SESUG Paper 141-2023

So Close Yet How Far Away – Closest City Macro

Brooke Ellen Delgoffe, Marshfield Clinic Health System

ABSTRACT

Assuming many businesses have multiple locations, the task of finding which one is closest to your customer is a common one. Maybe you are trying to approximate minimum traveling distance for reimbursement, invitation lists to a given event, or recommending a list of locations by distance; the applications are plentiful.

This paper will introduce resources, such as SASHELP.ZIPCODE and ZIPCITYDISTANCE() or GEODIST(), for finding the zip code distance between two cities, a list of cities, or specific zip codes. It will cover concepts for calculating distance and present a macro with different options for calculation. Subtopics will include PROC SQL for multiple distance considerations and pitfalls for each presented method. References to existing publications and code base will facilitate readers in understanding options beyond what is covered in this paper.

INTRODUCTION

The concept of ZIP codes originated in 1963 as a way to improve the speed of mail delivery by the US Postal System (Terrall, 2013), but their uses today go far beyond the post office. Since zip codes are no longer breaking news in the SAS programming world either many existing papers go over much of the same concepts. To the extent possible, we provide these resources.

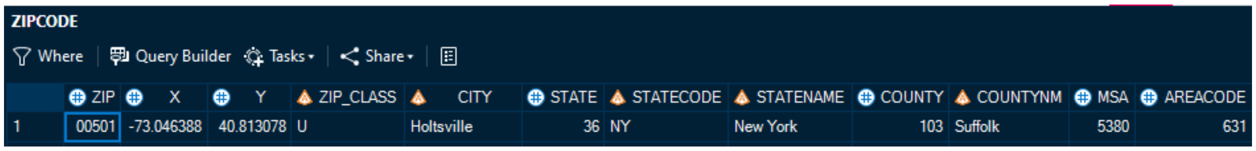
Information covered in this paper will seek to inform you on methods for obtaining information about zip codes using resources provided by SAS and potential uses for the resulting data. No prior knowledge of these resources is assumed; details cover not only what the resources are, but also where to find them and how they can be used.

The macro %FIND_CLOSESTCITY will be used to demonstrate utility of several of the resources covered. It is provided in full executable form in the appendix. Full code is provided and will not be specifically broken down/explained in the body of the paper, but background on the purpose it was designed for is given.

WHERE TO FIND INFORMATION ABOUT ZIPCODES

SAS provides an amazingly helpful dataset in the SASHELP library called ZIPCODES (subsequently referred to as SASHELP.ZIPCODES). In that dataset there is information about each zip code, including coordinates, area codes, cities, states, and aliases. Some of the information, like state in Data Preview 1,is presented in multiple forms (STATE, STATECODE, and STATENAME) to assist with matching or reporting of this data in standard form.

In this same dataset, it will also give you the "centroid" of that zip code using X and Y coordinates that align with latitude (Y) and longitude(X).



	ZIP	X	Y	ZIP_CLASS	CITY	STATE	STATECODE	STATENAME	COUNTY	COUNTYNM	MSA	AREACODE
1	00501	-73.046388	40.813078	U	Holtsville	36	NY	New York	103	Suffolk	5380	631

Data Preview 1. SASHELP.ZIPCODES

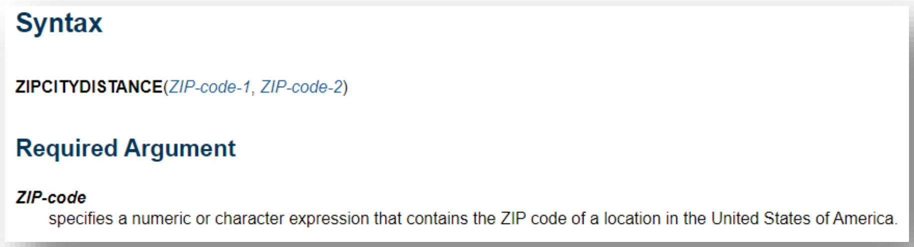
Fickbohm summarized the contents and gave instructions on how to obtain those datasets (Fickbohm, 2007). Hadden and Zdeb expanded on the different variables available in this dataset in their ZIP Code 411 papers (Hadden & Zdeb, 2006; 2010) and provided a link to the SAS Maps and Geocoding (*See* Recommended Reading) that is still valid today. As previously noted, the contents of the dataset are regularly updated but the variables have remained the same.

FUNCTIONS FOR CALCULATING DISTANCE

Both functions for calculating distance do so via geodetic distance and therefore allows calculating the same results via the Haversine formula, as was the case prior to these functions (Hadden & Zdeb, 2010).

ZIP CODE DISTANCE

When it comes to determining the distance between two zip codes there is a very helpful function called ZIPCITYDISTANCE(). This function calculates the distance (in miles) between two zip code centroids. The only required arguments are the two zip codes in question (See Figure 1 below).



