

Paper 5026-2020

Application of Google Distance Matrix API to investigate healthcare access: a study on Hokkaido, Japan

Anna Tsutsui and Yuko Ohno, Graduate School of Medicine, Osaka University

ABSTRACT

Recently, a web service called web Application Programming Interface (API) has attracted the attention of academic researchers for evaluating healthcare access by estimating travel times and distances. Particularly, Google Maps API is now being adopted as a new and simple tool for spatial analyses. Google Maps API requires information on the search criteria but provides the calculation results based on their own data and calculation algorithms. However, most researchers have used R or Python for this purpose. In this paper, we introduce the key steps for conducting similar studies using SAS® by providing an overview of Google Maps API (especially Google Distance Matrix API), Uniform Resource Identifier (URI), JavaScript Object Notation (JSON) format, and JSON map file. An analytical example is also provided to investigate healthcare access for children with cancer in Hokkaido, Japan. By using the Google Distance Matrix API, we obtained the estimated travel times to one childhood cancer hub hospital from residences by considering the locations of local government offices. The estimated departure times were then derived, and the results were visualized on a choropleth map. Our study suggests that the healthcare access is likely to be investigated by using SAS and web APIs, especially Google Distance Matrix API. To assess healthcare access properly, researchers will need large-scale real-world data and software that enables a series of processing. One such software will be SAS.

INTRODUCTION

Many past studies have estimated travel times or distances by using geographical information systems (GIS) software. However, there are a few challenges in using the software because it requires the input of a well-defined road network dataset and significant efforts in implementing the task (Wang & Xu, 2011).

Recently, a web service called web Application Programming Interface (API) has attracted the attention of presenters in SAS® Global Forums (Gadidov, Zhang, & Zhou, 2018; Roy & Na, 2012), as well as academic researchers to evaluate healthcare access by estimating travel times and distances (Shaw et al., 2017; Wang & Xu, 2011). Web APIs generally require the information on the search criteria but provide the calculation results based on its own data and calculation algorithm. However, R and Python are popularly adopted.

Therefore, this study provides the key steps for conducting similar studies using SAS by presenting an overview of Google Maps API (especially Google Distance Matrix API), Uniform Resource Identifier (URI), JavaScript Object Notation (JSON) format, and JSON map files. An analytical example is also provided, showing the SAS code to read a shapefile and plot a choropleth map.

OVERVIEW OF GOOGLE MAPS API

GOOGLE MAPS API

APIs are a set of programming tools that enable a program to communicate with another program or operating system and help software developers create their applications (i.e., pieces of software) (Oxford University Press, n.d.). "Web API" is an API for either a web server or a web browser. With web APIs, users can utilize various web services from a web browser, software, or other HTTP clients.

Google currently offers various web API services for maps, places, and routes in the framework of the Google Maps Platform. It is called the Google Maps API and its service is similar to the Google Maps website. However, unlike the website, the platform introduces a pay-as-you-go model. The charge per request is low, and US\$200 credit is available each month (Google Maps Platform, 2020c).

The maximum number of "Queries Per Second" (QPS) is also specified per API. Some APIs are limited to 50 QPS (Google Maps Platform, 2019). SAS users, in this case, should add code such as the SLEEP function in the DATA step to decrease QPS. Google publishes policies and terms of service, and users are encouraged to confirm these in advance.

GOOGLE DISTANCE MATRIX API

The Google Distance Matrix API is an API for routes. The Google Maps Platform (2020a) explained that the service calculates the most efficient routes considering the travel time and other factors. The number of destinations and origins may each be one or more. Theoretically, a maximum of 40,000 requests are available free of charge each month (Google Maps Platform, 2020c). The QPS limit is 1,000 (Google Maps Platform, 2019); no special coding is likely required to decrease QPS.

The API does not provide the details of routes but provides the estimated travel times and distances based on a user request. The response data structure from the API is simple when users specify one destination and one origin in a request. Users who do not know the data format (JSON format) will find out that it will take 1,865 km and 17 hours 22 minutes from Denver to Dalles:

```
{
  "destination_addresses" : [ "The Dalles, OR 97058, USA" ],
  "origin_addresses" : [ "Denver, CO, USA" ],
  "rows" : [
    {
      "elements" : [
        {
          "distance" : {
            "text" : "1,865 km",
            "value" : 1865271
          },
          "duration" : {
            "text" : "17 hours 22 mins",
            "value" : 62543
          },
          "status" : "OK"
        }
      ]
    }
  ],
  "status" : "OK"
}
```

The API has some limitations. As per our investigations, the details of the calculation algorithms were unpublished. Moreover, the traffic information is unavailable without fulfilling specified conditions (Google Maps Platform, 2020b).

