The Huff Model in QGIS: A Goodlife Fitness Case Study

There is interest in developing an open source resource or tool to solve Huff Model problems. It has been suggested that the capabilities of QGIS (geographic information software) could be extended with the use of a plugin written in the Python programming language. To assist in the understanding of the tasks required for such a project, a hypothetical case study is used. This approach allows for the creation of a dataset that can be employed to explore a project of this nature. Goodlife Fitness is the focus of the case study and their co-ed clubs in the Toronto census metropolitan area (CMA) have been used to explore the existing capabilities of standard spreadsheet applications and a standard installation of QGIS 2.14-Essen. Additional capabilities that would be required of a new plugin are also considered.

The Huff Model may be implemented in one of several ways depending on the user's needs and the data available. The two key inputs into the Huff Model are a measure of distance and a measure of utility. A measure of distance may be based on one of two methods: euclidean or network distance. Euclidean distance is “as the crow flies” (based on a straight line) while network distance is based on some existing network of paths such as a road network, river system, or electrical cabling. A measure of utility may be based on size (such as square footage), number of employees, temperature, ratings, or any other measure of attractiveness.

The intent of this work is to develop a QGIS plugin that provides the ability for the user to select the implementation of the Huff Model that meets their needs and matches to the data they have available. This case study is used to explore both the preparation of the required datasets and the development of the required Python code needed to extend QGIS with the ability to complete Huff Model analyses.

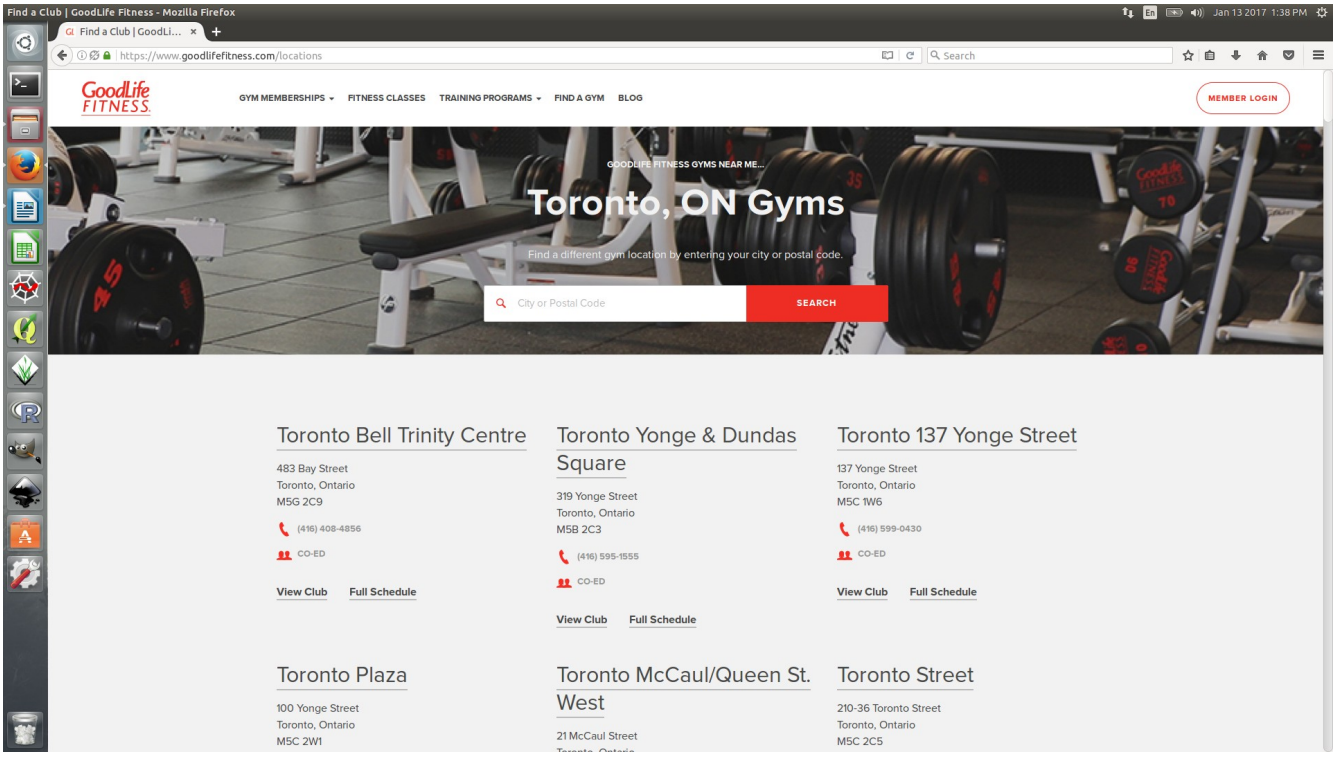
Part I: Data Preparation

Part I deals with the preparation of three core datasets for use with the Huff Model and potential plugin:

1. Point-based data for Goodlife Fitness locations in the Toronto CMA,
2. Regional data for the Toronto CMA's census tracts,
3. Road network data for the Toronto CMA.

Step 1: Getting Goodlife Fitness Data

To make use of the Huff Model with QGIS, data is required. Depending on the particular situation, data may not be readily available and may need to be collected before the Huff Model may be used. To start exploring Goodlife Fitness, data has been manually collected from the Goodlife Fitness web site at http://www.goodlifefitness.com/ , as presented below, and entered/pasted into a spreadsheet. Note that the web site expects location sharing and this may need to be turned off in the browser to show the list of locations. On a desktop/notebook computer, GPS is not usually available and a pop-up in the browser may appear. Selecting the option not to share location forces the presentation of the list of all locations in the general area. If no results are offered under the search field, location sharing is likely the issue. Note the crossed-out site symbol beside the lock in the address bar of the web browser below.



Data collected for each co-ed club located in the Toronto CMA includes: location name, street address, city, province, and postal code. An additional field has been created to hold an arbitrary unique identifier “GFID”. The values for GFID are simply “GF” plus a number starting at 1001. This is done

to force the values in this field to be both unique and of string data type. This is important later in QGIS. The spreadsheet has been constructed as presented in the screen capture below. A total of 60 locations have been recorded for Goodlife Fitness locations within the Toronto CMA.

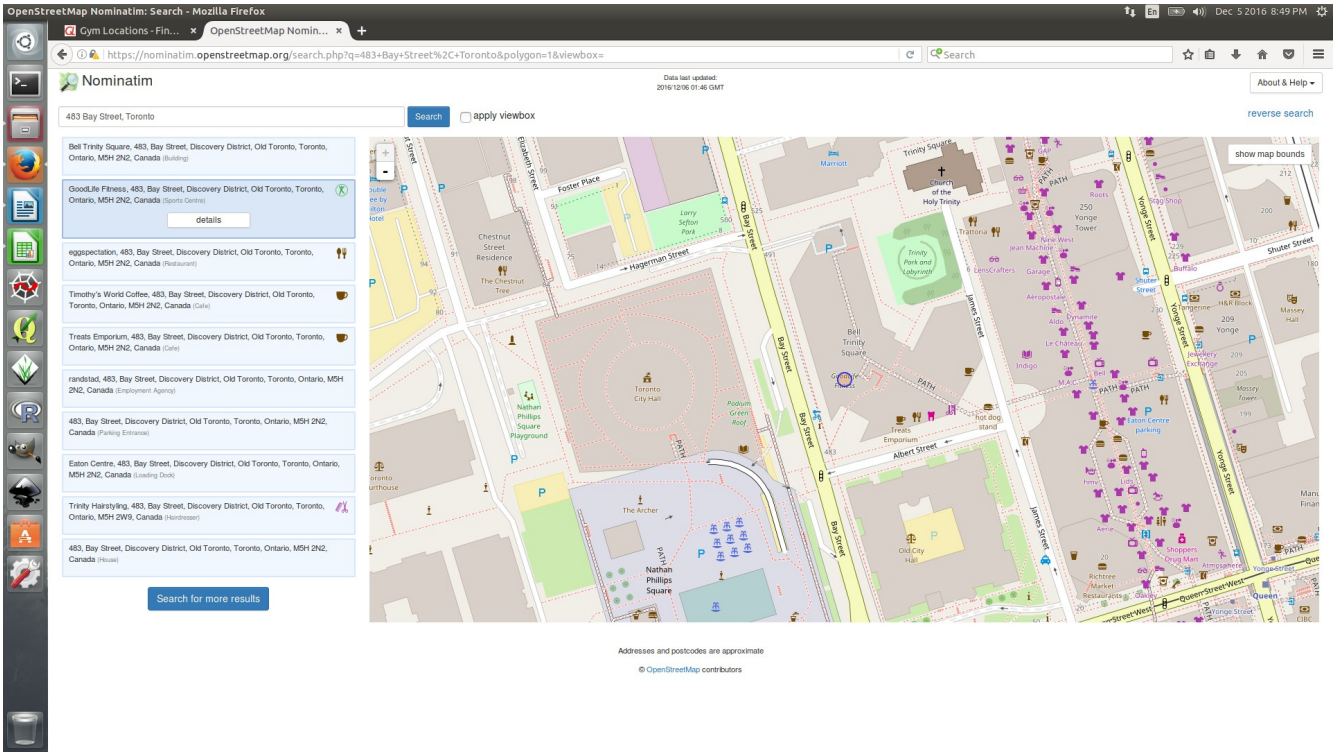
	A	B	C	D	E	F
1	GFID	Name	Address	City	Province	PostalCode
2	GF1001	Toronto Bell Trinity Centre	483 Bay Street	Toronto	Ontario	M5G2C9
3	GF1002	Toronto 137 Yonge Street	137 Yonge Street	Toronto	Ontario	M5C1W6
4	GF1003	Toronto Yonge & Dundas Square	319 Yonge Street	Toronto	Ontario	M5B2C3
5	GF1004	Toronto Plaza	100 Yonge Street	Toronto	Ontario	M5C2W1
6	GF1005	Toronto McCaul/Queen St. West	21 McCaul Street	Toronto	Ontario	M5T1V7
7	GF1006	Toronto Wellington West	111 Wellington St. West	Toronto	Ontario	M5J2S6
8	GF1007	Toronto Richmond John	267 Richmond Street West	Toronto	Ontario	M5V3M6
9	GF1008	Toronto Union Station	7 Station Street	Toronto	Ontario	M5J1C3
10	GF1009	Toronto Richmond/Bathurst	555 Richmond Street West	Toronto	Ontario	M5V3B1
11	GF1010	Toronto Manulife Centre	210 - 55 Bloor St. W.	Toronto	Ontario	M4W1A5
12	GF1011	Toronto Bloor Yorkville	80 Bloor Street West	Toronto	Ontario	M5S2V1
13	GF1012	Toronto Bloor Park	8 Park Road	Toronto	Ontario	M4W3G8
14	GF1013	Toronto College Street	533 College Street	Toronto	Ontario	M6G1A8
15	GF1014	Toronto King/Liberty	85 Hanna Ave. Suite 200	Toronto	Ontario	M6K3S3

It should also be noted that Goodlife Fitness has not been consistent with the format of their addresses. In some cases units or suites come after the street name and in other cases they come before. Abbreviations are also used inconsistently. This formatting may create issues if the process of identifying the associated latitude and longitude for each address is automated. The use of the old borough names and some punctuation used may also be problematic and may need to be corrected.

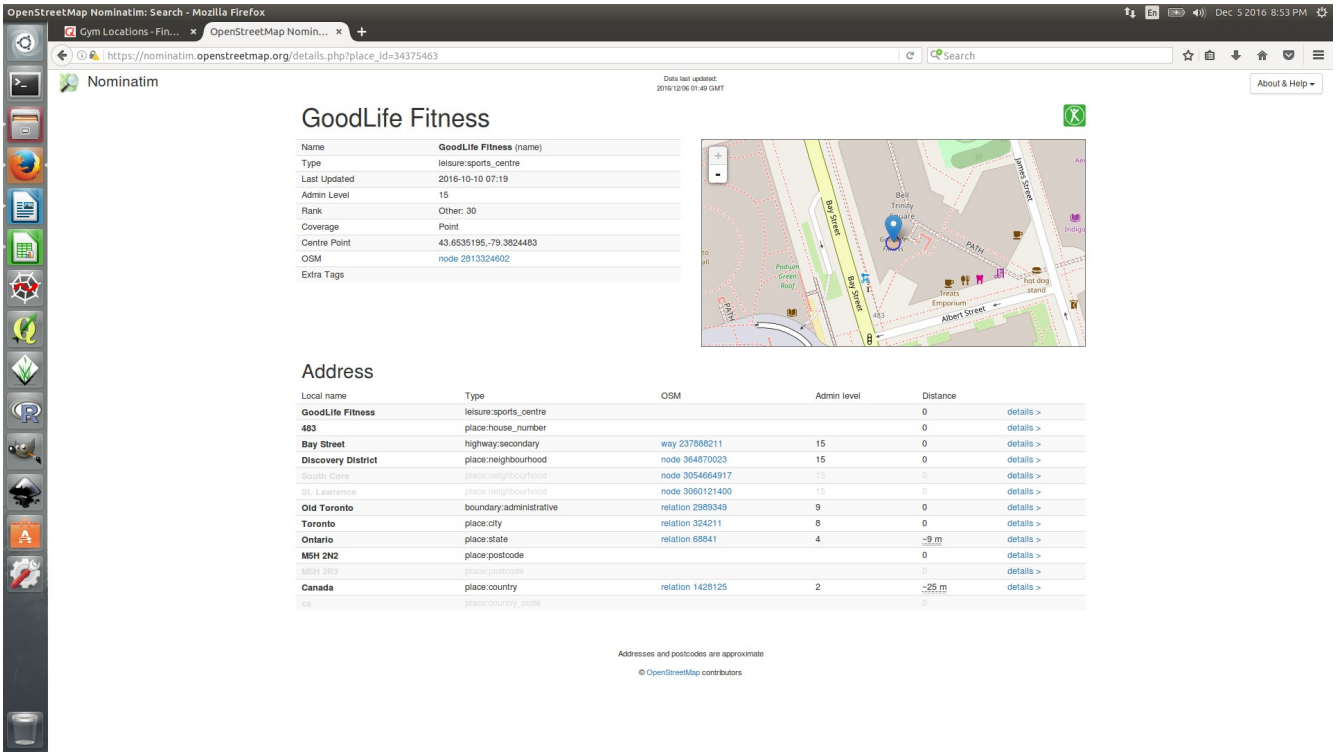
Step 2: Getting Latitude and Longitude Values for Goodlife Fitness Addresses

Geocoding addresses has recently become more challenging as services that have been freely available in the past have evolved into fee-based services. To find an open source, free, and efficient method of geocoding addresses is a problem all by itself. As the number of addresses in projects using the Huff Model are relatively small, the latitude and longitude for each address have been manually sourced and added to the current spreadsheet. Two new columns were created, “Lat” and “Lon”, and then the cells in these columns were populated with the appropriate values.

To obtain the latitude and longitude coordinates for each address, OpenStreetMap's Nominatim geocoding service was used. See: https://nominatim.openstreetmap.org/ . Searching for the first address, “483 Bay Street, Toronto”, found the first Goodlife Fitness location. See the screen capture below.

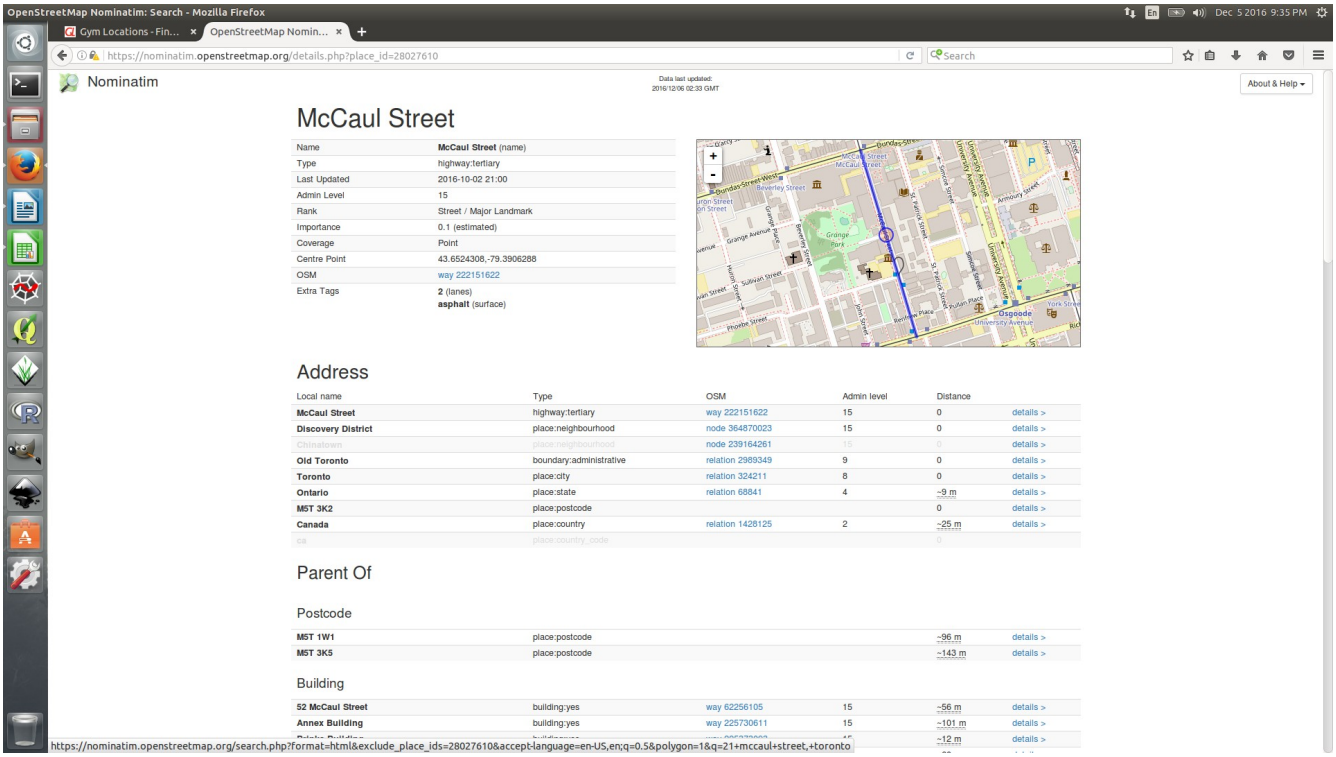


Clicking on the “details” button for the Goodlife Fitness result presents a second page with location details including the latitude and longitude. See the next screen capture.