Syntax

ZIPCITYDISTANCE(*ZIP-code-1*, *ZIP-code-2*)

Required Argument

ZIP-code
specifies a numeric or character expression that contains the ZIP code of a location in the United States of America.

Figure 1. ZIPCITYDISTANCE() Syntax and Required Arguments from SAS Documentation (SAS Institute Inc., 2016)

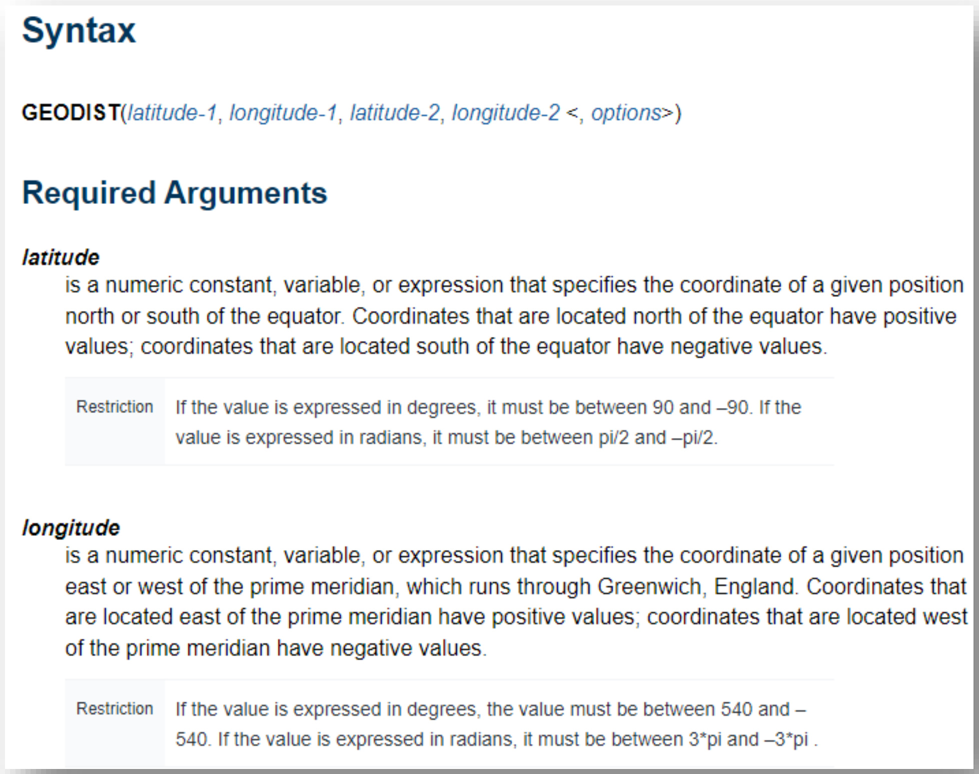
While reviewing the documentation you will soon discover that this function also uses the SASHELP.ZIPCODES dataset. Therefore, the zip codes you provide the function must also be in the dataset. Otherwise you will see the following note:

```
NOTE: Invalid argument to function ZIPCITYDISTANCE(<ZIP CODE>,<INVALID
ZIP CODE>) at line [location].
```

You will also notice a missing value for the associated variable. Since this is a NOTE and not an ERROR, these can sometimes be easy to miss. Adding a simple check of the distinct zip code values presented against the SASHELP.ZIPCODES dataset should help to avoid inadvertent missing data.

COORDINATE DISTANCE

The complement of the ZIPCITYDISTANCE() function for using coordinates as inputs instead of zip codes is function GEODIST(). GEODIST() calculates the distance (in miles by default) between two sets of coordinates. It also has options for changing the units (kilometers, degrees, radians). The required arguments are presented in Figure 2 below.



Syntax

GEODIST(*latitude-1*, *longitude-1*, *latitude-2*, *longitude-2* <, *options*>)

Required Arguments

latitude
is a numeric constant, variable, or expression that specifies the coordinate of a given position north or south of the equator. Coordinates that are located north of the equator have positive values; coordinates that are located south of the equator have negative values.

Restriction	If the value is expressed in degrees, it must be between 90 and –90. If the value is expressed in radians, it must be between pi/2 and –pi/2.

longitude
is a numeric constant, variable, or expression that specifies the coordinate of a given position east or west of the prime meridian, which runs through Greenwich, England. Coordinates that are located east of the prime meridian have positive values; coordinates that are located west of the prime meridian have negative values.

Restriction	If the value is expressed in degrees, the value must be between 540 and –540. If the value is expressed in radians, it must be between 3*pi and –3*pi .

Figure 2. GEODIST() Syntax and Required Arguments from SAS Documentation (SAS Institute Inc., 2016)

POTENTIAL USE CASES AND METHODS OF APPLICATION

After locating the zip code information and understanding the functions, you'll then be left to wonder if they can help you achieve your goals. If your goal is one of these, then the answer is yes.