USAGE OF THE GOOGLE DISTANCE MATRIX API

API KEY

Before starting the API, users must obtain an identifier called an "API key" on the platform. The key is a unique identifier associated with each user's project. It is issued if the user creates a Google account and then completes the specific procedures on the platform.

Because real keys should be hidden for security reasons, this paper represents an issued key as "<API_KEY>" for convenience.

BASIC USAGE WITH URI

After gaining the API key, users can access the Distance Matrix API through an HTTP request by using a string of characters. The string is called URI or Uniform Resource Locator (URL). The basic form is as follows (Google Maps Platform, 2020b):

`https://maps.googleapis.com/maps/api/distancematrix/outputFormat?parameters`
The three **bold** parts should be changed in each request. The first, "https," may be changed to "http" although it is not recommended for security reasons. The second, "**outputFormat**," specifies the response data format, which is either "json" or "xml." When users adopt the "https" and JSON format, the URI is specified as follows:

`https://maps.googleapis.com/maps/api/distancematrix/json?parameters`
The third, "**parameters**," specifies the other settings. The parts must include at least three parameters: KEY, ORIGINS, and DESTINATIONS. The KEY parameter specifies the API key, and the ORIGINS and DESTINATIONS parameters specify the locations. These parameters should be joined by the ampersand (&). For example, these parameters are expressed as follows when users request a travel time and distance from Denver to Dalles:

```
key=<API_KEY>&origins=Denver&destinations=Dalles
```

Users can then test the URI with a web browser by entering it in the address bar. Figure 1 shows a response from the API based on the following URI:

```
https://maps.googleapis.com/maps/api/distancematrix/json?key=<API_KEY>&origins=Denver&destinations=Dalles
```

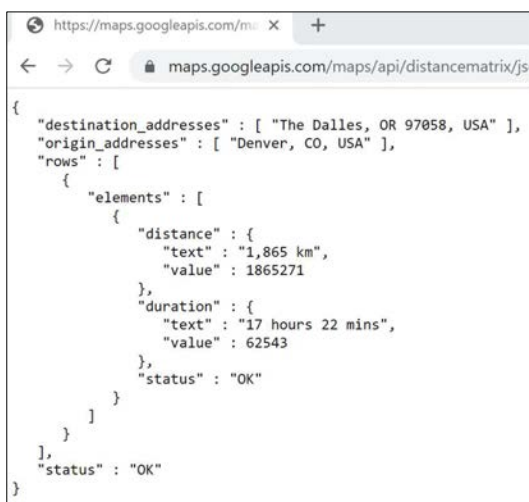


Figure 1 Response provided from the API using a web browser

ADVANCED USAGE WITH URI – DETAILS OF “PARAMETERS”

Users can specify the various setting details by adding other information in the “parameters.” This paper presents two types of parameters.

First, the **MODE** parameter specifies the method of transportation. By default, the setting is driving. It may be changed to other transportation modes: walking, bicycling, and transit. Note that the available modes differ depending on the country and region (Google Maps Platform, 2019). An example to specify the transit mode is as follows:

```
mode=transit
```

Second, the **ARRIVAL_TIME** specifies the arrival time, and the **DEPARTURE_TIME** specifies the departure time. These parameters cannot coincide. Each time should be a time in the future and be specified by the cumulative number of seconds since midnight, January 1, 1970, UTC. When the departure time is July 30, 2012, at 9:45 am, the parameter is specified as follows (Google Maps Platform, 2020b):

```
departure_time=1343641500
```

SAS users can obtain the cumulative seconds in one statement:

```
%let _sec=%sysfunc(sum("30JUL2012T09:45:00"dt, - 315619200)) ;
```

This is because a SAS DateTime value is the cumulative number of seconds since midnight of January 1, 1960. This implies that the base times between SAS and the API differ by ten years, or 315,619,200 seconds. Thus, SAS users can obtain the cumulative seconds by adding -315,619,200 seconds to a SAS DateTime value.

SAS CODE

Once the URI is considered appropriate, SAS users can send it to the API, receive the results, and read the JSON data into SAS. SAS originally reads JSON data by parsing techniques or the PROC DS2 procedure, and with SAS 9.4 M4, SAS implemented JSON engine (Hemedinger, 2016). Therefore, the SAS code was simple to create in SAS Studio 3.8 (SAS 9.4.1M6, University Edition) and are as follows:

```
filename MAP_API url "<URI>" debug ;  
libname MAP_API json ;
```

The FILENAME statement enables access to the web API by using the URL access method. Here, "<URI>" indicates an arbitrary URI. The LIBNAME statement provides read-only sequential access to JSON data (SAS Institute Inc., 2017).