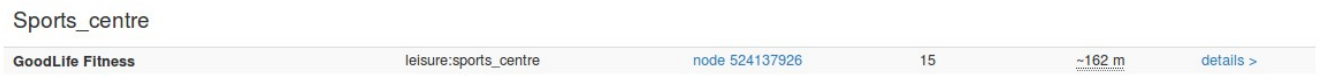


It should be noted that Nominatim seems to get the postal codes wrong and using postal codes in

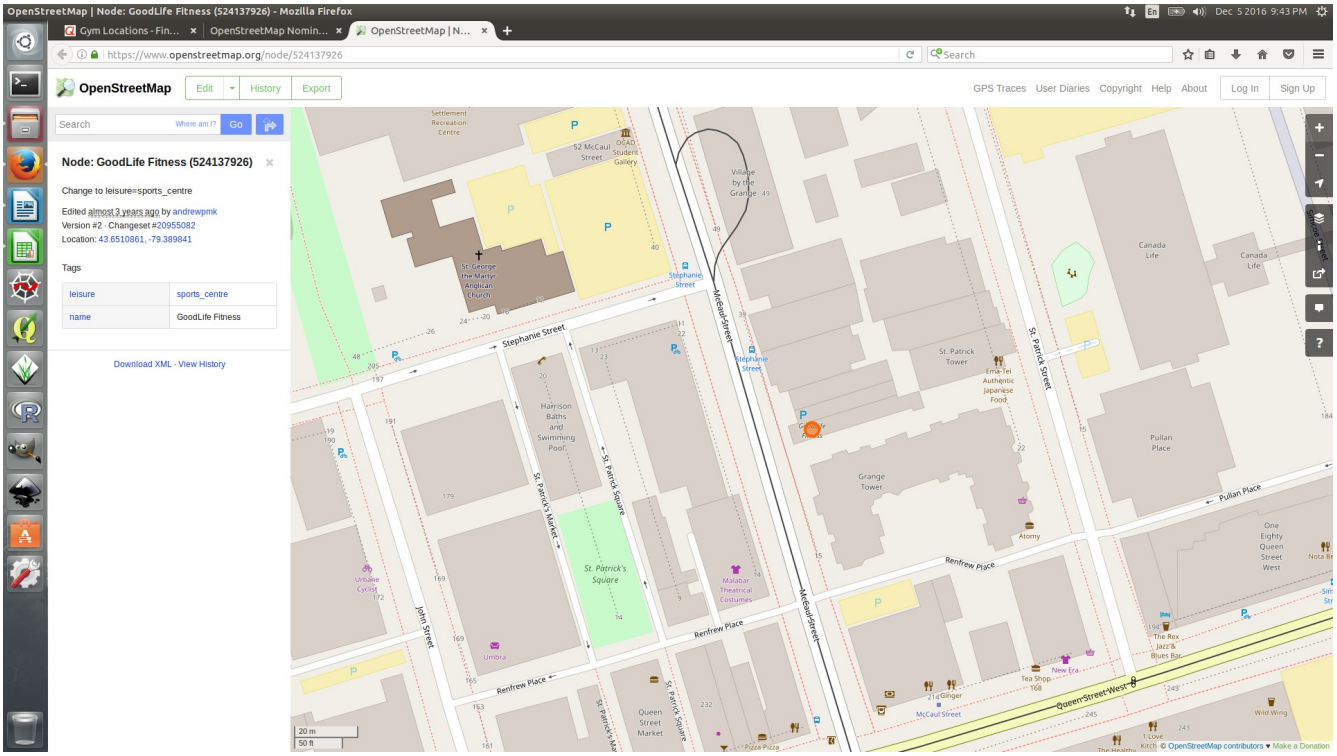
queries may not be useful; a street address and city seems to work best. The “Centre Point” item in the upper table presents the latitude and longitude values. While the example in the screen capture above is pretty accurate, not all are. Some results use a centre point of the line segment for the street that the point of interest is located on. See the screen capture below for an example of how the location on McCaul Street in Toronto is presented in this way. Note the blue line between Queen Street West and Dundas Street West. The circle in the middle of the line represents the centre point offered for this address.



A better result may be offered by scrolling down the page. If the Goodlife Fitness location has been recorded by OpenStreetMap, it may have a more accurate location. See the next screen capture slice.



This Goodlife Fitness location is listed and additional information is available by clicking the blue “node” link. This opens a page on openstreetmap.org with more accurate location information. See the following screen capture and note the orange dot marking the Goodlife Fitness location.



With the addition of the latitude and longitude values, the spreadsheet evolved as presented in the screen capture below.

	A	B	C	D	E	F	G	H
1	GFID	Name	Address	City	Province	PostalCode	Lat	Lon
2	GF1001	Toronto Bell Trinity Centre	483 Bay Street	Toronto	Ontario	M5G2C9	43.6535195	-79.3824483
3	GF1002	Toronto 137 Yonge Street	137 Yonge Street	Toronto	Ontario	M5C1W6	43.6511	-79.3784479
4	GF1003	Toronto Yonge & Dundas Square	319 Yonge Street	Toronto	Ontario	M5B2C3	43.6579071	-79.3815849
5	GF1004	Toronto Plaza	100 Yonge Street	Toronto	Ontario	M5C2W1	43.64924	-79.3780769
6	GF1005	Toronto McCaul/Queen St. West	21 McCaul Street	Toronto	Ontario	M5T1V7	43.6510861	-79.389841
7	GF1006	Toronto Wellington West	111 Wellington St. West	Toronto	Ontario	M5J2S6	43.6464153	-79.3831754
8	GF1007	Toronto Richmond John	267 Richmond Street West	Toronto	Ontario	M5V3M6	43.648791	-79.3918819
9	GF1008	Toronto Union Station	7 Station Street	Toronto	Ontario	M5J1C3	43.6445604	-79.3835105
10	GF1009	Toronto Richmond/Bathurst	555 Richmond Street West	Toronto	Ontario	M5V3B1	43.646697	-79.4020449
11	GF1010	Toronto Manulife Centre	210 - 55 Bloor St. W.	Toronto	Ontario	M4W1A5	43.6696112	-79.3886006
12	GF1011	Toronto Bloor Yorkville	80 Bloor Street West	Toronto	Ontario	M5S2V1	43.66992405	-79.3901913276
13	GF1012	Toronto Bloor Park	8 Park Road	Toronto	Ontario	M4W3G8	43.6714601	-79.3847340999
14	GF1013	Toronto College Street	533 College Street	Toronto	Ontario	M6G1A8	43.655568	-79.4111385
15	GF1014	Toronto King/Liberty	85 Hanna Ave. Suite 200	Toronto	Ontario	M6K3S3	43.6401067	-79.4197758
16	GF1015	Toronto St. Clair/Yonge	12 St. Clair Avenue East	Toronto	Ontario	M4T1L7	43.6886189	-79.3936835
17	GF1016	Toronto Danforth Pape	635 Danforth Ave.	Toronto	Ontario	M4K1R2	43.6786807	-79.3453936
18	GF1017	Toronto Mount Pleasant/Davisville	250 Davisville Avenue	Toronto	Ontario	M4S2L9	43.700477	-79.3865509
19	GF1018	Toronto Coxwell/Gerrard	280 Coxwell Avenue	Toronto	Ontario	M4L3B6	43.673324	-79.319625
20	GF1019	Toronto Yonge & Eglinton	2300 Yonge Street , Yonge Eglinton Centre	Toronto	Ontario	M4P1E4	43.7067505	-79.399362

With all of the location data collected, only the “size” or attractiveness data used by the Huff Model formula needed to be provided for and this was done next.

Step 3: Generating Data to Represent a Goodlife Fitness Location's Size / Attractiveness

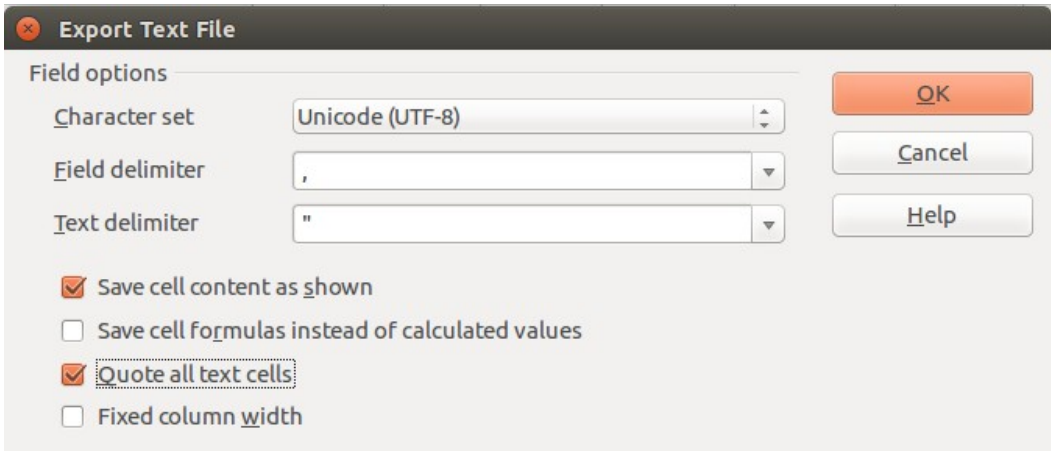
Without having actual data for each Goodlife Fitness location, data needed to be randomly generated for the purposes of this case study. To provide guidance to this task, a Toronto Star article was leveraged, see: https://www.thestar.com/business/2015/01/20/fitness-chain-interested-in-target-properties.html . This article states that typical locations are about 24,000 square feet in size. With this

in mind club sizes have been randomly generated by a spreadsheet formula to be within 15,000 and 35,000 square feet (some are larger). The resulting formula for LibreOffice Calc is "RANDBETWEEN(15000,35000)". This new data appears as the last column "SizeSqFt" in the spreadsheet. The completed spreadsheet is presented in the screen capture below.

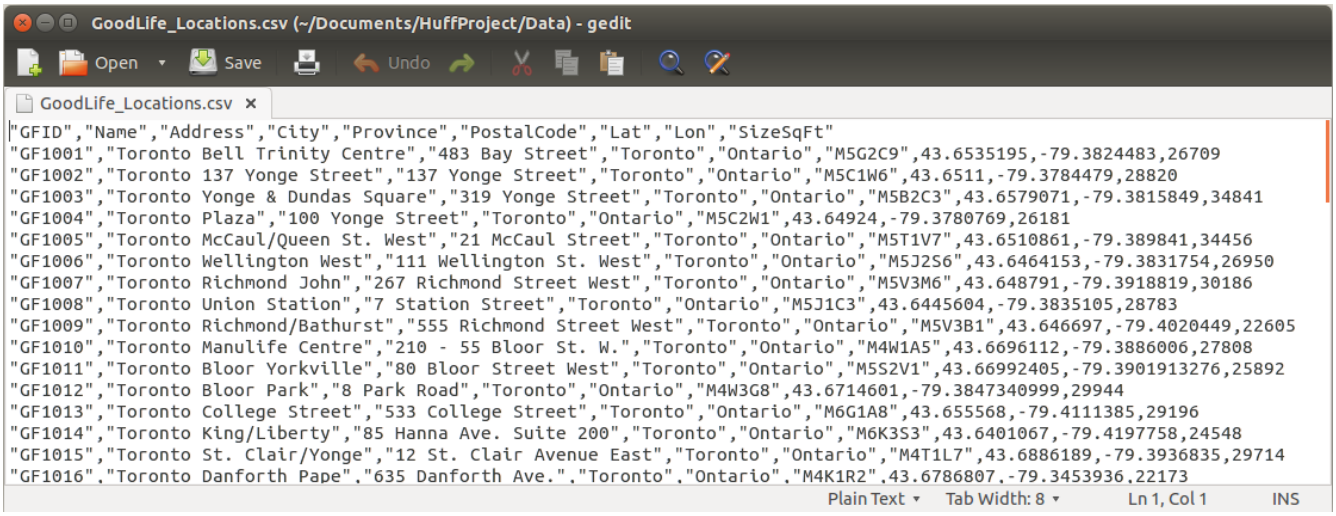
	A	B	C	D	E	F	G	H	I
1	GFID	Name	Address	City	Province	PostalCode	Lat	Lon	SizeSqFt
2	GF1001	Toronto Bell Trinity Centre	483 Bay Street	Toronto	Ontario	M5G2C9	43.6535195	-79.3824483	26709
3	GF1002	Toronto 137 Yonge Street	137 Yonge Street	Toronto	Ontario	M5C1W6	43.6511	-79.3784479	28820
4	GF1003	Toronto Yonge & Dundas Square	319 Yonge Street	Toronto	Ontario	M5B2C3	43.6579071	-79.3815849	34841
5	GF1004	Toronto Plaza	100 Yonge Street	Toronto	Ontario	M5C2W1	43.64924	-79.3780769	26181
6	GF1005	Toronto McCaul/Queen St. West	21 McCaul Street	Toronto	Ontario	M5T1V7	43.6510861	-79.389841	34456
7	GF1006	Toronto Wellington West	111 Wellington St. West	Toronto	Ontario	M5J2S6	43.6464153	-79.3831754	26950
8	GF1007	Toronto Richmond John	267 Richmond Street West	Toronto	Ontario	M5V3M6	43.648791	-79.3918819	30186
9	GF1008	Toronto Union Station	7 Station Street	Toronto	Ontario	M5J1C3	43.6445604	-79.3835105	28783
10	GF1009	Toronto Richmond/Bathurst	555 Richmond Street West	Toronto	Ontario	M5V3B1	43.646697	-79.4020449	22605
11	GF1010	Toronto Manulife Centre	210 - 55 Bloor St. W.	Toronto	Ontario	M4W1A5	43.6696112	-79.3886006	27808
12	GF1011	Toronto Bloor Yorkville	80 Bloor Street West	Toronto	Ontario	M5S2V1	43.66992405	-79.3901913276	25892
13	GF1012	Toronto Bloor Park	8 Park Road	Toronto	Ontario	M4W3G8	43.6714601	-79.3847340999	29944
14	GF1013	Toronto College Street	533 College Street	Toronto	Ontario	M6G1A8	43.655568	-79.4111385	29196
15	GF1014	Toronto King/Liberty	85 Hanna Ave. Suite 200	Toronto	Ontario	M6K3S3	43.6401067	-79.4197758	24548
16	GF1015	Toronto St. Clair/Yonge	12 St. Clair Avenue East	Toronto	Ontario	M4T1L7	43.6886189	-79.3936835	29714
17	GF1016	Toronto Danforth Pape	635 Danforth Ave.	Toronto	Ontario	M4K1R2	43.6786807	-79.3453936	22173
18	GF1017	Toronto Mount Pleasant/Davisville	250 Davisville Avenue	Toronto	Ontario	M4S2L9	43.700477	-79.3865509	31587
19	GF1018	Toronto Coxwell/Gerrard	280 Coxwell Avenue	Toronto	Ontario	M4L3B6	43.673324	-79.319625	25485
20	GF1019	Toronto Yonge & Eglinton	2300 Yonge Street , Yonge Eglinton Centre	Toronto	Ontario	M4P1E4	43.7067505	-79.399362	26724
21	GF1020	Toronto Dunfield	110 Eglinton Ave. East	Toronto	Ontario	M4P1A6	43.7075676	-79.3952713	21273
22	GF1021	North York Parkview	250 Ferrand Drive	Toronto	Ontario	M3C3G8	43.7195101	-79.3319053	30079
23	GF1022	North York York Mills	4025 Yonge Street	North York	Ontario	M2P2E3	43.7445107	-79.4060282	32926
24	GF1023	Etobicoke King's Mill	3280 Bloor Street West , Centre Tower 2nd Floor, Suite Mez 1	Etobicoke	Ontario	M8X2X3	43.6357244	-79.5622346	25042
25	GF1024	North York Madison Centre	4950 Yonge Street	North York	Ontario	M2N6K1	43.7648919	-79.4119615	23681

Step 4: Creating a Comma-Separated Values (.csv) File for the Goodlife Fitness Data

Data that has been prepared for the Goodlife Fitness locations needed to be put into a format that can be imported into QGIS. A common file format that can be used for this task is the comma-separated values (.csv) format that reduces the data to a plain-text file. This was achieved by using the "Save As..." command and saving the file as a different file type. Note that some additional specifications are usually required; a delimiter needs to be specified (the "comma" in "comma-separated values") and quotes ("") to indicate text values. All text should also be in quotes. Some of the data in the addresses has multiple parts and is already separated by commas and this ensures it is compartmentalized properly.