CLEANING

Address information can be very dirty data. Just as the USPS determined zip codes would help reduce the amount of sorting, so can you! (Terrall, 2013) The zip code itself can tell you some information about the location without considering the other parts of the address:

> "the five-digit ZIP number is a structured code in which the first digit identifies one of ten large areas of the Nation, and the second digit indicates a State, a geographic portion of a heavily populated State, or two or more less populated States. The third digit identifies a major destination area within a State, which may be a large city post office or a major mail concentration point (Sectional Center) in a less populated area. Five hundred fifty-three of these Sectional Centers have been designated across the country. The final two digits indicate either a postal delivery unit of a larger city post office, or an individual post office served from a Sectional Center." (U.S. Postal Service, 1963)

As such, the zip code can be used to narrow in on the valid addresses parts applicable to that zip code. Consider the example of zip code is 54449 (Table 1) and city is Marshfield (Table 2), transposed for visibility below. In Table 3, the SPEDIS() function is used to evaluate spelling distance between

Marshfield and other city names; this identifies other valid city names that could be mistaken as misspellings or the barrier between knowing what was being misspelled.

Table 1. SASHELP.ZIPCODE Results for ZIP = 54449

ZIP	54449
X	-90.187920
Y	44.612100
ZIP_CLASS	
CITY	Marshfield
STATE	55
STATECODE	WI
STATENAME	Wisconsin
COUNTY	141
COUNTYNM	Wood
MSA	0
AREACODE	715
AREACODES	715
TIMEZONE	Central
GMTOFFSET	-6
DST	Y
PONAME	Marshfield
ALIAS_CITY	
ALIAS_CITYN	Bakerville||Lindsey||McMillan
CITY2	MARSHFIELD
STATENAME2	WISCONSIN

Table 2. SASHELP.ZIPCODE Results for City='Marshfield'

ZIP	02050	05658	54404	54449	54472	65706
X	-70.736310	-72.374820	-90.168447	-90.187920	-90.175187	-92.905687
Y	42.126517	44.328420	44.667961	44.612100	44.662245	37.339115
ZIP_CLASS			U		U	
CITY	Marshfield	Marshfield	Marshfield	Marshfield	Marshfield	Marshfield
STATE	25	50	55	55	55	29
STATECODE	MA	VT	WI	WI	WI	MO
STATENAME	Massachusetts	Vermont	Wisconsin	Wisconsin	Wisconsin	Missouri
COUNTY	23	23	141	141	141	225
COUNTYNM	Plymouth	Washington	Wood	Wood	Wood	Webster
MSA	1120	0	0	0	0	7920
AREACODE	781	802	715	715	715	417
AREACODES	781/339	802	715	715	715	417
TIMEZONE	Eastern	Eastern	Central	Central	Central	Central
GMTOFFSET	-5	-5	-6	-6	-6	-6

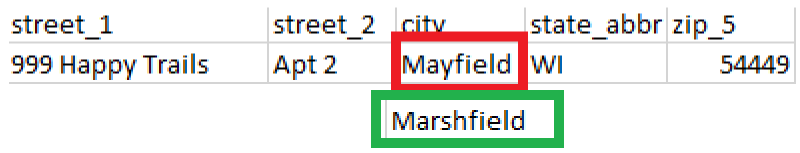
DST	Y	Y	Y	Y	Y	Y
PONAME	Marshfield	Marshfield	Marshfield	Marshfield	Marshfield	Marshfield
ALIAS_CITY						
ALIAS_CITYN				Bakerville|| Lindsey|| McMillan		
CITY2	MARSHFIELD	MARSHFIELD	MARSHFIELD	MARSHFIELD	MARSHFIELD	MARSHFIELD
STATENAME2	MASSACHUSETTS	VERMONT	WISCONSIN	WISCONSIN	WISCONSIN	MISSOURI

Table 3. SASHELP.ZIPCODES Results for SPEDIS('Marshfield',city) le 20

ZIP	CITY	STATENAME
02048	Mansfield	Massachusetts
02050	Marshfield	Massachusetts
05658	Marshfield	Vermont
16933	Mansfield	Pennsylvania
30055	Mansfield	Georgia
38236	Mansfield	Tennessee
44901	Mansfield	Ohio
44902	Mansfield	Ohio
44903	Mansfield	Ohio
44904	Mansfield	Ohio
44905	Mansfield	Ohio
44906	Mansfield	Ohio
44907	Mansfield	Ohio
54404	Marshfield	Wisconsin
54449	Marshfield	Wisconsin
54472	Marshfield	Wisconsin
57460	Mansfield	South Dakota
61854	Mansfield	Illinois
65704	Mansfield	Missouri
65706	Marshfield	Missouri
71052	Mansfield	Louisiana
72944	Mansfield	Arkansas
76063	Mansfield	Texas
98830	Mansfield	Washington