The DEBUG option in the FILENAME statement is useful because it provides the following debugging information to the SAS log:

```
NOTE: JSON data is only read once. To read the JSON again, reassign the  
      JSON LIBNAME.  
NOTE: >>> GET  
      /maps/api/distancematrix/json?key=<API_KEY>&origins=Denver&  
      destinations=Dalles HTTP/1.0  
NOTE: >>> Host: maps.googleapis.com:443  
NOTE: >>> Accept: */*  
...
```

Meanwhile, because the URI contains the ampersand (&), SAS attempts to resolve macro variables of each parameter name and displays the following log:

```
WARNING: Apparent symbolic reference ORIGINS not resolved.  
WARNING: Apparent symbolic reference DESTINATIONS not resolved.
```

To avoid the warnings, each ampersand in URI should be masked by %STR() function:

```
https://maps.googleapis.com/maps/api/distancematrix/json?key=<API_KEY>%str(  
&)origins=Denver%str(&)destinations=Dalles
```

SAS CODE – SPECIFIC LOCATION NAME

When we sent a request with a location name including space by SAS Studio 3.8 or a request with a Japanese location name by SAS Studio 3.7.1 (SAS 9.4.1M5), we found error messages in the log:

```
ERROR: Bad request. Use the debug option for more info.  
ERROR: Error in the LIBNAME statement.
```

A URI should originally consist of a limited set of characters: digits, letters, and a few graphic symbols (Berners-Lee, Fielding, & Masinter, 2005). Therefore, the cause was presumed to be related to the characters not allowed in the URI. We then added the URLENCODE function to the SAS code to perform a percent-encoding function that encodes characters that might otherwise be significant when used in a URL (SAS Institute Inc., 2016). Consequently, the characters were changed to strings repeated with "%xx" (e.g., "紐育" to "%E7%B4%90%E8%82%B2"):

```
option urlencoding=utf8 ;  
%let _origins=%qsysfunc(urlencode(紐育)) ; /*紐育: New York*/  
%let _destinations=%qsysfunc(urlencode(Washington DC)) ;  
filename MAP_API url  
"https://maps.googleapis.com/maps/api/distancematrix/json?key=<API_KEY>%str  
(&origins=&_origins.&_str(&destinations=&_destinations.)" encoding="utf8"  
debug ;  
libname MAP_API json ;
```

Here, the first line set an encode for percent-encoding. The encode was not Shift_JIS but UTF-8 because we experienced an error in our SAS session. The second and third lines created macro variables of origin and destination. Further, the %QSYSFUNC function called the URLENCODE function, and the URLENCODE function executed percent-encoding. The %QSYSFUNC instead of the %SYSFUNC was used to escape the following numerous log messages:

```
WARNING: Apparent invocation of macro E7 not resolved.  
WARNING: Apparent invocation of macro B4 not resolved.  
...
```

The fourth to the eighth last lines were almost the same as in the previous code. However, the location names in the <URI> were replaced by macro variables and the ENCODING option was added.

READ JSON DATA

JSON DATA

JSON is a data-interchange format that is commonly used in web APIs. It is a text format based on a subset of JavaScript that is language independent. The structure of the JSON data is simple, human-readable, and machine-readable.

JSON is built on two structures ("JSON," n.d.):

- First, a collection of name/value pairs.
- Second, an ordered list of values.

Moreover, the data take the following forms in JSON ("JSON," n.d.):

- Object: This is an unordered set of name/value pairs. It is surrounded by braces. Each name is followed by a colon and a value. The pairs are separated by commas:

```
{"text" : "1,865 km", "value" : 1865271}
```

- Array: This is an ordered collection of values. It is surrounded by brackets. The values are separated by commas. A simple example is provided as follows, because the array

structure in Figure 1 is complex:

```
{ "place": [ "New York", "Washington, DC" ],
  "score": [ 8, 6, 4, 5 ],
  "player": [{ "name": "Ken", "age": 14 }, { "name": "Taro", "age": 15 } ] }
```

- Value: This is a string in double quotes; a number; a true or false; a null; an object; or an array. These structures can be nested as shown in Figure 1.

JSON DATA INTO SAS

Figure 2 shows a list of SAS datasets in a library that was read into SAS. Although the data structure in Figure 1 was simple, the data was split into a total of seven datasets.



Figure 2. List of SAS datasets in a library that was read into SAS

The ALLDATA dataset contained all the JSON data (Figure 3). The data have four types of variables (i.e., P, P1-P4, V, Value). The JSON data were also stored in the other datasets per object. The datasets in this case contained a part of the data in one record with two to four variables (Figure 4).