The result of the file export can be viewed in a common text editor and the new file is presented below. Note the quotation marks on the text fields and that the first row includes the column header values.

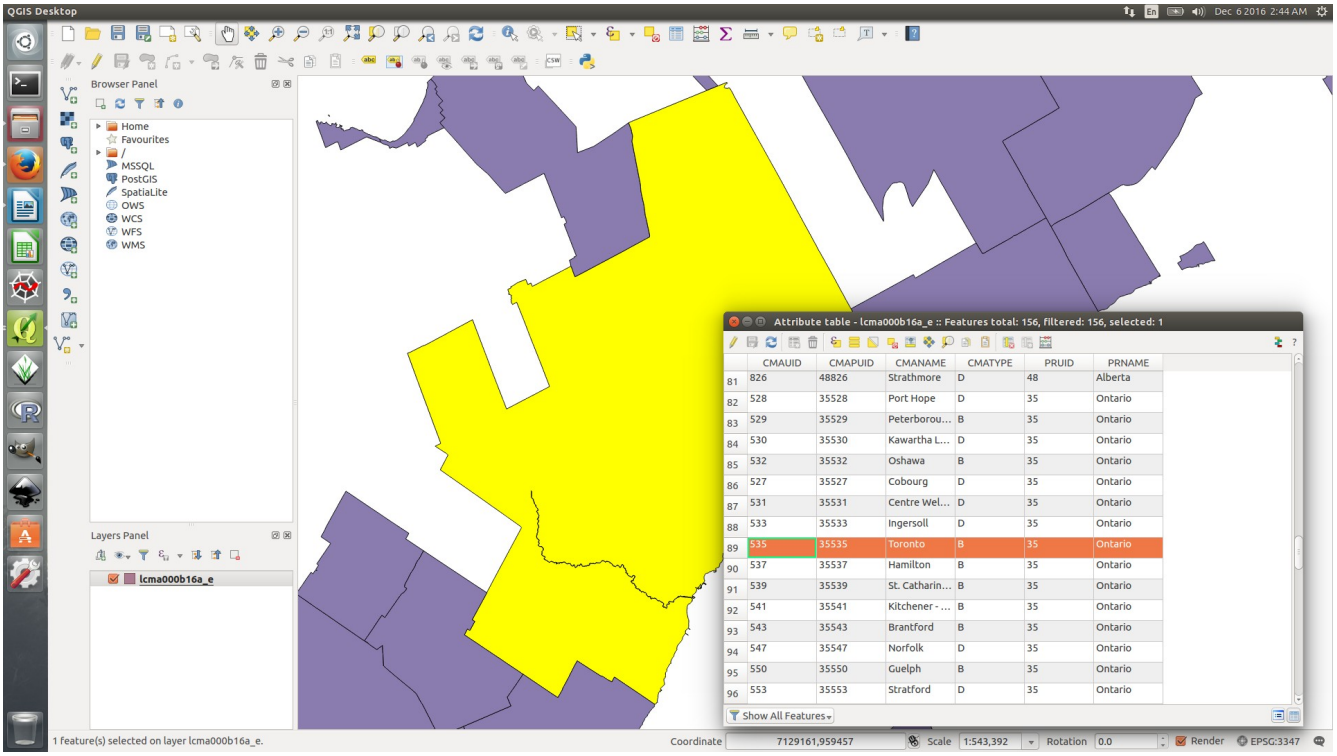


Step 5: Collecting Census Boundary Data Files from Statistics Canada

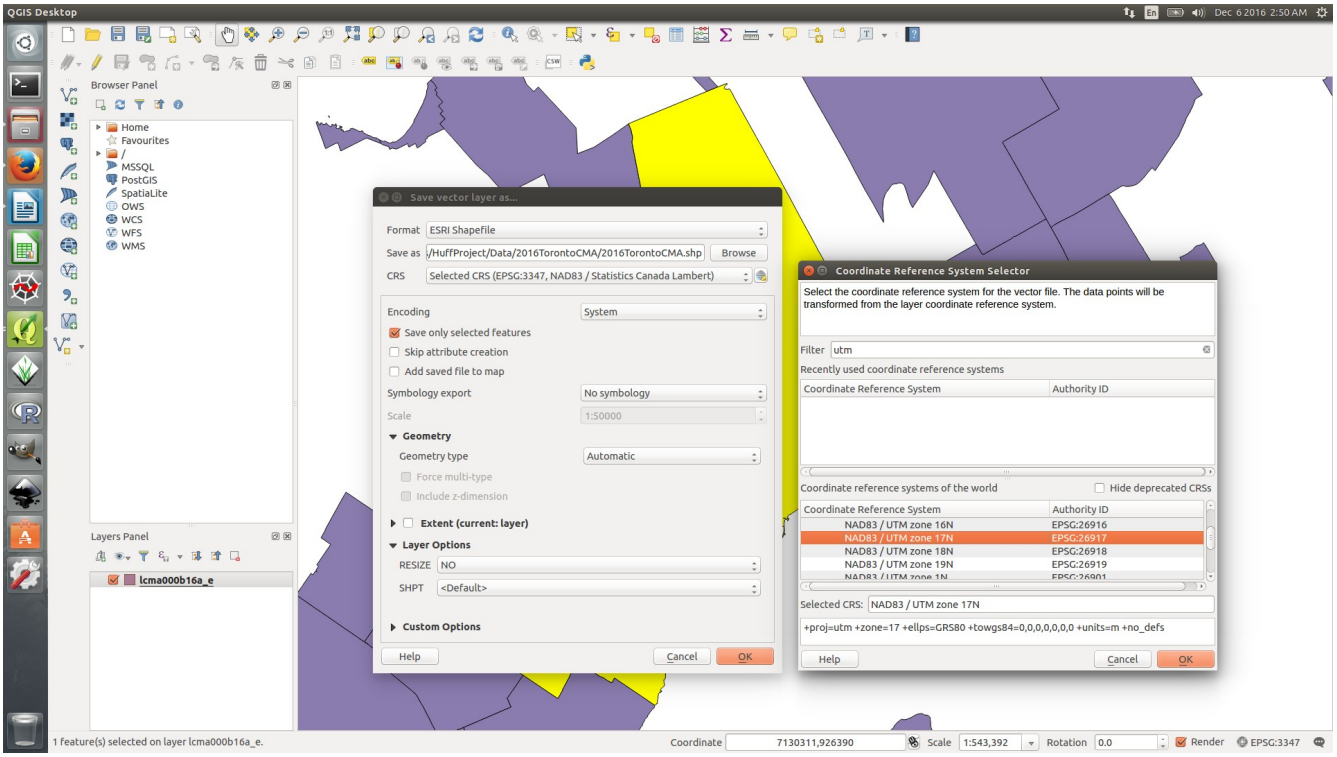
In addition to data for Goodlife Fitness locations, data was also required for the study area. In this case, the Toronto CMA has been specified as the study area and census tracts have been used for the regions for which Huff probabilities are calculated. Geographic boundary files are required and they are available from Statistics Canada. 2016 boundary files are available at this location: https://www12.statcan.gc.ca/census-recensement/2011/geo/bound-limit/bound-limit-2016-eng.cfm . The data files provided by Statistics Canada include regions for all of Canada. To work with only the Toronto CMA (the study area), the data of interest needed to be extracted. This was done in QGIS and is described below. Boundary files have been obtained for both CMA and census tract level regions.

Step 6: Extracting Regional Data from Statistics Canada Shapefiles

In QGIS, the CMA boundary file has been opened and the Toronto CMA has been selected in the attribute table as presented in the first screen capture below. The selected CMA has then been exported as a new shapefile as shown in the second screen capture. The process also allowed for the new file to be saved with a more appropriate coordinate reference system for the study area.

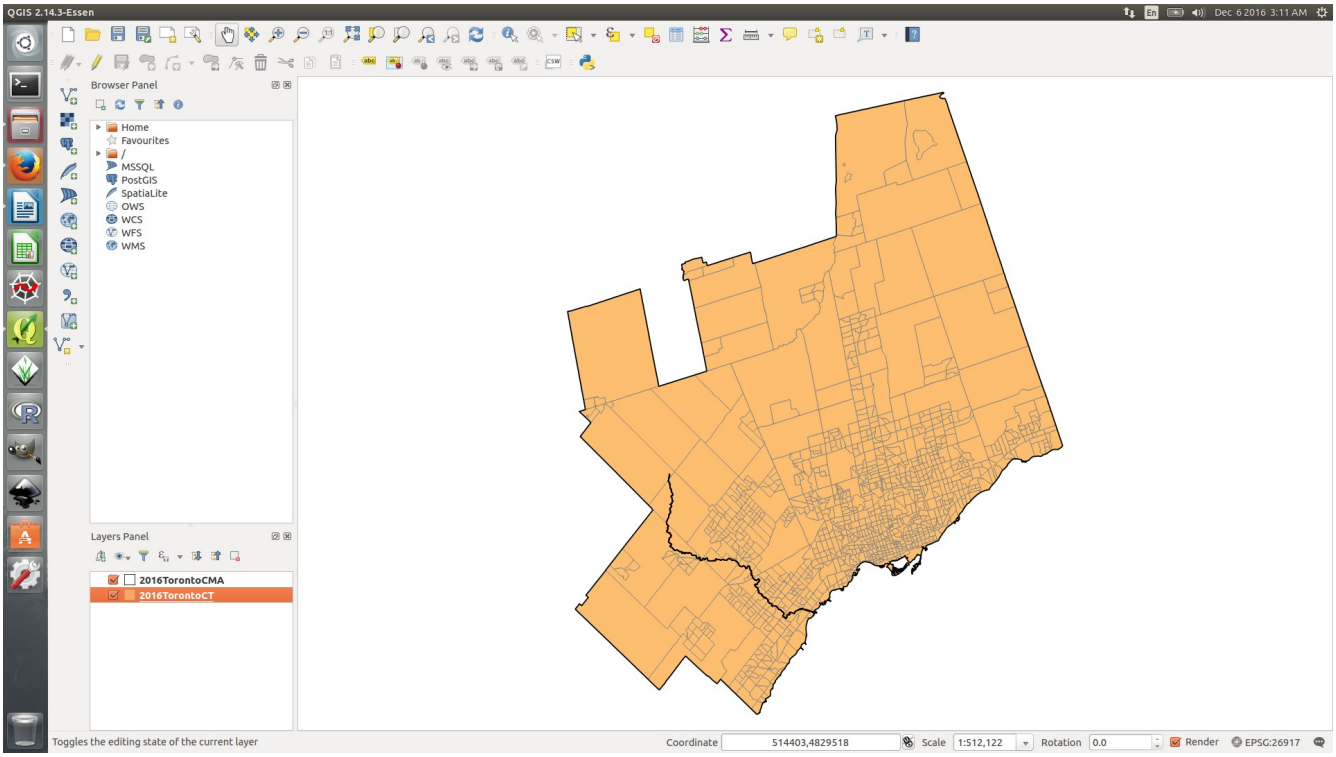


The export process appears in the next screen capture.



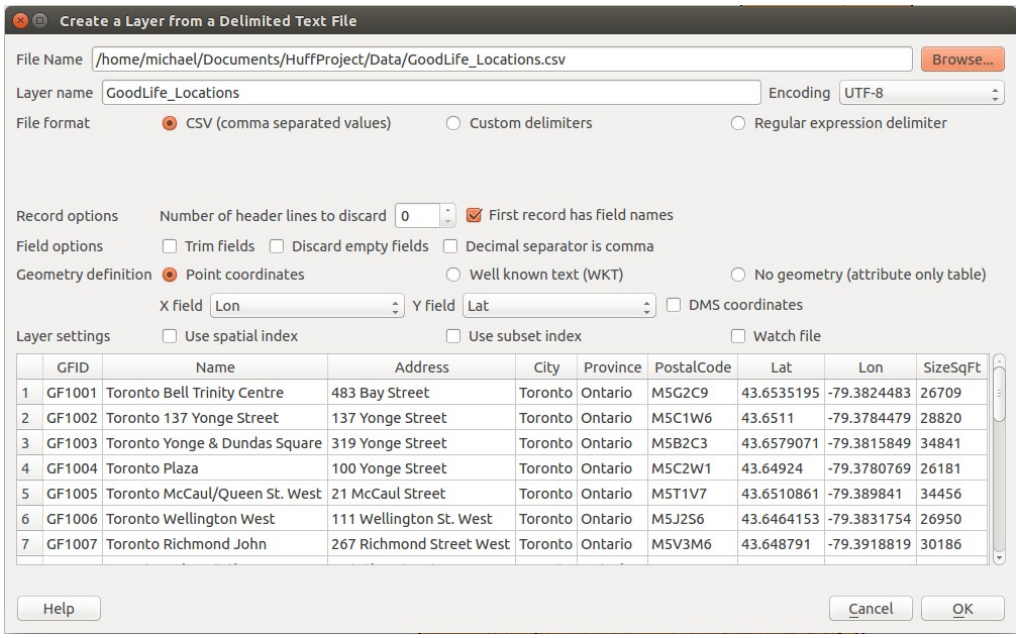
The process was repeated for the census tract files.

The result is two new sets of data files that match the study area. The census tract files will do the "work" while the CMA files have been used to create a presentation boundary. Screen capture below.



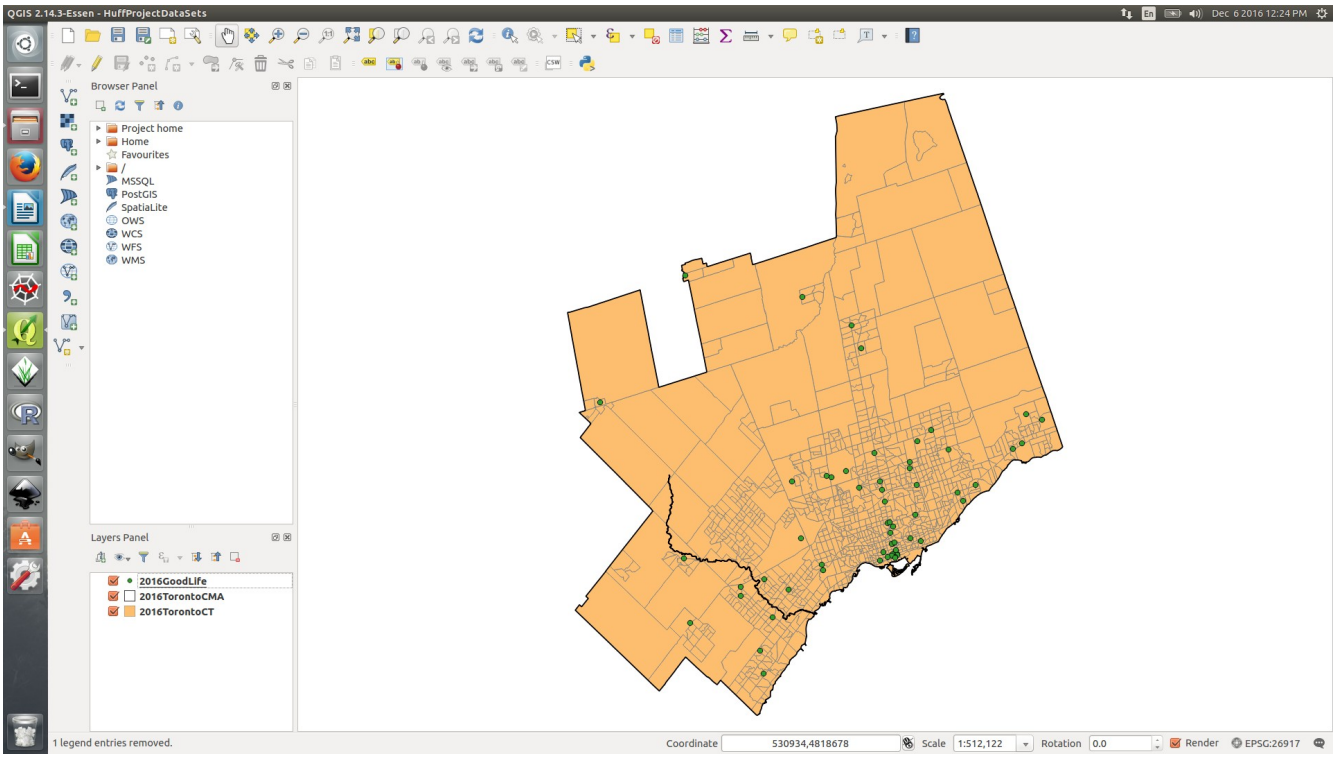
Step 7: Creating the Point Layer for Goodlife Fitness Locations

QGIS provides functionality that makes adding data stored in comma-separated value (.csv) files easy and straightforward. The next screen capture illustrates this with the required import settings.