Identifying Entry Errors

Using that super helpful SASHELP.ZIPCODES dataset, you should have a list of all cities, states, and area codes for a given zip code. In Table 1, it is clear that if you live in zip code 54449, that you live in Marshfield, WI. However, while your mailing address may reflect Marshfield's zip code you may live in an area that goes by a different local name: Bakerville, Lindsey, or McMillian. Still if zip code is listed as 54449 and the city name isn't one of those 4 names, then it could be a fixable mistake (like in Figure 3.) or a case that needs to be handled differently.



street_1	street_2	city	state_abbr	zip_5
999 Happy Trails	Apt 2	Mayfield	WI	54449
		Marshfield		

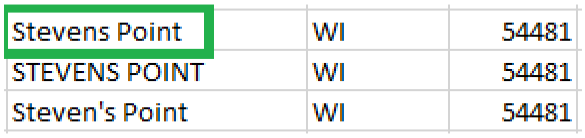
Figure 3. Example of Identified Data Error and Correction

Considering the other direction is where things get tricky. If the zip code was not listed or does not align with what you think valid values should be, then you might want to identify all applicable zip codes for a given city. In Table 2, there appears to be multiple matches that come back that tell us we need to provide more information. First, there are results that appear to be out of state and can be eliminated. Finally, it appears that two of the zip codes are marked as ZIP_CLASS = 'U'. The field label tells us: P=PO Box U=Unique zip used for large orgs/businesses/bldgs Blank=Standard/non-unique. Based on that information, we land with one valid option; 54449 is for Marshfield, WI. Unfortunately, that may not always be the case.

In Table 3, there is a listing of cities that are within a spelling distance (See Recommended Reading) of 45 from Marshfield. While arbitrary, 45 identifies a variety of valid city names that could be easily confused with Marshfield. By using the zip code information we can quickly come up with this list of potential alternates. In the case of Wisconsin, there isn't two separately named cities that appear, but both Massachusetts and Missouri have both a Marshfield and a Mansfield. In this case, I would be less comfortable "fixing" information identified as mismatched. However, in the case of Marshfield, WI we can be pretty certain 54449 is the right zip code.

Standardizing Components of Addresses

In Figure 3, the problem was that the name didn't match because it was the wrong name, but what if instead it was the right name, but abbreviated or had extraneous punctuation like in Figure 4? In this case, we can use the data to select a standard spelling and presentation for the city.



Stevens Point	WI	54481
STEVENS POINT	WI	54481
Steven's Point	WI	54481

Figure 4. Selecting a Standard Name

MAPPING

When it comes to mapping, zip codes are one of the most commonly utilized regions, hypothetically due to size of region and variety of use cases. This is the case not only for the United States, but also worldwide. As demonstrated by Hadden, SASHELP.ZIPCODES can be used in conjunction with the GMAP and MAPIMPORT Procedures and annotation to map and highlight zip codes in a variety of ways (Hadden, 2014).

CALCULATING

If you have address data ready for use and have moved on to using distance functions, there is a few ways to do so.

Distance to Destination

The most basic use of the ZIPCITYDISTANCE() function allows you to calculate how close a reference zip code is from another. For example, if you wanted to tell someone about how far away they live from your business, you could write something like:

```
DIST_FROM_MFLD = ZIPCITYDISTANCE(54449,patient_zip);
```

In this case, DIST_FROM_MFLD would be the number of miles between the zip code centroid for 54449 and the centroid of the patient's zip code.

Destinations within a Distance

In macro %ZipsNear produced by Richard Koopman, you can identify a list of zip codes within a certain distance of a given zip code using SASHELP.ZIPCODES and latitude / longitude coordinates (Koopmann, 2015). This example uses the raw Haversine formula to calculate the distances. In the %FIND_CLOSESTCITY macro discussed and presented below, I instead use the ZIPCITYDISTANCE() function.

Closest Destination

Let's say you have a business that has multiple locations. In this case, a common conversation with a customer might indicate which location is closest. In the case of potential customers, indication of the closest location would also commonly appear on advertisements.

There are several ways to get to the same answer programatically, but the basic concept would be to use each reference zip code (business location) and compare it to the static reference zip code (person location), then take the minimum.

To find the distance of the closest zip code, you might create a statement like such:

```
mindistance=min(zipcitydistance(54401,zip_5),zipcitydistance(54402,zip_5),
zipcitydistance(54403, zip_5));
```

or use a pairwise statement in the data step, sort, and keep the first using statements like these:

```
   dist = zipcitydistance(pat_zip,ref_zip);

   proc sort; by person dist;

   if first.person;
```

Please note that each of those would be featured in a DATA step or SORT procedure and would not be used as is.