Columns	Total rows: 14	Total columns: 7
<input checked="" type="checkbox"/> Select all		
<input checked="" type="checkbox"/> P	1	1 destination_addresses
<input checked="" type="checkbox"/> P1	2	2 destination_addresses destination_addresses1
<input checked="" type="checkbox"/> P2	3	1 origin_addresses
<input checked="" type="checkbox"/> P3	4	2 origin_addresses origin_addresses1
<input checked="" type="checkbox"/> P4	5	1 rows
<input checked="" type="checkbox"/> V	6	2 rows elements
<input checked="" type="checkbox"/> Value	7	3 rows elements distance
	8	4 rows elements distance text
	9	4 rows elements distance value

Figure 3. ALLDATA dataset

Columns	Total rows: 1	Total columns: 3
<input checked="" type="checkbox"/> Select all		
<input checked="" type="checkbox"/> ordinal_root	1	1
<input checked="" type="checkbox"/> ordinal_destination_addresses		
<input checked="" type="checkbox"/> destination_addresses1		1 The Dalles, OR 97058, USA

Figure 4. DESTINATION_ADDRESSES dataset

MODIFY JSON DATA BY JSON MAP FILE

JSON MAP FILE

Because the original JSON data had a simple structure, the previous datasets also had a simple structure. Hence, users can easily create an analysis-ready dataset through programming.

Meanwhile, users can change how SAS reads the JSON data by editing the file known as the JSON map. The JSON engine uses a JSON map file to describe the datasets in the specified JSON file (SAS Institute Inc., 2017). The file is generated by SAS when specified. Although the extension is ".map," it is text data and can be edited by a general text editor. Therefore, for example, users can specify SAS to store all information in one record in one dataset (Figure 5 and Figure 6).

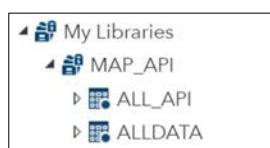
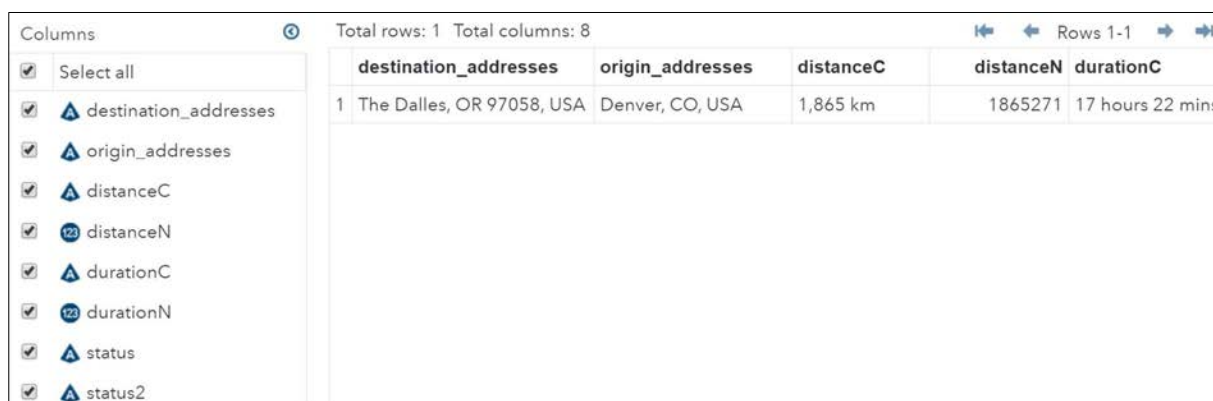


Figure 5. List of SAS datasets in a library that was generated using a JSON map file

A screenshot of the SAS dataset viewer for the 'ALL_API' dataset. The interface shows a 'Columns' panel on the left with a list of variables: 'Select all', 'destination_addresses', 'origin_addresses', 'distanceC', 'distanceN', 'durationC', 'durationN', 'status', and 'status2'. All variables are checked. The main area shows a table with 8 columns and 1 row. The columns are 'destination_addresses', 'origin_addresses', 'distanceC', 'distanceN', and 'durationC'. The data in the first row is: 'The Dalles, OR 97058, USA', 'Denver, CO, USA', '1,865 km', '1865271', and '17 hours 22 mins'.

	destination_addresses	origin_addresses	distanceC	distanceN	durationC
1	The Dalles, OR 97058, USA	Denver, CO, USA	1,865 km	1865271	17 hours 22 mins

Figure 6. ALL_API dataset

GENERATE A JSON MAP FILE BY SAS

Users can generate a JSON map file by adding the AUTOMAP and MAP options in the LIBNAME statement. The option "AUTOMAP=CREATE" indicates a request for SAS to generate a JSON map and to write it to the MAP="map physical name" that ends up with ".map" (SAS Institute Inc., 2017).

```
filename MAP_API url "<URI>" debug ;  
libname MAP_API json map="<map physical name>" automap=create ;
```

Alternatively, users can add the FILENAME statement and specify the MAP fileref in the MAP option as shown below:

```
filename JMAP "<map physical name>" ;  
filename MAP_API url "<URI>" debug ;  
libname MAP_API json map=JMAP automap=create ;
```

Output 1 shows the excerpt from the JSON map file that was created from the above code. Users can read the contents similar to the JSON file format.