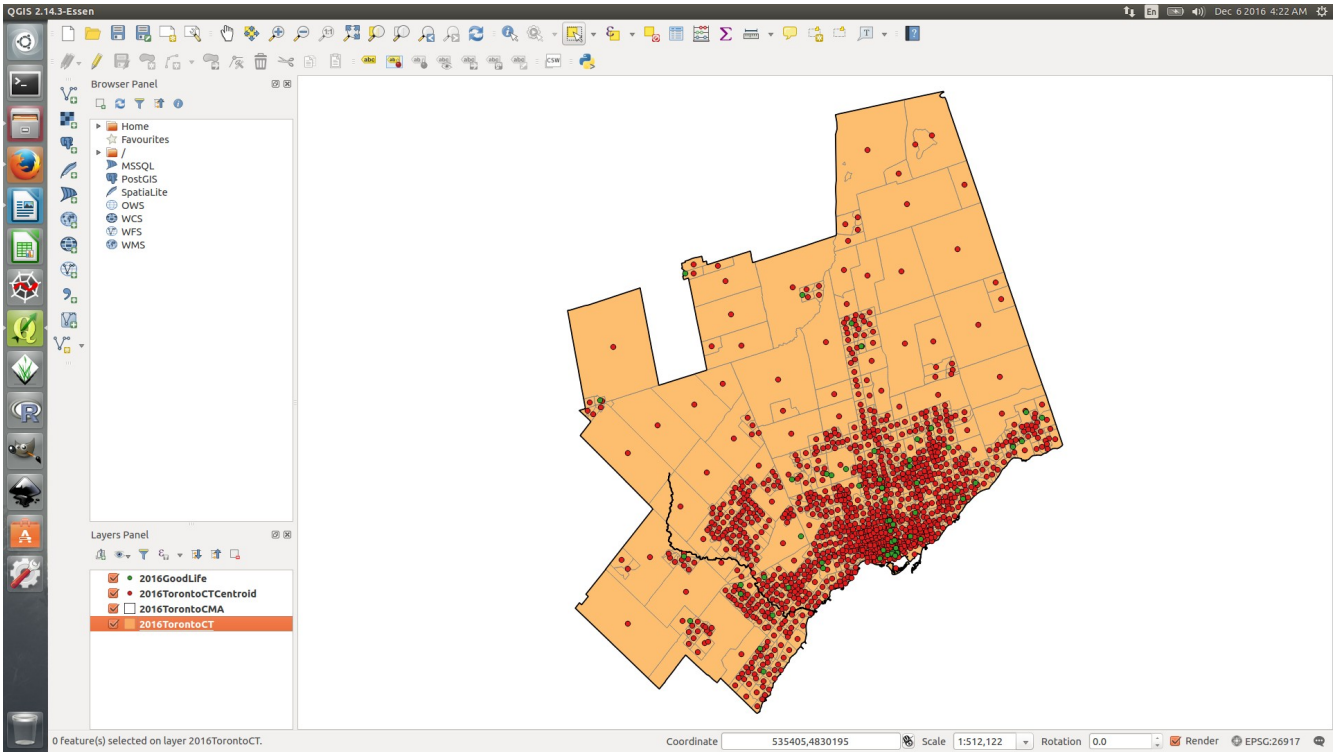
	GFID	Name	Address	City	Province	PostalCode	Lat	Lon	SizeSqFt
1	GF1001	Toronto Bell Trinity Centre	483 Bay Street	Toronto	Ontario	M5G2C9	43.6535195	-79.3824483	26709
2	GF1002	Toronto 137 Yonge Street	137 Yonge Street	Toronto	Ontario	M5C1W6	43.6511	-79.3784479	28820
3	GF1003	Toronto Yonge & Dundas Square	319 Yonge Street	Toronto	Ontario	M5B2C3	43.6579071	-79.3815849	34841
4	GF1004	Toronto Plaza	100 Yonge Street	Toronto	Ontario	M5C2W1	43.64924	-79.3780769	26181
5	GF1005	Toronto McCaul/Queen St. West	21 McCaul Street	Toronto	Ontario	M5T1V7	43.6510861	-79.389841	34456
6	GF1006	Toronto Wellington West	111 Wellington St. West	Toronto	Ontario	M5J2S6	43.6464153	-79.3831754	26950
7	GF1007	Toronto Richmond John	267 Richmond Street West	Toronto	Ontario	M5V3M6	43.648791	-79.3918819	30186

A second screen completes the process and allows for the specification of the coordinate reference system to be used for the data import. This should be “WGS 84”. Once QGIS has created the point layer, it needs to be saved. As part of the step of saving the new file, the coordinate reference system has been changed to match the other data files. This process has been used to import the Goodlife Fitness location data from the comma-separated values (.csv) file created in Step 4. The three data layers are presented together in a new project in QGIS in the screen capture below.



Step 8: Creating Centroids from the Census Tract Layer

The Huff Model requires that distances are calculated between each region (consumers) and each target location (centres). In this case study, census tracts are the regions and Goodlife Fitness locations are the target locations. To specify a distance as required, a single point needs to be used for each census tract and the census tract's centroid has been selected for this purpose. QGIS can prepare new data files with centroids from the polygons of an input file, see the Vector>Geometry Tools>Polygon Centroids menu. With the new centroid file, distances have been calculated between Goodlife Fitness locations and census tract centroids (shown below in later steps). The new centroid layer has been added to the project and is presented in the screen capture below.

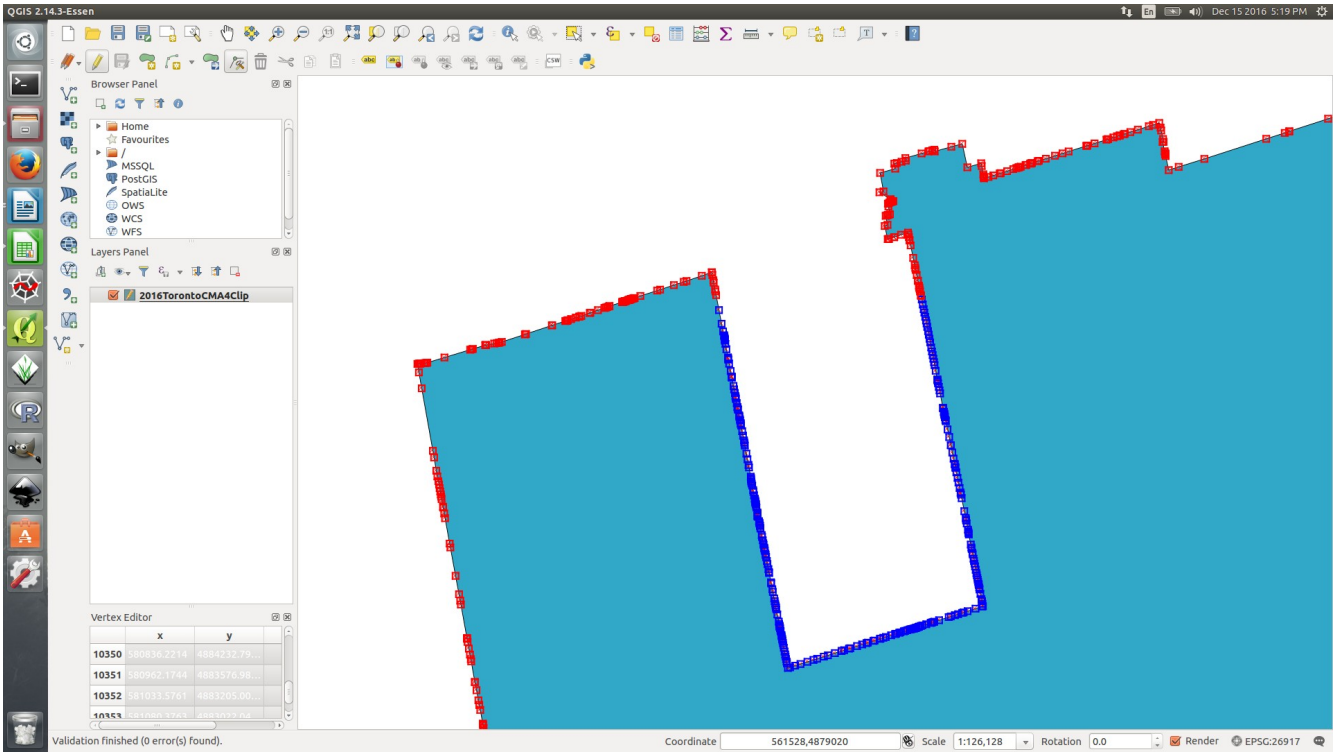


Step 9: Getting Ontario Road Data for Network Distance Calculations

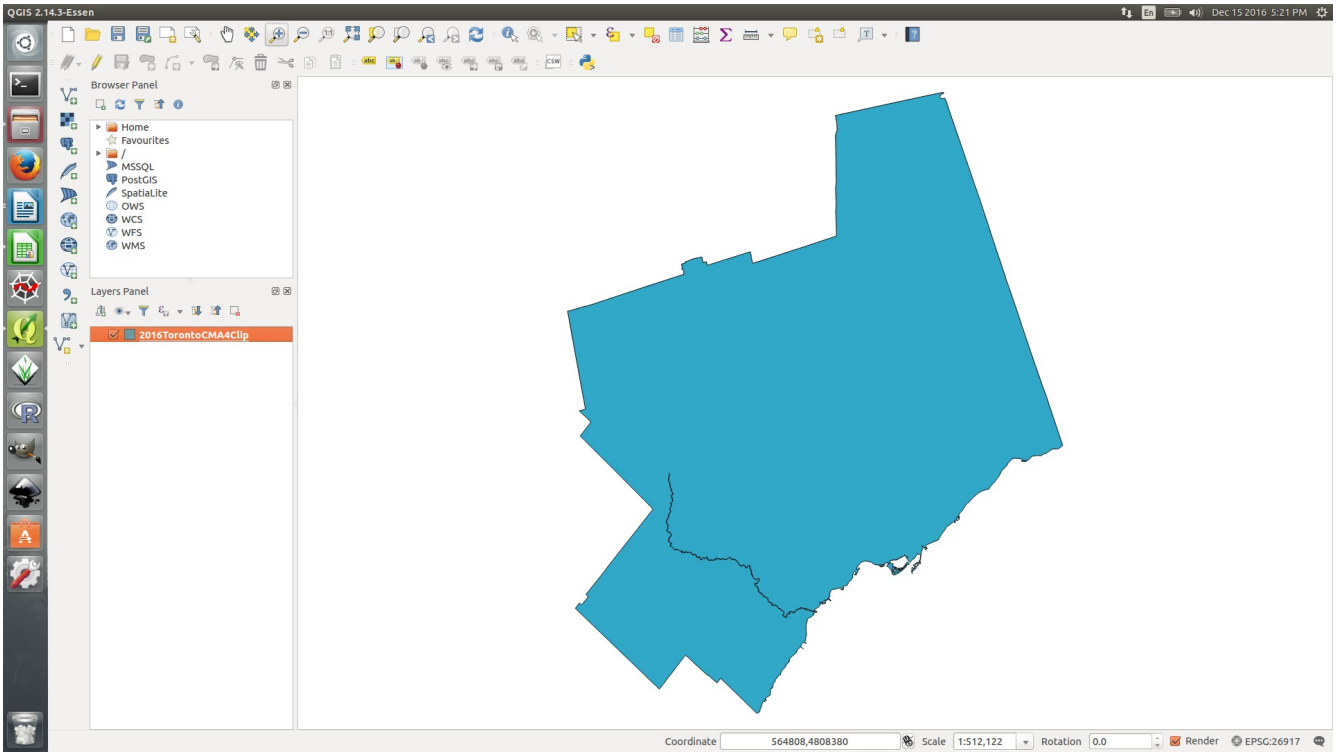
While there are a few sources available to obtain free and open source road network data for the Toronto CMA, Natural Resources Canada has been used here. This data allows for the calculation of distances and also has speed limit information for road segments. Shapfiles are available on GeoGratis for the “GeoBase National Road Network – ON” dataset; this may be found as a .zip file here: http://ftp.geogratis.gc.ca/pub/nrcan_rncan/vector/geobase_nrn_rrn/on/nrn_rrn_on_shp_en.zip . This download provides the road network for all of Ontario. Data has been decompressed and then put into the proper coordinate system (NAD83 / UTM zone 17N [ESPG:26917]) so that it is compatible with the study area and files developed earlier.

Step 10: Preparing a Clipping Template

The data provided by GeoGratis for the Ontario road network needed to be modified to only include those roads within the Toronto CMA. When considering the shape of the Toronto CMA, this didn't make complete sense. The northwestern part of the Toronto CMA has a large rectangular area cut out of it. If network (road) distances are to be calculated between the regions on each side of this void, it doesn't make sense that the drivers have to go around the void to reach their destinations when roads exist to carry them back and forth through the void. This would add a significant distance to their trip and distort the results of the true distance between possible origins and destinations. Instead, a road network has been provided that covers the Toronto CMA and this void. To achieve this goal, a copy of the Toronto CMA boundary file was modified to remove the void. This was done below using the editor in QGIS. The following screen shot presents this mode of QGIS.



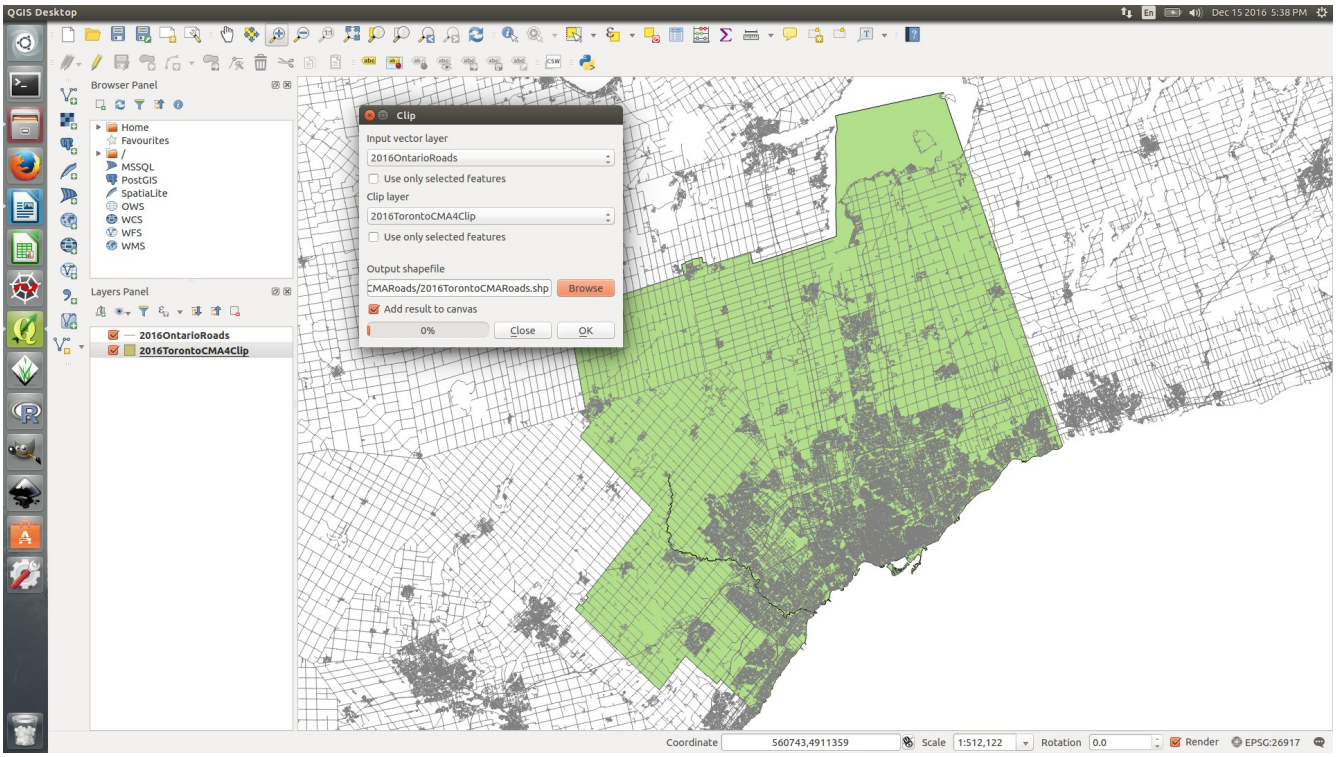
Using the “Node” tool vertices (in blue) have been removed in bulk in a few steps. The result is a new Toronto CMA boundary file without the large void. This is shown in the screen capture below.