%FIND_CLOSESTCITY MACRO: CONCEPT APPLICATION

Consider the macro %FIND_CLOSESTCITY, defined in APPENDIX 1. The main purpose of the macro is to compare a reference zip code to a pre-defined list of reference zip codes. It explores:

- Minimal cleaning before merging to SASHELP.ZIPCODES
- How to obtain a list of reference zip codes by providing the cities and states and using SASHELP.ZIPCODES
- Using PROC SQL to do the pairwise distance calculations using ZIPCITYDISTANCE () function
- Applying “closest” cities to a row per person identifier input

Development of this macro stemmed from desire to provide the nearest recruitment location for a given patient address. The study in question, *All of Us*® (U.S. Department of Health & Human Services (HHS), 2023), is a multi-site program that offers an in person enrollment visit and bio specimen collection. Study inclusion and exclusion criteria defined a list of potential participants to target. Since there were “pop-up” enrollment events in different cities and a separate set of consistent places potential enrollees could attend, it was important to identify the closest location for each potential participant given a list of input locations. This executable takes the minimum known information: cities available for recruitment and patient for targeting and runs from there. The interim data sets that identify closest zip code for a given reference zip code were often used to determine area coverage for the study, so there are options which allow for only generation of that file. Likewise, the distance that a person is willing to travel might be arbitrary, but knowing how far away each additional zip code was and how many persons could be added, was often used to make operational decisions on what radius to cover with marketing and targeted mailing campaigns. Once a radius was selected, it needed to be applied, so additional options are present to accomplish a final patient list faster by only considering those in a given radius to begin with. By providing options, this same macro can accomplish many tasks in a timely manner.

There are some concepts applied in this macro that go outside the topic of zip codes. For example. “Check Steps” that provide custom error messages to be sure the process stops and gives explanations if assumptions of the data are not met. To learn more about custom error messages, good times to do so, and macros for executing checks, check out Offensive Programming: A Threesome of Error-Throwing Macros from SESUG 2022 (McMullen, 2022).

CONCLUSION

From their origination in 1963, zip codes have helped to save time in a variety of ways. SAS provides several resources for making zip codes helpful to you: data sets, functions, and utilization procedures. Each of these resources can save you time and help you to accomplish data cleaning, calculation, and mapping tasks. Further, a combination of these resources can propel you toward of variety of ends such as finding a nearest location, locations in a given radius, mapping, or calculation of distances. Many different papers, presentations, and macros already exist to help you achieve all of these ends. Finally, this paper directs you to resources while providing yet another macro to help you understand the basic concepts and apply them.

REFERENCES

Fickbohm, D. (2007). SAS PROVIDES ZIPCODES. *Western Users of SAS Software (WUSS).* San Francisco, CA: LexJansen. Retrieved from https://www.lexjansen.com/wuss/2007/CodersCorner/COD_Fickbohm_SASProvides.pdf

Hadden, L. (2014). Where in the World Are SAS/GRAPH® Maps? *MidWest SAS Users Group (MWSUG) Conference.* Chicago, IL: MidWest SAS Users Group. Retrieved from https://www.mwsug.org/proceedings/2014/DV/MWSUG-2014-DV07.pdf

Hadden, L., & Zdeb, M. (2006). A Well-Kept SAS® Secret. *North East SAS Users Group (NESUG) Conference.* Philadelphia: LexJansen. Retrieved from https://www.lexjansen.com/nesug/nesug06/po/po07.pdf

Hadden, L., & Zdeb, M. (2010). ZIP Code 411: Decoding SASHELP.ZIPCODE and Other SAS® Maps Online. *SAS Global Forum.* Seattle: LexJansen. Retrieved from https://support.sas.com/resources/papers/proceedings10/219-2010.pdf

Koopmann, R. (2015, March 13). ZIPSNEAR.SAS. MN, USA: GitHub Gist. Retrieved from https://gist.github.com/rkoopmann/0cc179f496bd026a0171

McMullen, Q. (2022). Offensive Programming: A Threesome of Error-Throwing Macros. *SouthEast SAS Users Group (SESUG) Conference.* Mobile, AL: LexJansen. Retrieved from https://www.lexjansen.com/sesug/2022/SESUG2022_Paper_187_Final_PDF.pdf

SAS Institute Inc. (2016). *SAS® 9.4 Functions and CALL Routines: Reference, Fifth Edition.* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2021, March 31). *SAS® 9.4 and SAS® Viya® 3.5 Programming Documentation*. Retrieved from SAS® Help Center: https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/pgmsashome/home.htm

Terrall, E. (2013, June 28). ZIP Code Introduced. *This Month in Business History*. Retrieved from https://guides.loc.gov/this-month-in-business-history/july/zip-code-introduced#:~:text=The%20ZIP%20in%20ZIP%20Code,the%20speed%20of%20mail%20delivery.