Output 1. JSON map file (excerpt)

```
{
  "DATASETS": [
    {
      "DSNAME": "destination_addresses",
      "TABLEPATH": "/root/destination_addresses",
      "VARIABLES": [
        {
          "NAME": "ordinal_root",
          "TYPE": "ORDINAL",
          "PATH": "/root"
        },
        {
          "NAME": "ordinal_destination_addresses",
          "TYPE": "ORDINAL",
          "PATH": "/root/destination_addresses"
        },
        {
          "NAME": "destination_addresses1",
          "TYPE": "CHARACTER",
          "PATH": "/root/destination_addresses/destination_addresses1",
          "CURRENT_LENGTH": 19
        }
      ]
    },
    ...
  ]
}
```

Here, the JSON map consists of a DATASETS array (SAS Institute Inc., 2017).

```
{"DATASETS": [...]}
```

Each object in the DATASETS array describes a dataset in the library. Then, each dataset object contains a DSNAME string to specify a dataset name, a TABLEPATH string to specify how to delimit dataset observations, and an optional VARIABLES array:

```
"DSNAME": "destination_addresses",
"TABLEPATH": "/root/destination_addresses",
"VARIABLES": [...]
```

Each object in the VARIABLES array describes a variable in that dataset:

```
"NAME": "destination_addresses1",
"TYPE": "CHARACTER",
"PATH": "/root/destination_addresses/destination_addresses1",
"CURRENT_LENGTH": 19
```

Here, a NAME string specifies a variable name; a TYPE string specifies a variable type, namely NUMERIC, CHARACTER, or ORDINAL (ordinal variables), and a PATH string specifies the path in the JSON map file for the variable. When the variable type is CHARACTER, a CURRENT_LENGTH string is also included to show the maximum length of the variables in the JSON data. This is for documentation only and is ignored by the JSON engine.

MODIFY JSON DATA BY JSON MAP FILE

In this study, the JSON map file was edited to create one dataset as shown in Figure 5 and Figure 6, by following these steps.

1. Change the DSNAME of the first dataset to ALL_API
2. Change the TABLEPATH of the first dataset to "/root"
3. Remove all variables whose TYPE is ORDINAL
4. Move all variables to the VARIABLES array of the first dataset
5. Remove all second and subsequent dataset objects from the DATASETS array

6. Change the NAME when variable names are redundant because the dataset contained two "text" and two "distance" names (Figure 1)
7. Rename the NAME for simplification

Appendix 1 shows the edited JSON map file. Once the map file is modified, users can modify to AUTOMAP=REUSE in the LIBNAME statement:

```
libname MAP_API json map="<map physical name>" automap=reuse ;
```

Consequently, the SAS dataset structure will be changed by referring to the map file.

ANALYTICAL EXAMPLE

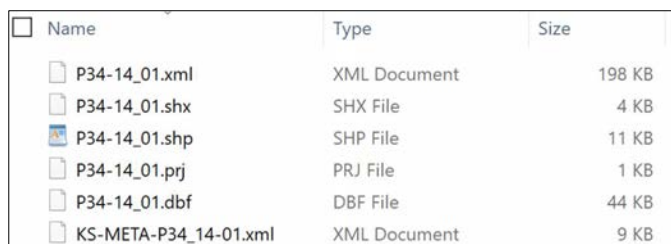
BACKGROUND

For childhood cancer, which is classified as a rare cancer, the centralization of cancer hospitals is important for providing quality medical treatment and care. The Hokkaido area, which is the northernmost island in Japan, has only one hospital designated as a childhood cancer hub hospital. Therefore, patients living far away from the hospital can spend much time traveling to the hospital. Hence, we investigated healthcare access from with respect to the travel times, using the Google Distance Matrix API.

