: Google Maps often returns 2 or 3 distances; that is why successful searches are  */
/*                 segmented into the Shortest Drives and the Longer Drives tabs.                */
/*                                                                                               */
/*  Change Log:                                                                                  */
/*                                                                                               */
/*************************************************************************************************/

options symbolgen mprint mlogic;

/*********************************/
/* Include the macro program.    */
/*********************************/

%INCLUDE "<Put your directory address here>\Google Maps Driving Distance Search Macro.sas";

/*****************************************/
/* Execute the macro program.            */
/*****************************************/
/* Do NOT put quotes around either input.*/
/*****************************************/
%GETDIST(EXCELFIL=,
         OUTDIR=);
```