U.S. Department of Health & Human Services (HHS). (2023, September 6). *National Institutes of Health* . Retrieved from All of Us Research Program: https://www.joinallofus.org/

U.S. Postal Service. (1963). Annual Report of the Postmaster General. p. 8.

ACKNOWLEDGMENTS

This paper, presentation, and macro would not have come into existence if not for the National Institutes of Health national *All of Us Research Program®* (National Institutes of Health , 2023) led locally at Marshfield Clinic Health System by Scott Hebbring, PhD. To learn more about this exciting program, visit https://www.joinallofus.org/ .

I want to thank and acknowledge the many authors that have published on this material previously and acknowledge Louise Hadden for being my first insight into the topic. As I began my programming, her papers were the first to assist me.

RECOMMENDED READING

- *Maps and Geocoding Files - https://support.sas.com/en/knowledge-base/maps-geocoding.html*

- *The SPEDIS Function* - *https://documentation.sas.com/doc/en/vdmmlcdc/8.1/lefunctionsref/p0vmuxh8ljfn7on164nsgvmdrc5d.htm*

CONTACT INFORMATION

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

Brooke Ellen Delgoffe, M.S.
Marshfield Clinic Health System
delgoffe.brooke@marshfieldresearch.org
brooke_delgoffe@hotmail.com
https://www.linkedin.com/in/brookedelgoffe/