DATA – SHAPEFILE

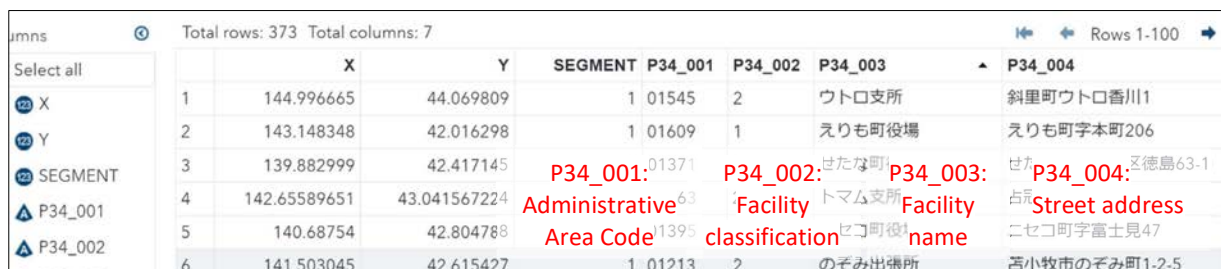
For the location data of patients, we used local government office data in a shapefile format (National Land Numerical Information download service, 2014). The shapefile format is an industry-standard format for storing the geometric location and associated attribute information. The "A" shapefile contains multiple files with different extensions (Figure 7). To read the data into SAS, at least the ".dbf," ".shp," and ".shx" files need to be stored in a same folder. We wrote the code (Odom, 2018) and read into SAS (Figure 8):

```
proc mapimport datafile=".\\P34-14_01_GML\\P34-14_01.shp" out=RAW.SHP ; run ;
```



Name	Type	Size
P34-14_01.xml	XML Document	198 KB
P34-14_01.shx	SHX File	4 KB
P34-14_01.shp	SHP File	11 KB
P34-14_01.prj	PRJ File	1 KB
P34-14_01.dbf	DBF File	44 KB
KS-META-P34_14-01.xml	XML Document	9 KB

Figure 7. "A" shapefile



	X	Y	SEGMENT	P34_001	P34_002	P34_003	P34_004
1	144.996665	44.069809	1	01545	2	ウトロ支所	斜里町ウトロ香川1
2	143.148348	42.016298	1	01609	1	えりも町役場	えりも町字本町206
3	139.882999	42.417145		01371		せたな町	せたな町 延徳島63-1
4	142.65589651	43.041567224		01395		トマム支所	占冠
5	140.68754	42.804788				セコ町役場	ニセコ町字富士見47
6	141.503045	42.615427	1	01213	2	のぞみ出張所	苫小牧市のぞみ町1-2-5

Figure 8. RAW.SHP dataset that was created from the shapefile

The data included the location data of patients (n=373). We excluded branch offices, branches and contact offices (n=184), and Sapporo City Hall (n=1) because the city had ward offices data. The analysis finally included 188 offices.

METHODS – DISTANCE MATRIX API AND CHOROPLETH MAP

We created a basic URI to place requests for calculating the estimated travel time from an office to Hokkaido University Hospital, the hub hospital, by driving to arrive at 10 a.m. on Wednesday, February 5, 2020. We then created and executed the SAS code to utilize the Distance Matrix API repeatedly by the %DO loop (Appendix 2). Consequently, we obtained the travel times in the area. The case for arriving on Wednesday, June 3, 2020, was also investigated because traffic conditions in this area are affected by snow in the winter season.

After obtaining the results from the API, we derived estimated departure times by subtracting each estimated travel time from 10 a.m. We then derived the `_DEPARTURE` variable, divided it into five groups every 1.5 hours, and created the `_dprtfmt` format for the variable.

The results were plotted on a choropleth map by the PROC GMAP procedure. Here, the data had a 5-digit code to identify administrative areas (e.g., "01101"). SAS, on the other hand, had a map dataset of Japan, JAPAN, in the MAPSGFK library, which contained the same code in ID variable with a prefix of "JP-" (e.g., "JP-01101"). Therefore, we created the ID variable in the result dataset and then executed the following code. Here, we referred only to the Hokkaido data of JAPAN map data by using WHERE= dataset option:

```
legend1 origin=(0,60) pct mode=share shape=bar(5,2) across=1 value=(h=2.5)
label=none ;
proc gmap data=_API map=MAPSGFK.JAPAN(where=(ID1="JP-01")) all ;
  id ID ;
  choro _DEPARTURE / outline=black legend=legend1 ;
  format _DEPARTURE _dprtfmt. ;
run ;
```

The estimated travel times on February 5 and June 3 were also compared.

RESULTS

Figure 9 shows the estimated departure time for arriving at the hub hospital at 10 a.m. The residents in most areas required more than 1.5 hours to travel, and therefore, they had to depart before 8:30 a.m.