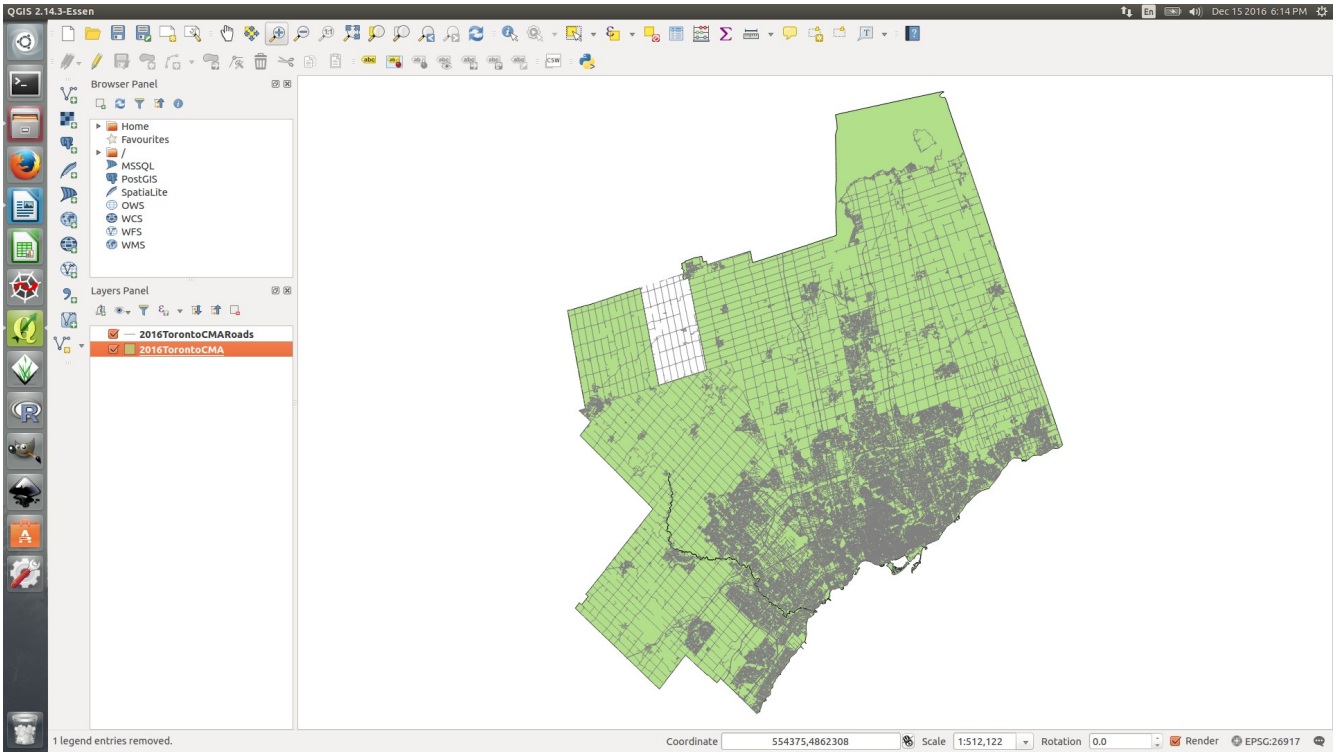
This version of the Toronto CMA has been used to clip the Ontario road network below.

Step 11: Clipping the Ontario Road Data to Match the Toronto CMA

The Ontario road network data has been clipped to match the area covered by the Toronto CMA plus the void in the northwest region. The data obtained from GeoGratis provides the road network for all of Ontario and only roads that service the Toronto CMA are required. The “Clip” tool is available in the Vector>Geoprocessing Tools>Clip menu. The following screen capture presents the settings for the clip. The reprojected file provided by GeoGratis is one of the inputs. The second input, the clip layer, is the boundary file prepared in Step 10 above. The results have been output to a new file. Due to the size of the road network file, this process takes some time to complete – about 35 minutes on a reasonably capable computer.

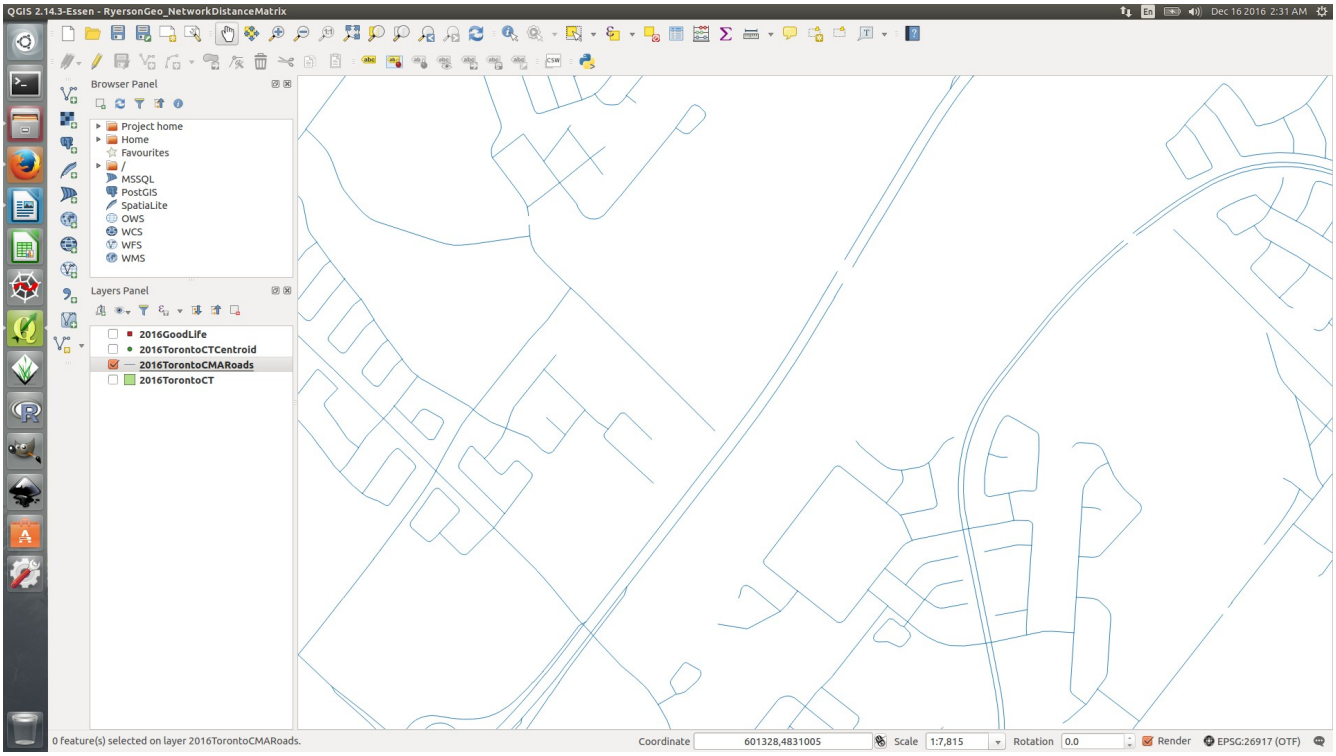


The resulting road network shapefile appears layered on top of the original Toronto CMA shapefile in the screen capture below.

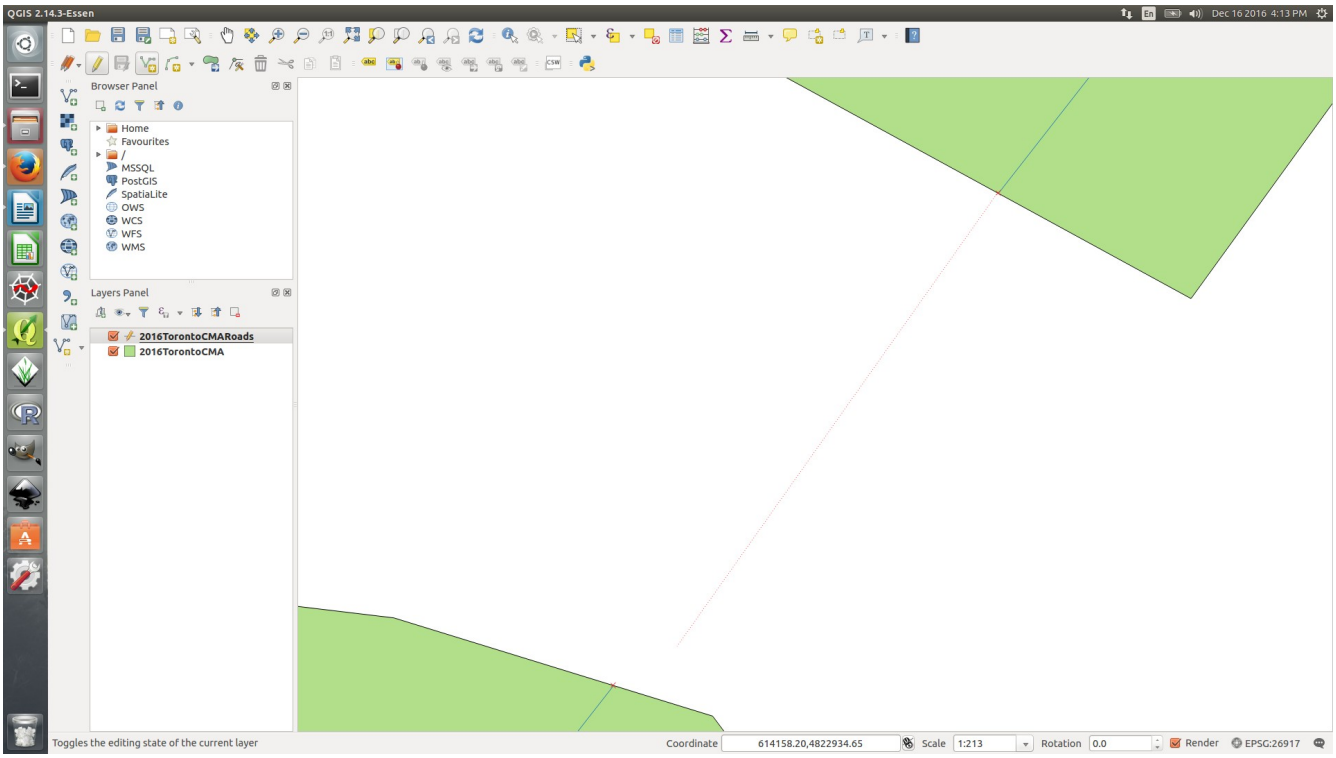


Step 12: Repairing a Road Network Dataset (OPTIONAL)

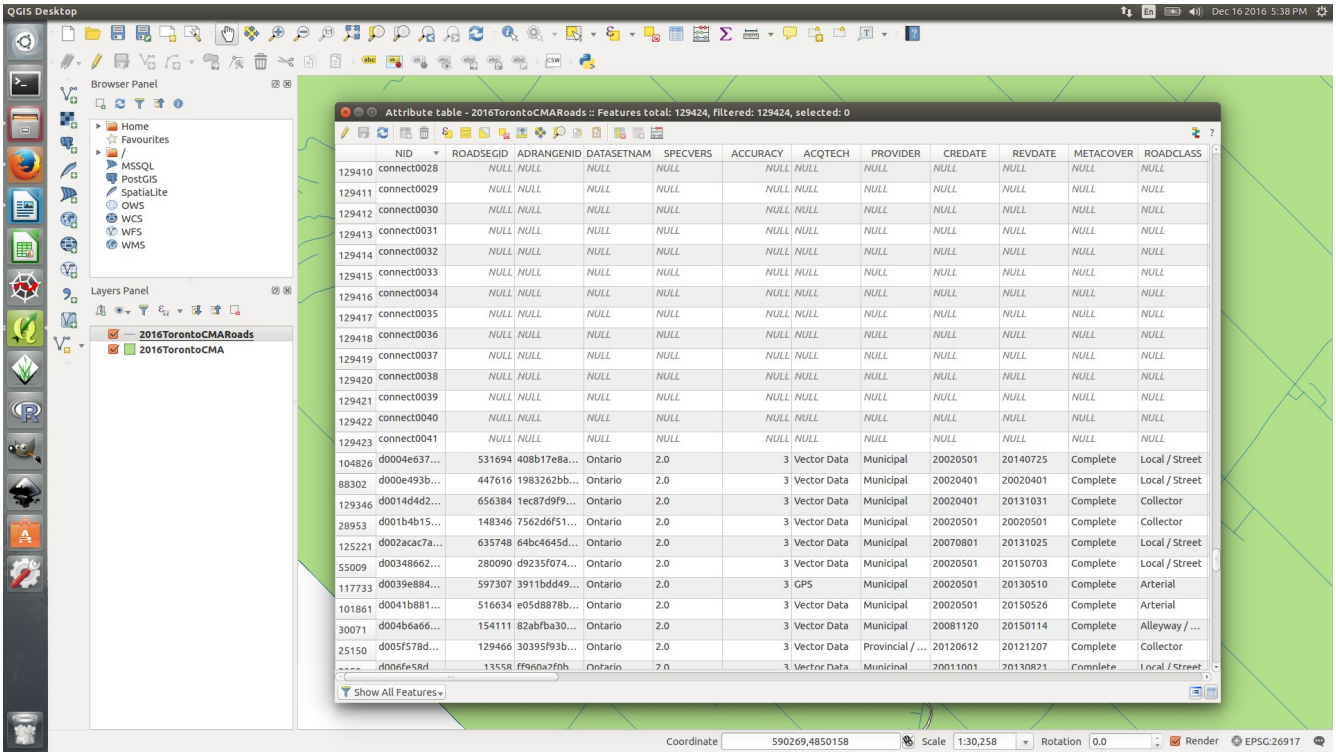
It seems that there are connectivity issues in the road network provided by GeoGratis and they coincide with the Credit River. See the screen capture below.



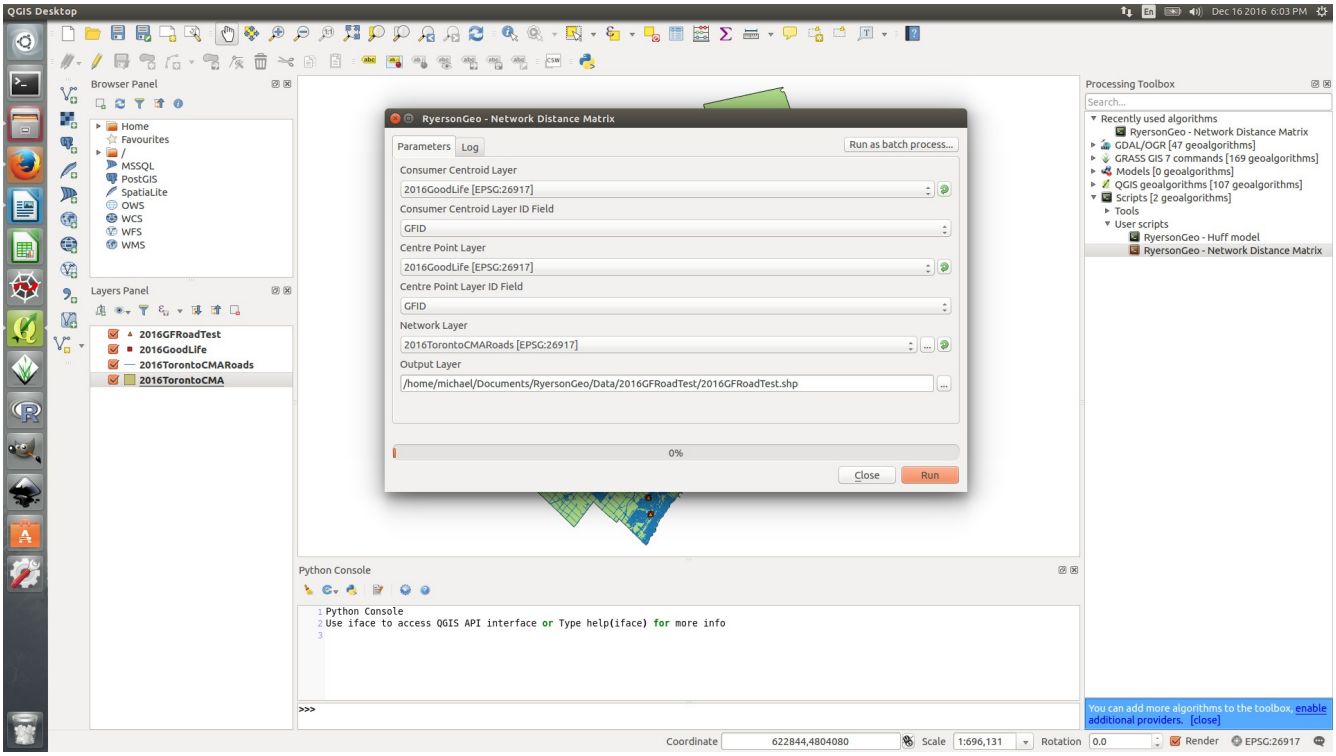
Starting at Lake Ontario, roads are broken and these breaks follow the river north. This results in the Mississauga region of the CMA road network being an island. That is, it is not accessible to the rest of the Toronto CMA. This issue required corrective action before the road network data could be used to calculate network distances between various origins and destinations. To implement the correction, the editor in QGIS has been employed. Using the OpenStreetMap web site as a reference, the “Add Feature” Tool has been applied to create segments to reconnect the roads. See the screen capture below.