APPENDIX 1: %FIND_CLOSESTCITY MACRO FULL CODE

```
/**********************************************
PROGRAM         : Closest City in List (Zip Code)
PROGRAMMER      : Brooke Ellen Delgoffe
DATE            : 03-24-2022

PURPOSE: Used to determine which city specified
is closest based on zip code of the identifier and
a given list of city options.

Example Interpretation:
---------------------------------------------
Of the options Marshfield and Wisconsin Rapids
the closest city to the patient is Marshfield, based on
zip code distance of 32.3 miles between the patients zipcode of 54401
(givenin input dataset) and zipcodes of Marshfield (54402,54472,54449) and
Wisconsin Rapids (54494,54495)

WORK.ZIP_CODE_DISTANCES =
ROW_ZIP   CITY_ZIP    CITY              miles
54401     54404 Marshfield              32.3
54401     54472 Marshfield              32.8
54401     54449 Marshfield              35.5
54401     54495 Wisconsin Rapids  39.8
54401     54494 Wisconsin Rapids  42.2

Considerations:
---------------------------------------------
--Uses zipcode distance to determine "closest"
--All Zip Codes associated with a given city are
   used to determine distance to that city
--In the case of equal distance, alphabetical first is
  arbitrarily selected
--May not have a variable in the input dataset named "identifier"
```

```
    that is not used as the identfier variable. Use a
    rename parameter on input/output dataset names to avoid this
--To run without an identifier, create a column named identifer with
    value of zero that is droped in the drop statement on the named output
(OUTDAT = CLOSEST_CITY (drop=identifier)).


Minimum Definition:
-------------------
%find_closestcity(POPDAT=SAMPLE (rename=(ZIP=ZIP_5))
               ,CITY_LIST=%bquote(Marshfield-Wisconsin Rapids)
               );

Full Definition:
------------------
%find_closestcity (POPDAT=INDAT
                                  ,OUTDAT=CLOSEST_CITY
                                  ,CITY_LIST=%bquote(Marshfield)
                                  ,IDENTIFIER=[Variable Name]
                                  ,ZIP_OUT=[Y,N,[Name]]
                                  ,STATE_LIST=%bquote(WI)
                                  ,CLEAN_DATASETS=[Y/N]
                                  );

Parameters:
----------------------------------------------
POPDAT          : Input Dataset
                 REQUIRED VARIABLES: IDENTIFIER, ZIP_5

CITY_LIST : Dash delimited list of cities to
                consider. Contains spaces and wrapped
                with %bquote() in most cases.
                DEFAULT: Marshfield

OUTDAT          : (Optional) Name of output dataset. Can contain libname.
                DEFAULT: CLOSEST_CITY

IDENTIFIER  : (Optional) Set the name of the variable which will be used
                as a distinct identifier.
                Default: MHN

ZIP_OUT       : (Optional) Indicates whether a list of zipcodes
                found for the city list is generated.
                VALUES:[Y,N,[Name]] DEFAULT: N
                      N: No dataset created
                      Y: Yes --Datasets CLOSEST_CITIES & ZIP_CODE_DISTANCES
                            are created with listing of
                            zipcodes
             [Name]: Specified name is used for
                         dataset containing list of
                         closest zip codes used. May contain
                         libname reference.

STATE_LIST      : (Optional) Dash delimited list of states to
                consider. Using two-letter acronym and
                wrapped     with %bquote() in most cases.
                DEFAULT: WI
```

```
CLEAN_DATASETS : Indicates whether interim datasets
                should be deleted.
                VALUES:[Y,N]    DEFAULT: Y

ZIP_DAT_ONLY   : Allows for running only the input
                      city list to zip codes portion.
                VALUES: [Y,N] DEFAULT: N

W_IN_MILES      : Indicates how many miles from to consider.
                Adjusts interpretation to "Closest within X Miles".
                While it will not error if decimals given the
                measurement is in integer format so integers
                are expected.
                VALUES:[Number}, A   DEFAULT: A
                      A: Any Number of Miles

Areas for Improvement:
-----------------------------------------------
-Currently considers all valid combinations of city list
and state list. In the case where not all combinations
of city and state are desired, this may cause unintended
results. (i.e. Include Marshfield, WI , but not Marshfield, IN
when both WI and IN are in the state list)

***********************************************/

/****************************/
/*       Settings               */
/****************************/
options minoperator mindelimiter=','; *allow for use of in() with macro
statements;

/*******************************************************
*                                                      *
*                Define Macro Parameters               *
*                                                      *
*******************************************************/


%macro find_closestcity (
POPDAT=INDAT /*Input Dataset containing: IDENTIFIER , ZIP_5*/
,OUTDAT=CLOSEST_CITY /*Output Dataset*/
,CITY_LIST=%bquote(Marshfield) /*Dash delimited list of Cities considered*/
,ZIP_OUT=N /*Indicates if Zipcodes for CITY_LIST are output to a dataset*/
,STATE_LIST=%bquote(WI) /*Dash delimited List of States(two-letter)
considered*/
,CLEAN_DATASETS=Y /*Indicates whether interim datasets should be deleted*/
,ZIP_DAT_ONLY=N /*Allows for Running Just the Input Zip Code portion*/
,W_IN_MILES=A /*Allows for Maximum Mileage to be set*/
,IDENTIFIER=MHN /*Variable Name to Use as Distinct Identifier*/
                              );
/****************************/
/*  Set identifier              */
/****************************/
*To be sure they can select any variable name, set that variable to the
same name in a copy dataset. Do not change original dataset.;
```

```
%if &IDENTIFIER. ne IDENTIFIER %then %do;
   data IN_DATA;
   set &POPDAT.;
   IDENTIFIER = &identifier.;
   run;
   %end;

/***************************/
/*  Check Parameters       */
/***************************/