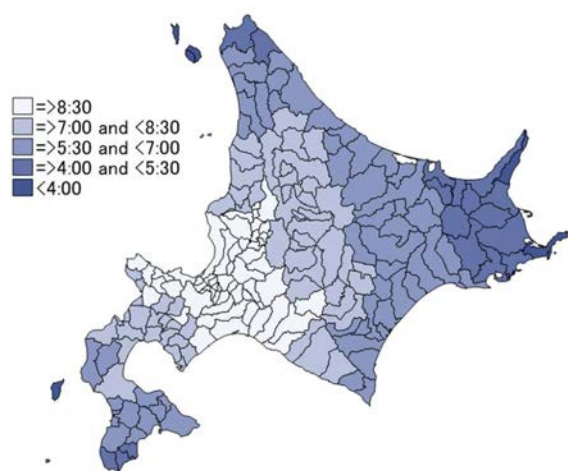


Figure 9. Estimated departure time

When the estimated travel times on February 5 and June 3 were compared, the times were the same in all the results.

DISCUSSIONS

This study obtained the estimated departure times for arriving at the childhood cancer hub hospital in the Hokkaido area. A previous study reported that only 14.1% of parents of children who had received cancer treatment considered a travel time exceeding one hour was acceptable for outpatient clinic visits (Sakaguchi, Oda, Shinkoda, & Manabe, 2014). The study suggests that a visit to the hub hospital may be a burden for residents in most areas.

The Distance Matrix API service did not consider the effect of snow, even though the actual travel times in the Hokkaido area could be affected. The results therefore should be carefully interpreted.

We used small data in this study. To assess healthcare access properly, researchers will need large-scale real-world data. Therefore, they need software that ensures stable and reliable routines, and that enables large amounts of repetitive processing, complicated data handling, and a variety of statistical analyses. One such software will be SAS.

CONCLUSION

This study provided the key steps to investigate healthcare access by showing an overview of the Google Maps API (especially the Google Distance Matrix API), URI, JSON format, and JSON map file. An analytical example was also provided. The web APIs have some limitations, such as unrevealed calculation algorithms or restriction of available functions; however, our study suggested that the healthcare access is likely to be investigated by using SAS and web APIs, especially Google Distance Matrix API.

REFERENCES

- Berners-Lee, T., Fielding, R., & Masinter, L. (2005). Uniform Resource Identifier (URI): Generic Syntax, *STD 66*, RFC 3986, doi:10.17487/RFC398
- Gadidov, B., Zhang, L., & Zhou, Y. (2018). Reducing Traveling Times for the Cobb County Fire Department. *Paper presented at the SAS Global Forum 2018*, Denver, CO, US. Retrieved from <https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2439-2018.pdf>
- Google Maps Platform. (2019). Google Maps Platform FAQ. Retrieved from <https://developers.google.com/maps/faq>
- Google Maps Platform. (2020a). Developer Guide - Directions API. Retrieved from <https://developers.google.com/maps/documentation/directions/intro>
- Google Maps Platform. (2020b). Developer Guide - Distance Matrix API. Retrieved from <https://developers.google.com/maps/documentation/distance-matrix/intro>
- Google Maps Platform. (2020c). Google Maps Platform Billing. Retrieved from <https://developers.google.com/maps/billing/gmp-billing>
- Hemedinger, C. (2016). Reading data with the SAS JSON libname engine. *SAS Blogs*. Retrieved from <https://blogs.sas.com/content/sasdummys/2016/12/02/>
- "JSON." (n.d.). Introducing JSON. Retrieved from <https://www.json.org/json-en.html>
- National Land Numerical Information download service. (2014). Local Government Office Data Version 1.0. Retrieved from <http://nlftp.mlit.go.jp/ksj-e/gml/datalist/KsjTmplt-P34.html>
- Odom, E. (2018). SGMAP: Using MAPIMPORT for Polygon Maps. *SAS Blogs*. Retrieved from <https://blogs.sas.com/content/graphicallyspeaking/2018/02/23/>

Oxford University Press. (Ed.) (n.d.). Oxford Learner's Dictionaries. Retrieved February 27, 2020, Retrieved from <https://www.oxfordlearnersdictionaries.com/>

Roy, A., & Na, Y. (2012). Batch Production of Driving Distances and Times Using SAS® and Web Map APIs. *Paper presented at the SAS Global Forum 2012*, Orlando, FL, US. Retrieved from <http://support.sas.com/resources/papers/proceedings12/091-2012.pdf>

Sakaguchi, S., Oda, M., Shinkoda, Y., & Manabe, A. (2014). Parents' perception of pediatric cancer centers in Japan. *Pediatrics International*, *56*(2), 196-199. doi:10.1111/ped.12246

SAS Institute Inc. (2016). URLENCODE Function. In *SAS® 9.4 Functions and CALL Routines: Reference* (5th ed., pp. 1341-1342). Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2017). LIBNAME Statement: JSON Engine In *SAS® 9.4 Global Statements: Reference* (pp. 149-167). Cary, NC: SAS Institute Inc.