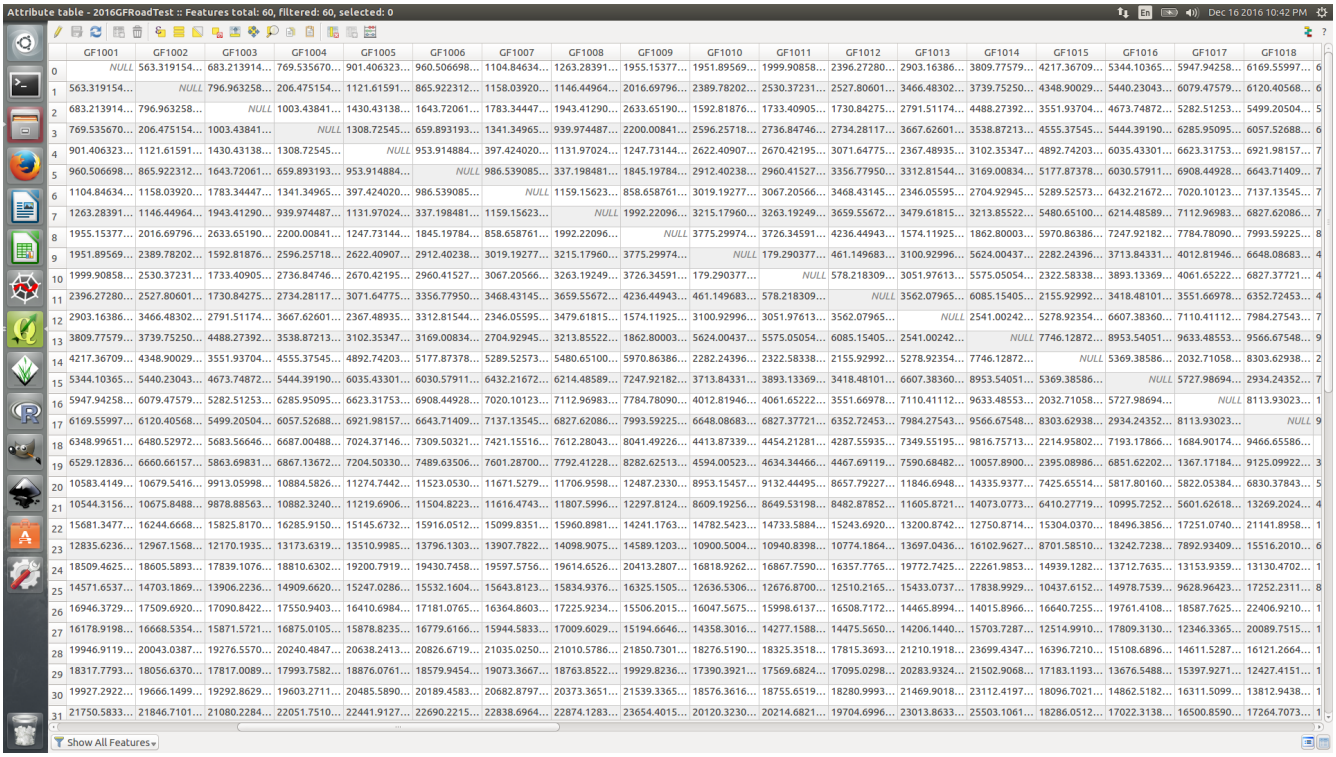
When adding these connecting segments into the road network, the user is prompted to fill in the content for the attribute table. Only a value for the unique identifier (nid) has been provided here; in this case “connect####”. The first segment was assigned “connect0001” and this value was incremented by one for each correction that followed. A total of 41 segments have been added to reconnect the roads. The attribute table for the road network data appears below. With the exception of the unique identifier, all other fields are left as NULL. For this case study, this should not be a problem. For more advanced use of the QGIS network analysis tools, it might be necessary to fill in some of the missing values such as street names, directionality, and speed limits.



With the edits to the road network complete, the data files have also been tested to verify that there are no other issues with connectivity and that the edits have corrected the problems around the Credit River. To achieve this, the network distance matrix python script from Step 14 below has been used.



Setting the script to use Goodlife Fitness locations as both origins and destinations provides a sixty by sixty table of distances. This output is captured in a new shapefile and is reviewed below.



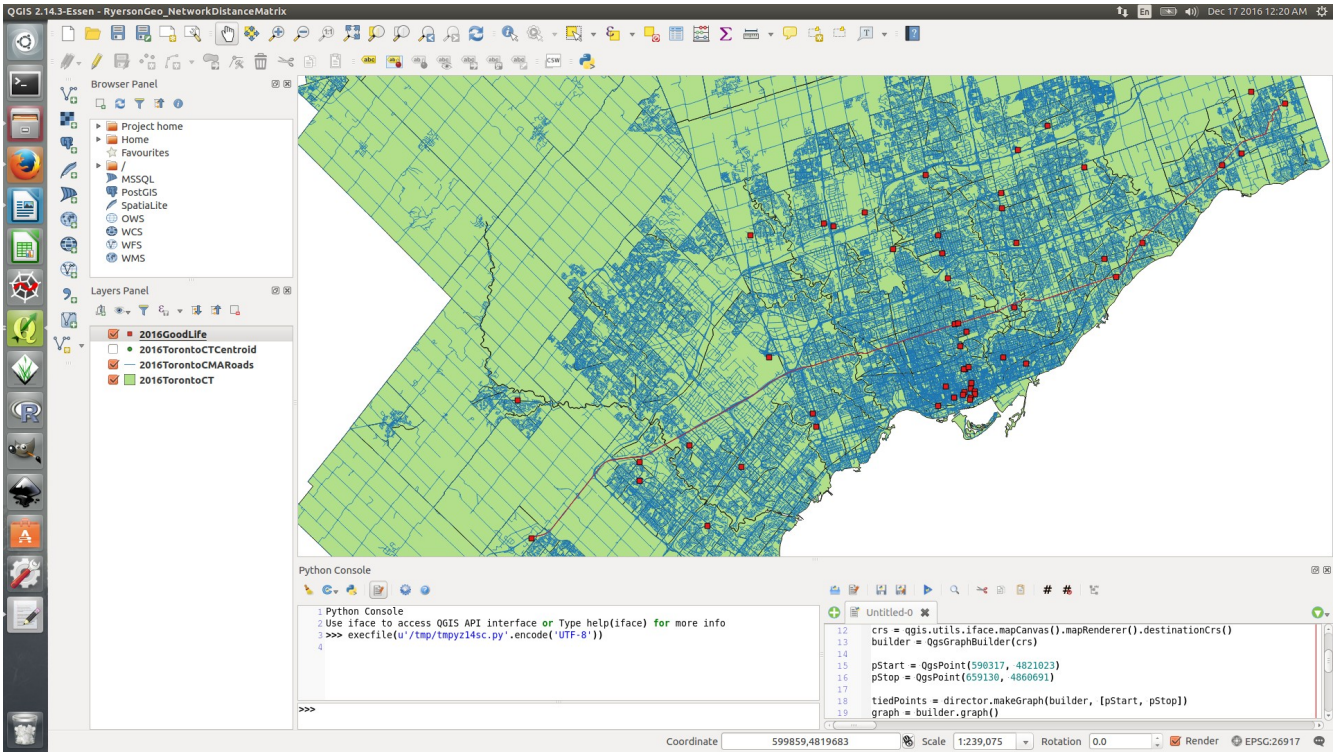
The script has calculated network distances for each pair of Goodlife Fitness locations. There are only two exceptions to this behaviour. If a path cannot be found between the two points, then a NULL will be recorded. The second exception is when the first point is the same as the second point, and this will also result in a NULL. A total of sixty NULL values were recorded in the attribute table presented above and each case was for a location being paired with itself (indicated by the diagonal line of NULLs). With only the sixty expected NULL values appearing in the attribute table, there is confidence that there are no further connectivity issues with the road network sourced from GeoGratis.

There is still a question of whether or not all the connectivity issues have been eliminated so that the best results are being offered. That is, to be sure paths are not being constructed to go around a large inaccessible area, a few spot tests have been completed with a few of the Goodlife Fitness sites located around the Toronto CMA. The sample script provided by QGIS while documenting the Network Analysis tool provides Python code that is easily modified for this purpose. Results are also compared with Google Maps “directions” feature. See the table below for the locations tested and then the results in the screen capture that follows.

Origin ID	Origin Club Name	Destination ID	Destination Club Name	Calculated Network Distance (m)	Google Distance (km)	Good Route?
GF1056	Milton Steeles	GF1018	Toronto Coxwell / Gerrard	53,196	60.2	Yes

GF1056	Milton Steeles	GF1053	Ajax	83,660	87.3	Yes
GF1056	Milton Steeles	GF1060	Alliston	82,127	89.2	Yes
GF1044	Mississauga Heartland	GF1047	Mississauga Millcreek	6,172	7.0	Yes

A screen capture for the point-to-point routing test appears below.

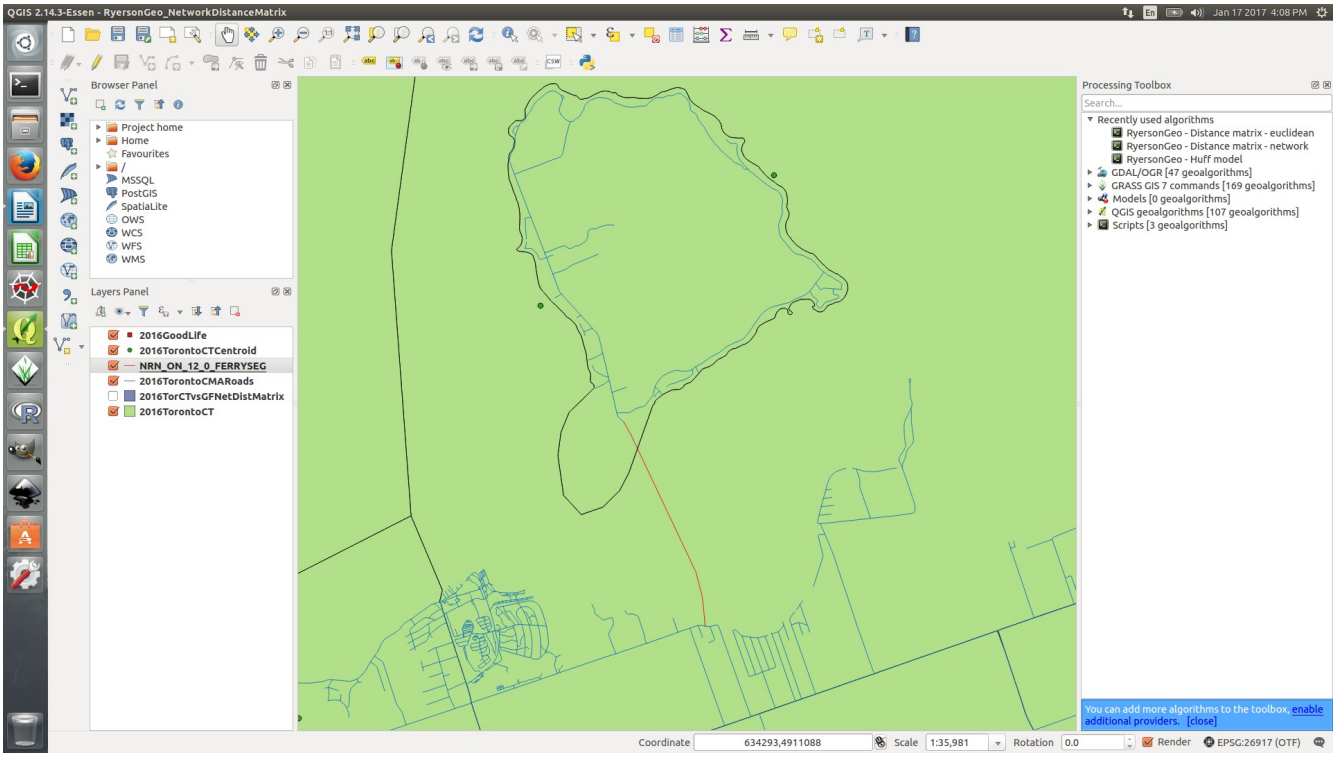


The six sites tested were selected to verify mobility across the Credit River. The red line drawn on the map in the screen capture above indicates the shortest path route produced for the Milton Steeles and Ajax locations. Each test yielded positive results. Based on distance alone, the algorithm used by the QGIS API performs better than Google Maps. It should be noted that using distance alone is not always going to provide the best route. Using highways can result in trips longer on distance and shorter on time. Toll routes may also be used by QGIS where Google Maps may avoid them by default.

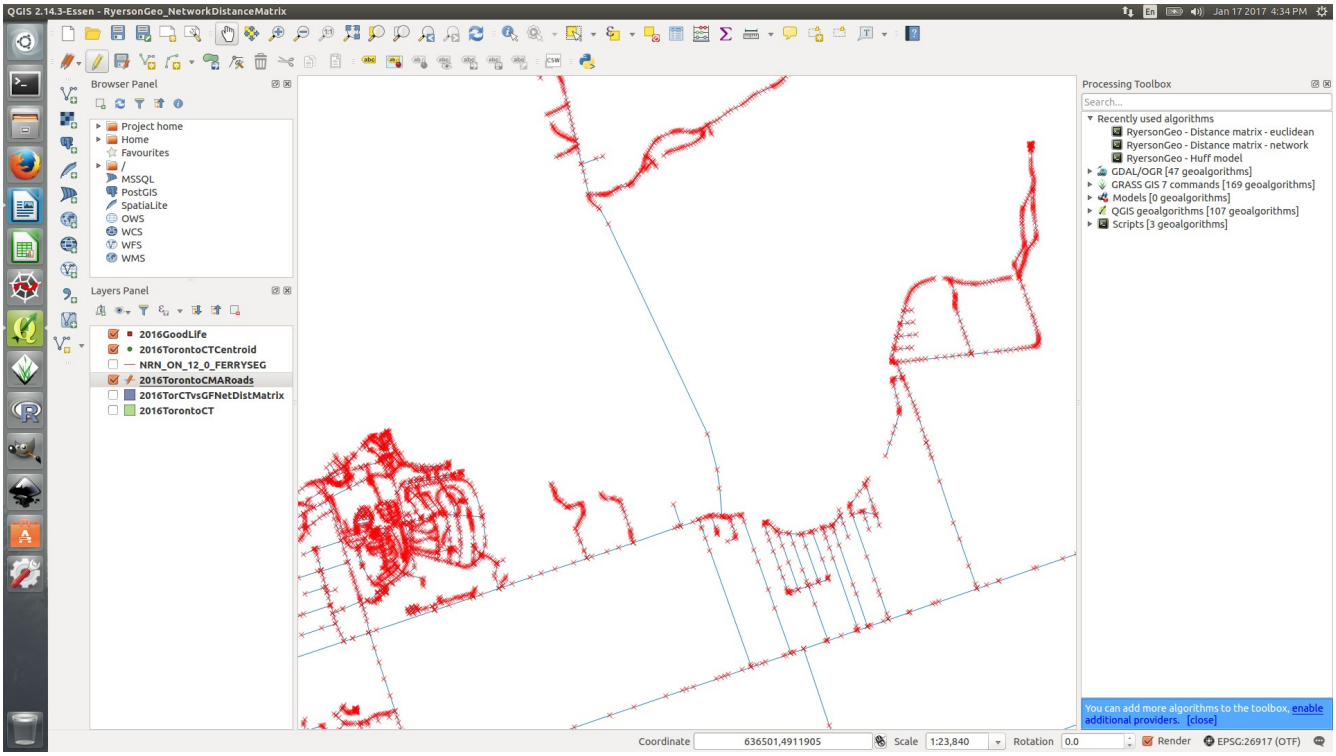
One additional issue needs to be corrected before this data may be used for Huff Model calculations. The Toronto CMA has a few islands and these need to be connected to the road network. Failure to correct for this will result in NULL values in the generated network distance matrix. The network distance matrix requires a value for each consumer-centre pair. If this is not provided for by the matrix, an error will be produced when attempting to calculate Huff Model probabilities. Aside from the NULL value being of a different data type than what the Huff Model script expects, a valid denominator is required for the Huff Model formula. If network issues cannot be resolved then either the euclidean approach should be used or the offending census tracts/regions should be removed from the study.