*makes sure the input dataset actually has data in it and has the
   identifier variable;
proc sql noprint;
select case when (select count(*) from IN_DATA) = 0 then "ERROR: No records
in input dataset."
                when n = 2 then "NOTE: Zip_5 and &Identifier. have been
found"
                else "ERROR: A Required Field (ZIP_5) or specified
identifier (&IDENTIFIER.) have not been found. Check input dataset contains
these variables and that they are spelled correctly."
                end as CHECK_MESSAGE
                into :CHECK_MESSAGE
    from (select count(*) as n
                from SASHELP.VCOLUMN
                where MEMNAME = 'IN_DATA'
                and upcase(NAME) in('ZIP_5',"IDENTIFIER")
          )
;
select compress(put(count(distinct identifier),10.),,'kdp') 'Valid
Identifiers in Input Dataset' into :IDENTIFIER_COUNT
   from IN_DATA
;
select compress(put(count(distinct zip_5),10.),,'kdp') 'Valid Zip Codes in
Input Dataset' into :ZIP_COUNT
   from IN_DATA
;quit;

%put &check_message.;
%put NOTE: There are &identifier_count. distinct identifiers in dataset
&POPDAT.;
%put NOTE: There are &zip_count. distinct values of ZIP_5 in dataset
&POPDAT.;


/***************************/
/*  City Zip Codes               */
/***************************/

*read over each city in the list and each state
in state list and create a row in the CITIES
dataset for each combination. Perform minimal
cleaning to be sure it can merge with the
SASHELP.ZIPCODES dataset;
data CITIES;
length CITY_RAW CITY $35.; *Use Length Defined in SASHELP.ZIPCODES;
%let n=1;
%do %while (%scan(&city_list.,&n.,'-') ne %str());
```

```
%let CITY=%scan(&city_list.,&n.,'-');
CITY_RAW="&CITY.";
CITY=propcase(CITY_RAW);
    %let s=1;
    %do %while (%scan(&state_list.,&s.,'-') ne %str());
    %let STATE=%scan(&state_list.,&s.,'-');
    STATE_RAW="&STATE.";
    STATECODE=upcase(STATE_RAW);
    OUTPUT;
    %let s= %eval(&s.+1);
    %end;*state list scan loop;
%let n= %eval(&n.+1);
%end;*city list scan loop;
run;
proc sort; by CITY STATECODE;

*create a copy of SASHELP.ZIPCODES (contains all zipcodes and city/state)
that is sorted by city and statecode, then merge (inner join) to
get only the ones that match between them;
proc sort data=SASHELP.ZIPCODE OUT=ZIPCODE_DAT; by city statecode;
data ZIPCODE_DAT;
merge ZIPCODE_DAT(in=z) CITIES(in=c);
by CITY STATECODE;
if z and c;
run;

*check that at least one combination of city and state is found;
title h=2 'Cities/States with Zip Codes Found';
proc sql;
select distinct CITY, STATECODE
    from ZIPCODE_DAT
;quit;
title;

*check input dataset contains 1+ IDENTIFIERS;
%if &IDENTIFIER_count. = 0 %then %do;
    %put NOTE: There were no identifiers identified in the input dataset.
Execution will stop.;
%end;
%else %if &ZIP_DAT_ONLY.=Y %then %do;
    %put NOTE: ZIP_DAT_ONLY is Yes. Execution will not continue;
%end;
%else %do;


    /****************************/
    /*    Inputted Zip Codes      */
    /****************************/
    *select distinct combinations of identifier and
    zip code from the input dataset;
    data ROWS_W_ZIP; IDENTIFIER=0; ZIP_5=00000; run; *initialize dataset to
avoid errors;

          proc sql;
          insert into ROWS_W_ZIP
                select distinct
                IDENTIFIER
                ,ZIP_5
                from WORK.IN_DATA
                where IDENTIFIER not in(.,0)
                and zip_5 not in(.,0)
          ;quit;


    data ROWS_W_ZIP;
    set ROWS_W_ZIP;
    where IDENTIFIER ne 0 and zip_5 not in(.,0); *Remove initialization
row;
    run;

    *Report and Store number of distinct identifiers found;
    proc sql;
    select count(distinct IDENTIFIER) 'Identifiers with Valid Zip Code'
into :valid_zip_cnt
          from ROWS_W_ZIP
    ;quit;

/*****************************/
/*  Zip Codes Found Check   */
/*****************************/
*check if any identifiers were found and only continue
to the identifier specific distance portion if there is at least one;
%if &valid_zip_cnt.=0 %then %do;
    %put ERROR: There were no identifiers with a valid zip code found.;
%end;
%else %do;

    /*****************************/
    /*        Closest City        */
    /*****************************/
    *create an entry for each zip code combination.
    Pull in the city from SASHELP for use in standardization.
    Calculate zipcode distance (between zipcode centroids).
    Apply limitation on distance conditionally.;

    proc sql;
    create table zip_code_distances as
          select distinct
          p.ZIP_5 as ROW_ZIP
          ,i.ZIP as CITY_ZIP
          ,i.CITY
          ,zipcitydistance(p.ZIP_5,i.ZIP) as miles
    from (select distinct ZIP_5 from ROWS_W_ZIP) p
    full join ZIPCODE_DAT i on p.ZIP_5 ne 0
    %if &W_IN_MILES. ne A %then %do;
    having miles between 0 and &W_IN_MILES.
          %end;
    order by ROW_ZIP, miles
    ;quit;

    *Restrict to Only Closest;
    data CLOSEST_CITIES;
    set ZIP_CODE_DISTANCES;
    by ROW_ZIP;
    if first.ROW_ZIP; *Closest City;
    run;

    /************************************/
    /*    Closest City per Identifier         */
    /************************************/
    *create an output data set that is everything from the original dataset
    with the new variables attached to end. Do Not Change Input Data.;

    proc sql;
    create table &OUTDAT. as
          select distinct
                i.*
                ,c.CITY as CLOSEST_CITY
                ,c.CITY_ZIP as CLOSEST_REF_ZIP
                ,c.MILES as MILES_TO_REF_ZIP
          from &POPDAT. i
          left join ROWS_W_ZIP z on i.&IDENTIFIER.=z.IDENTIFIER
          left join CLOSEST_CITIES c on z.ZIP_5=c.ROW_ZIP
    ;quit;

%if &ZIP_OUT. ne Y and &ZIP_OUT. ne N %then %do;
    *Conditionally create a named dataset containing the closest cities;
    data &ZIP_OUT.;
    set CLOSEST_CITIES;
    run;
%end; *end zip out named data loop;
%if &CLEAN_DATASETS.=Y %then %do;
    *Conditionally drop intermediate tables;
    proc datasets nolist;
    delete IN_DATA ROWS_W_ZIP CITIES ZIPCODE_DAT ;

    %if &ZIP_OUT. ne Y %then %do;
    *drop zip code distance datasets unless requested;
    delete CLOSEST_CITIES ZIP_CODE_DISTANCES;
    %end; *end zip out loop;
    run;
%end;*end clean datasets loop;

%end; *Valid Zip > 0 Check;
%end; *Parameter Check Loop;
%mend;
```