Shaw, B. I., Wangara, A. A., Wambua, G. M., Kiruja, J., Dicker, R. A., Mweu, J. M., & Juillard, C. (2017). Geospatial relationship of road traffic crashes and healthcare facilities with trauma surgical capabilities in Nairobi, Kenya: defining gaps in coverage. *Trauma surgery & acute care open*, *2*:e000130. doi:10.1136/tsaco-2017-000130

Wang, F., & Xu, Y. (2011). Estimating O-D Travel Time Matrix by Google Maps API: Implementation, Advantages, and Implications. *Annals of GIS*, *17*, 199-209. doi:10.1080/19475683.2011.625977

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Mr. Yutaka Morioka in the EPS Corporation for his invaluable support and kindness during the call for content to the SAS Global Forum 2020.

The basic idea used in the paper was presented at the Japan SAS Users Forum in Tokyo in September 2019.

CONTACT INFORMATION

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

Anna Tsutsui
Osaka University
anna.tsutsui@sahs.med.osaka-u.ac.jp / anna.tsutsui.114@gmail.com

APPENDIX

Appendix 1 Edited JSON map file

```
{
  "DATASETS": [
    {
      "DSNAME": "ALL_API",
      "TABLEPATH": "/root",
      "VARIABLES": [
        {
          "NAME": "destination_addresses",
          "TYPE": "CHARACTER",
          "PATH": "/root/destination_addresses/destination_addresses1",
          "CURRENT_LENGTH": 45
        },
        {
          "NAME": "origin_addresses",
          "TYPE": "CHARACTER",
          "PATH": "/root/origin_addresses/origin_addresses1",
          "CURRENT_LENGTH": 50
        },
        {
          "NAME": "distanceC",
          "TYPE": "CHARACTER",
          "PATH": "/root/rows/elements/distance/text",
          "CURRENT_LENGTH": 6
        },
        {
          "NAME": "distanceN",
          "TYPE": "NUMERIC",
          "PATH": "/root/rows/elements/distance/value"
        },
        {
          "NAME": "durationC",
          "TYPE": "CHARACTER",
          "PATH": "/root/rows/elements/duration/text",
          "CURRENT_LENGTH": 15
        },
        {
          "NAME": "durationN",
          "TYPE": "NUMERIC",
          "PATH": "/root/rows/elements/duration/value"
        },
        {
          "NAME": "status",
          "TYPE": "CHARACTER",
          "PATH": "/root/rows/elements/status",
          "CURRENT_LENGTH": 2
        },
        {
          "NAME": "status2",
          "TYPE": "CHARACTER",
          "PATH": "/root/status",
          "CURRENT_LENGTH": 2
        }
      ]
    }
  ]
}
```

```
}
```

Appendix 2 SAS Code to utilize the Google Distance Matrix API

```
* == Settings == ;

options urlencoding=utf8 ;
%let _key=<API_KEY> ;
%let _destinations=%qsysfunc(urlencode(Hokkaido University Hospital)) ;
%let _sec=%sysfunc(sum("05FEB2020T10:00:00"dt, - 315619200)) ;
filename JMAP "./Program/jmap.map" ;

* == %DO loop == ;

data _SHP ;
  set RAW.SHP(where=(P34_002="1" and P34_001^="01100")) ;
  _SEQ+1 ;
  call symput("_N", cats(put(_SEQ, 8.))) ;
run ;

%macro UTILIZE_API ;

  data _ALL ;
    length destination_addresses origin_addresses $200
           distanceC durationC $40 status status2 $10 ;
    call missing(of _ALL_) ;
    stop ;
  run ;

  %do i=1 %to &_N. ;

    data _NULL_ ;
      set _SHP ;
      where _SEQ=&i. ;
      call symput("_origins_tmp", cats(P34_004)) ;
    run ;
    %let _origins=%qsysfunc(urlencode(&_origins_tmp)) ;

    filename MAP_API url
    "https://maps.googleapis.com/maps/api/distancematrix/json?key=&_ke
    y%str(&)origins=&_origins.%str(&)destinations=&_destinations.%str(
    &)arrival_time=&_sec." encoding="utf8" debug ;
    libname MAP_API json map=JMAP automap=reuse ;

    data _MRG ;
      merge _SHP(where=( _SEQ=&i.)) MAP_API.ALL_API ;
    run ;

    data _ALL ;
      set _ALL _MRG ;
    run ;

  %end ;

%mend ;

%UTILIZE_API ;
```