The three census tracts with island issues are: 5350002.00, 5350476.01, and 5350476.02. These locations relate to Toronto Island and Lake Simcoe's Snake and Georgina Islands. Ferry data is included as a separate shapefile along with the road network data provided by GeoGratis and this is used to make the modifications to connect these three census tracts to the rest of the road network.

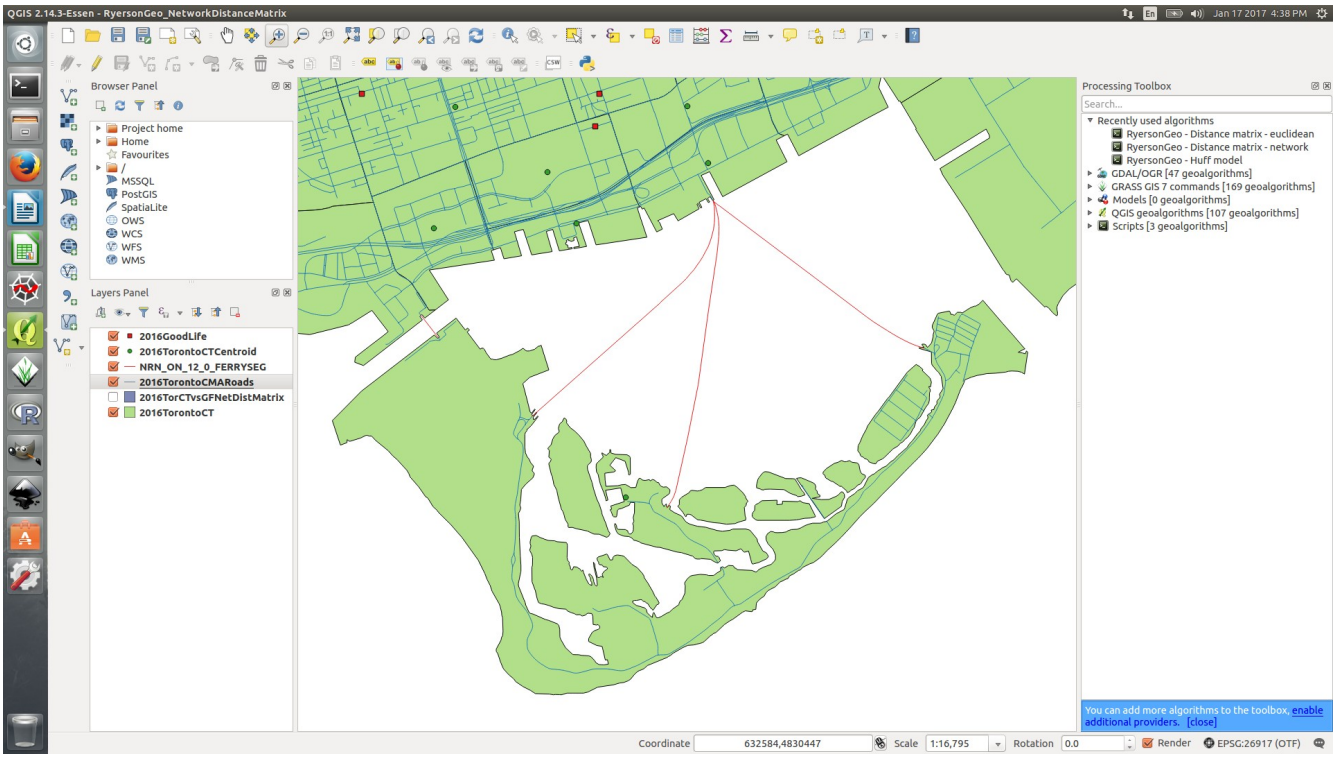
The Georgina Island issue has been resolved first. The screen capture below illustrates the connectivity issue.



Georgina Island, along with Snake Island (not shown), have been defined as a census tract by Statistics Canada. The green centroid point on the west side of Georgina Island is for that census tract. The green centroid point on the east side of Georgina island is for the mainland to the south of Georgina Island and water area. The connectivity issue results as both centroids are placed just off Georgina Island and they “snap” to the road network on the island. This part of the road network does not connect to the roads on the mainland. The available ferry is represented by the red line connecting the mainland to Georgina Island. This line has been duplicated with the QGIS editor to add it to the road network and is shown below.

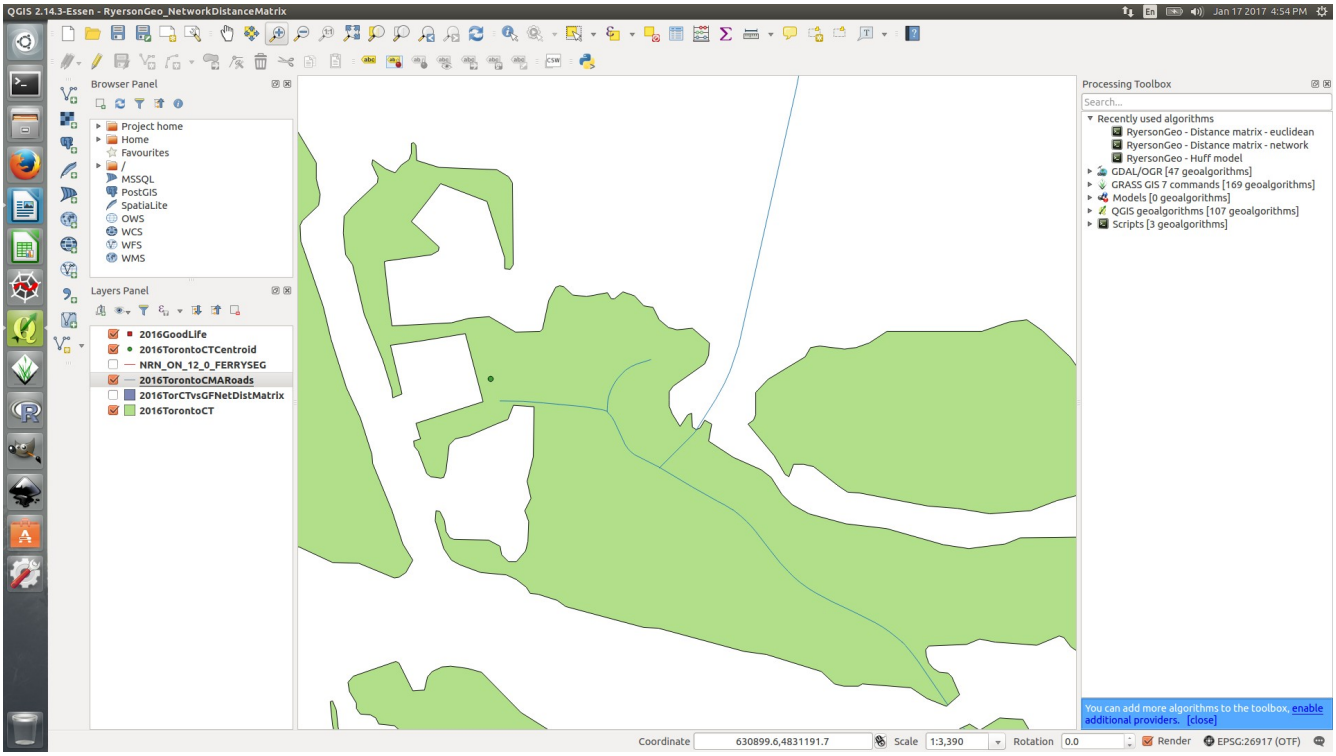


The situation with the Toronto Island issue is more complicated. See the screen capture below.



Three ferries connect mainland Toronto to Toronto Island. Two ferries connect to the larger island on the east and west. The middle ferry connects to an isolated island in the middle of the island cluster. The centroid for this census tract is located on that same middle island. To connect Toronto Island to

the larger road network, the middle ferry path has been duplicated with an additional line segment to connect to the road on that same island. The larger island and other roads do not matter here, only reaching the centroid does. The QGIS editor has been used in a similar manner as before. See the screen capture below.



The two new line features that have been added for Georgina and Toronto Islands have been created as connect0042 and connect0043 in the attribute table. The corrected road network data should now be suitable for the needs of this case study.

Part II: Creating Distance Matrices for Use in Huff Model Calculations

To automate the process of calculating Huff Model probabilities for a large number of consumers and centres, it is useful to calculate the distances between the two entities and store them in a distance matrix. Two matrices have been created below; an euclidean distance matrix and a network distance matrix. Each matrix has consumers in the rows of the matrix while the centres form the columns. The intersection of a consumer and centre holds the distance for that consumer-centre combination. In this case study, consumers are census tracts and centres are Goodlife Fitness locations. Point data are required for the calculation of distance. The Goodlife Fitness data has been prepared as point data in Steps 1-4, and 7. Census tract data has also been prepared and made available as point data (centroids) in Steps 5, 6, and 8. The network distance matrix requires the prepared road network data from Steps 9-12.

Python scripts have been written to use the above data to facilitate the calculation of the distances and store them in an "empty" shapefile for the consumer data. In this case study, the empty file is a census tract boundary file for the Toronto CMA containing only the polygons and CTUID for each census tract. The distances for each Goodlife Fitness location have been appended as columns. These new shapefiles provide for the distance matrix requirements of the Huff Model Python script developed in a later step. It should be noted that shapefiles have size limitations. A shapefile may contain a maximum of 255 fields. Using an "empty" file allows for up to 254 centres before needing to switch to a more advanced storage solution such as a database. An "empty" file has been prepared for use with each distance matrix method by removing all the fields except for the CTUID field (the identifier) and then saving it with a new name. This is specified as the output file for each of the two distance matrix scripts described below.

Step 13: The Euclidean Distance Matrix Python Script

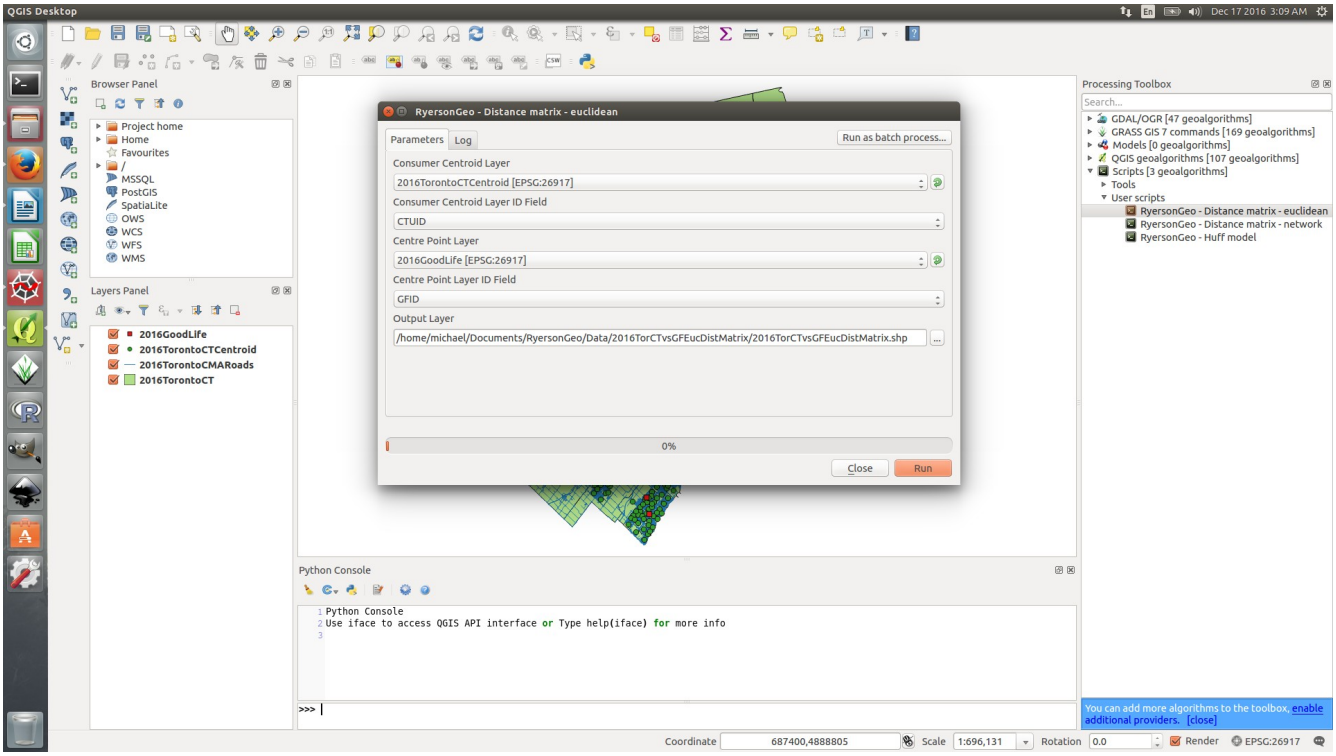
```
##RyersonGeo - Distance matrix - euclidean=name

##Consumer_Centroid_Layer=vector
##Consumer_Centroid_Layer_ID_Field=field Consumer_Centroid_Layer

##Centre_Point_Layer=vector
##Centre_Point_Layer_ID_Field=field Centre_Point_Layer

##Output_Layer=output file

# Script: RyersonGeo_-_Distance_Matrix_-_Euclidean.py
# Author: Michael Morrish
# Date: December 18, 2016
#
# This script takes in two input shapefiles and two field specifications and
# produces a euclidean distance matrix.

# Imports.
from PyQt4.QtCore import *
from PyQt4.QtGui import *

from qgis.core import *
from qgis.gui import *
```

```
# Get the layers.
lyrConsumer = processing.getObject(Consumer_Centroid_Layer)
lyrCentre = processing.getObject(Centre_Point_Layer)

# Get the fields.
fldConsumerID_index = lyrConsumer.fieldNameIndex(Consumer_Centroid_Layer_ID_Field)
fldCentreID_index = lyrCentre.fieldNameIndex(Centre_Point_Layer_ID_Field)

# Need to prepare output layer and add new fields.
# New fields are simply the ID of the Centre.
# Loop through each Centre to construct new field names.
lyrOutput = processing.getObject(Output_Layer)
provider = lyrOutput.dataProvider()

# Optimize feature request for this loop.
request1 =
QgsFeatureRequest().setFlags(QgsFeatureRequest.NoGeometry).setSubsetOfAttributes([Centre_Point_Laye
r_ID_Field], lyrCentre.fields() )

# The loop.
for centreFeature in lyrCentre.getFeatures(request1):

    # Capture value of fldCentreID_index (current ID).
    currentCentreID = centreFeature[fldCentreID_index]

    # Add and name the field. Double type for distances.
    new_field_name = currentCentreID
    provider.addAttributes([QgsField(new_field_name, QVariant.Double)])
    lyrOutput.updateFields()

# Set the output layer for editing.
lyrOutput.startEditing()

# Optimize feature request for outer nested loop.
request2 = QgsFeatureRequest().setSubsetOfAttributes([Consumer_Centroid_Layer_ID_Field],
lyrConsumer.fields() )

# Loop through each Consumer feature.
for consumerFeature in lyrConsumer.getFeatures(request2):

    # Capture value of fldConsumerID_index (current ID).
    currentConsumerID = consumerFeature[fldConsumerID_index]

    # Loop through each Centre feature.
    for centreFeature in lyrCentre.getFeatures():

        # Capture value of fldCentreID_index (current ID).
        currentCentreID = centreFeature[fldCentreID_index]

        # Create a measurement object.
        mObject = QgsDistanceArea()

        # Measure the euclidean distance.
        eDistance = mObject.measureLine(consumerFeature.geometry().asPoint(),
```

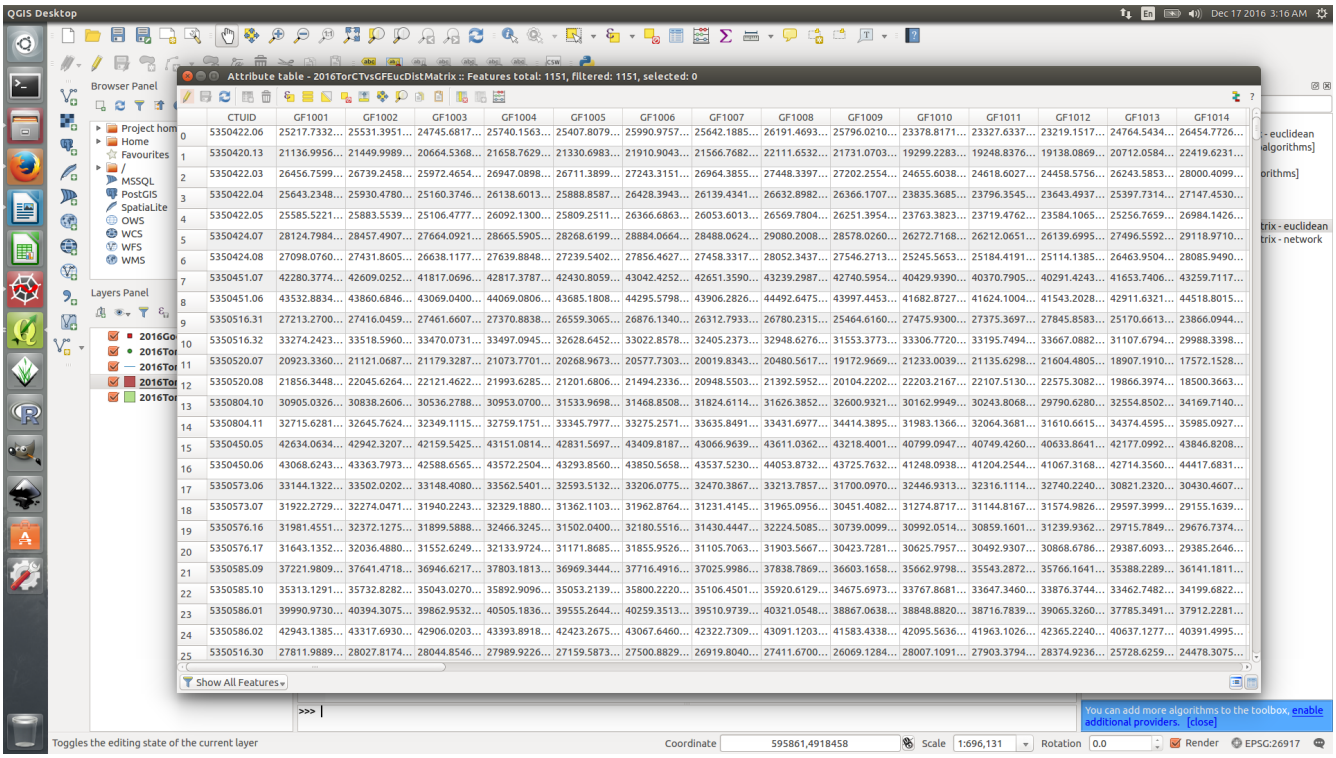
```
centreFeature.geometry().asPoint())

        # Set the euclidean distance value of the new Centre field for
        # the current Consumer and Centre.
        current_distmatrix_field = currentCentreID
        distmatrix_field_index = lyrOutput.fieldNameIndex(current_distmatrix_field)
        lyrOutput.changeAttributeValue(consumerFeature.id(), distmatrix_field_index,eDistance)

# Commit the changes to the layer.
lyrOutput.commitChanges()
```

The script developed to generate the euclidean distance matrix allows the user to specify the required parameters and then generates the distances as appended columns to the specified output file (the "empty" shapefile described above). This improves over previous methods explored and no longer presented here that involved the intermediate steps of creating a comma-separated values (.csv) file, creating a file (.csvt) to define field data types of the comma-separated values file, and having to import and join the data to a shapefile manually in QGIS.

The parameters required of the user for this script include the appropriate consumer data file and the specification of that file's unique identifier. This will be a point file. In this case study, the Toronto CMA centroid file has been specified for this purpose and the CTUID field has been specified as the ID field. The user also needs to specify the file containing the data for the centres. This is also a point-based file and the identifying field for it is also required. This case study uses point data for Goodlife Fitness locations and a GFID field for the identifier. Lastly, the user needs to specify the output file they wish to append the generated distance matrix to. As described above, this file is based on an "empty" census tract boundary shapefile. The screen capture below presents these settings.



The script completes all the calculations and file updates in about one minute on a reasonably capable computer. The attribute table for the output file is presented in the screen capture below. The distances calculated match those generated by the built-in QGIS “distance matrix” algorithm.



This shapefile may now be specified as an input file for the Huff Model script developed in a later section.

Step 14: The Network Distance Matrix Python Script

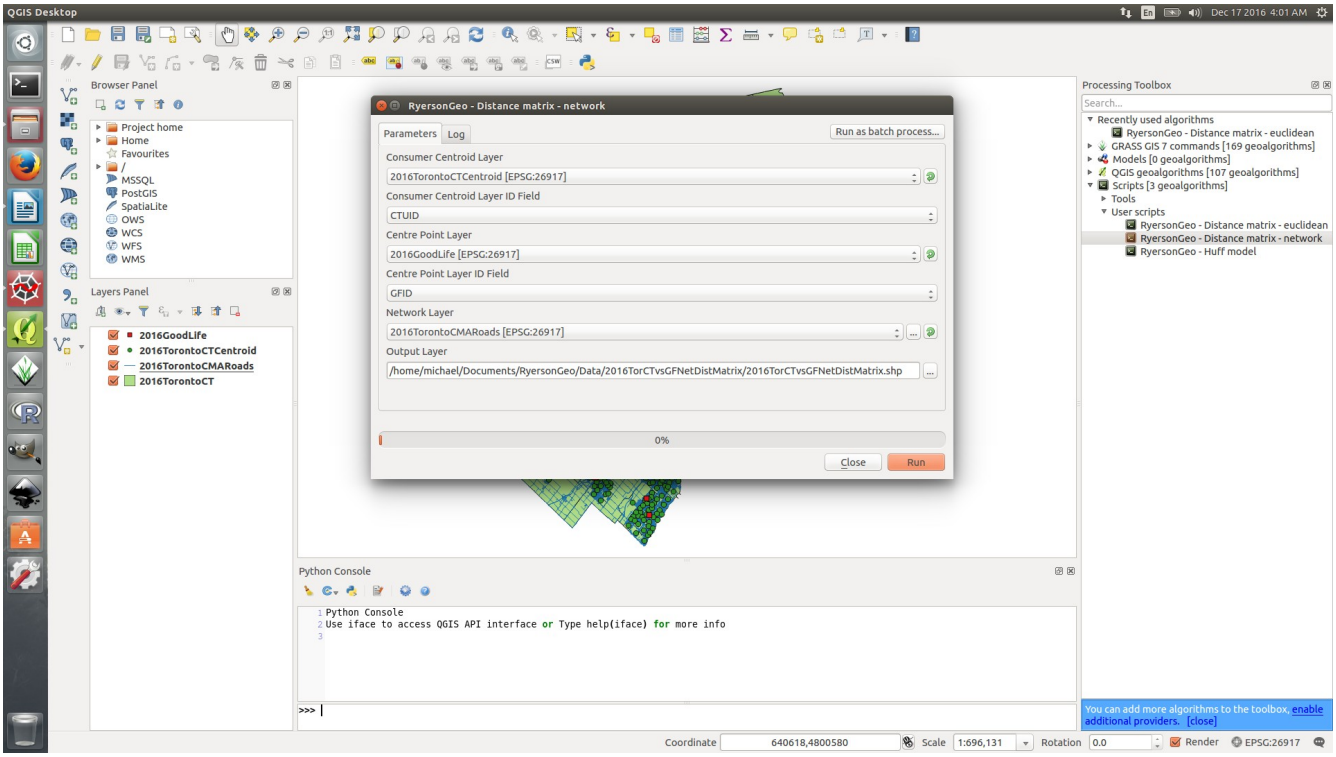
```
##RyersonGeo - Distance matrix - network=name

##Consumer_Centroid_Layer=vector
##Consumer_Centroid_Layer_ID_Field=field Consumer_Centroid_Layer

##Centre_Point_Layer=vector
##Centre_Point_Layer_ID_Field=field Centre_Point_Layer

##Network_Layer=vector

##Output_Layer=output file

# Script: RyersonGeo_-_Distance_Matrix_-_Network.py
# Author: Michael Morrish
# Date: December 18, 2016
#
# This script takes in three input shapefiles and two field specifications and
# produces a network distance matrix.
```

```
# Imports.
from PyQt4.QtCore import *
from PyQt4.QtGui import *

from qgis.core import *
from qgis.gui import *
from qgis.networkanalysis import *

# Get the layers.
lyrConsumer = processing.getObject(Consumer_Centroid_Layer)
lyrCentre = processing.getObject(Centre_Point_Layer)
lyrNetwork = processing.getObject(Network_Layer)

# Get the fields.
fldConsumerID_index = lyrConsumer.fieldNameIndex(Consumer_Centroid_Layer_ID_Field)
fldCentreID_index = lyrCentre.fieldNameIndex(Centre_Point_Layer_ID_Field)

# Setup the graph analysis.
director = QgsLineVectorLayerDirector(lyrNetwork, -1, '', '', '', 3)
properter = QgsDistanceArcProperter()
director.addProperter(properter)
crs = lyrConsumer.crs()
builder = QgsGraphBuilder(crs)

# Need to prepare output layer and add new fields.
# New fields are simply the ID of the Centre.
# Loop through each Centre to construct new field names.
lyrOutput = processing.getObject(Output_Layer)
provider = lyrOutput.dataProvider()

# Optimize feature request for this loop.
request1 =
QgsFeatureRequest().setFlags(QgsFeatureRequest.NoGeometry).setSubsetOfAttributes([Centre_Point_Laye
r_ID_Field], lyrCentre.fields() )

# The loop.
for centreFeature in lyrCentre.getFeatures(request1):

    # Capture value of fldCentreID_index (current ID).
    currentCentreID = centreFeature[fldCentreID_index]

    # Add and name the field. Double type for distances.
    new_field_name = currentCentreID
    provider.addAttributes([QgsField(new_field_name, QVariant.Double)])
    lyrOutput.updateFields()

# Set the output layer for editing.
lyrOutput.startEditing()

# Optimize feature request for outer nested loop.
request2 = QgsFeatureRequest().setSubsetOfAttributes([Consumer_Centroid_Layer_ID_Field],
lyrConsumer.fields() )

# Loop through each Consumer feature.
for consumerFeature in lyrConsumer.getFeatures(request2):
```

```
# Capture value of fldConsumerID_index (current ID).
currentConsumerID = consumerFeature[fldConsumerID_index]

# Loop through each Centre feature.
for centreFeature in lyrCentre.getFeatures():

    # Setup the graph analysis.
    director = QgsLineVectorLayerDirector(lyrNetwork, -1, '', '', '', 3)
    properter = QgsDistanceArcProperter()
    director.addProperter(properter)
    crs = lyrConsumer.crs()
    builder = QgsGraphBuilder(crs)

    # Capture value of fldCentreID_index (current ID).
    currentCentreID = centreFeature[fldCentreID_index]

    # Get the path start point (Consumer point location).
    consumerGeom = consumerFeature.geometry()
    pStart = QgsPoint(consumerGeom.asPoint().x(),consumerGeom.asPoint().y())

    # Get the path end point (Centre point location).
    centreGeom = centreFeature.geometry()
    pStop = QgsPoint(centreGeom.asPoint().x(),centreGeom.asPoint().y())

    # Build the graph.
    tiedPoints = director.makeGraph(builder, [pStart, pStop])
    graph = builder.graph()

    # Attach the start and end points to the graph.
    tStart = tiedPoints[0]
    tStop = tiedPoints[1]

    idStart = graph.findVertex(tStart)
    idStop = graph.findVertex(tStop)

    # Solve the shortest path problem with dijkstra.
    (tree, cost) = QgsGraphAnalyzer.dijkstra(graph, idStart, 0)

    if tree[idStop] == -1:
        # If a path cannot be found, print error msg.
        print "Path not found"
    else:
        # If a path can be found, build array of points in path.
        p = []
        curPos = idStop
        while curPos != idStart:
            p.append(graph.vertex(graph.arc(tree[curPos]).inVertex()).point())
            curPos = graph.arc(tree[curPos]).outVertex();

        p.append(tStart)

        # Create a new feature and put the shortest path polyline in it.
        # Then get the length of that line.
```

```
            ftShortestPath = QgsFeature()
            ftShortestPath.setGeometry(QgsGeometry.fromPolyline(p))
            shortestPathNetworkDistance = ftShortestPath.geometry().length()

            # Set the network distance value of the new Centre field for
            # the current Consumer and Centre.
            lyrOutput.startEditing()
            current_distmatrix_field = currentCentreID
            distmatrix_field_index = lyrOutput.fieldNameIndex(current_distmatrix_field)
            lyrOutput.changeAttributeValue(consumerFeature.id(),
distmatrix_field_index,shortestPathNetworkDistance)
            lyrOutput.commitChanges()

# Commit the changes to the layer.
lyrOutput.commitChanges()
```

The network distance matrix Python script was briefly seen in action in Step 12 and used to help verify the completeness of a road network data set. This script works in a similar fashion to the script developed to calculate euclidean distances. The user is required to provide the information for both the consumer and centre input files, and also the respective ID fields. In exactly the same way as the earlier script, the user also prepares and specifies an output file to hold the generated distance matrix. The difference in the required parameters for this script is the addition of a “Network Layer” parameter. This layer contains the line-based data used to calculate the network-based distances. This is usually a road network file, but it could also be something else, such as; a telephone pole/wire network, river system, or sewer network. The screen capture below presents the parameters specified for the Goodlife Fitness case study.

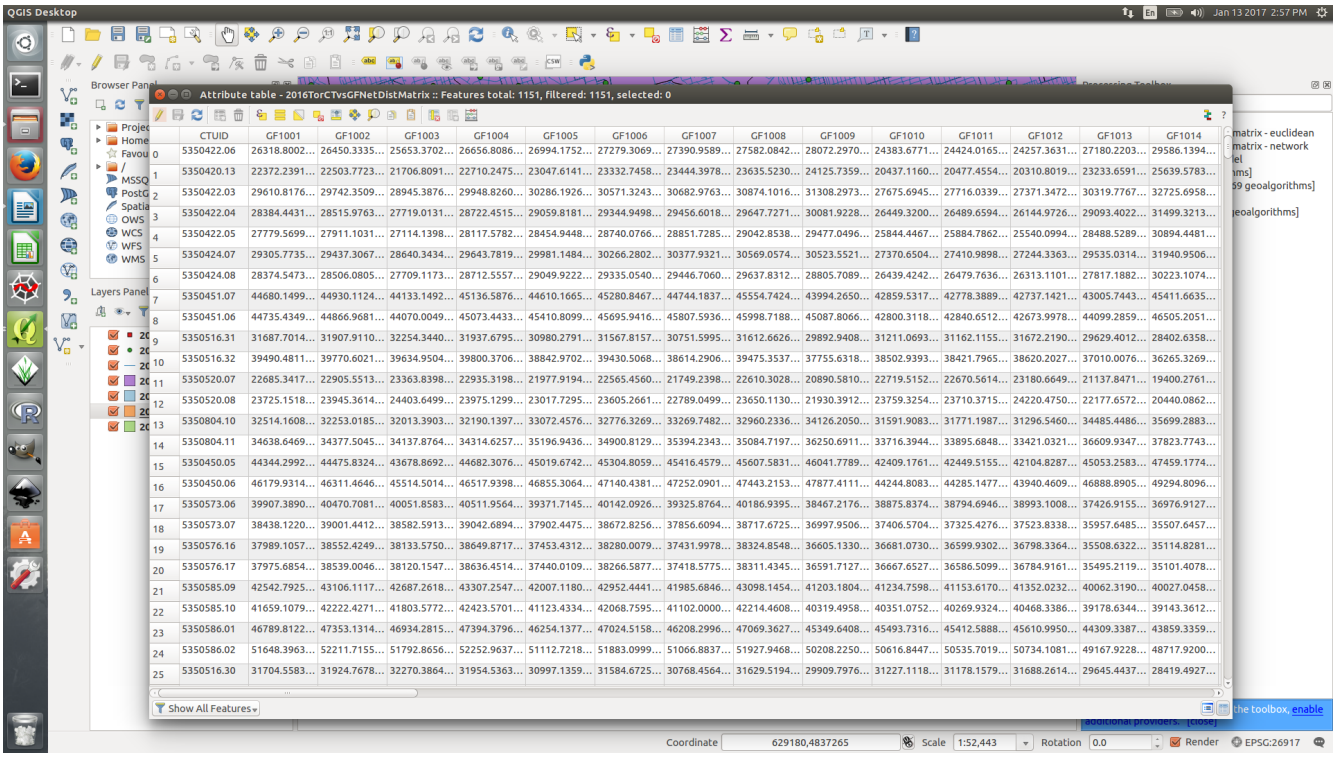


The script itself produces great results. The QGIS network analysis tools make use of the Dijkstra

method for finding the shortest distance between two points. This was observed in action in Step 12. The shortest path was provided in each of the test cases. The downside to the goal of minimizing distance is that it may not provide the most efficient or desirable travel route. For example, toll routes may be included in the resulting shortest path. For the purposes of this case study the results are acceptable.

The real issue with the algorithm provided by the QGIS networking analysis tools is the time it takes to complete the Dijkstra search for the shortest path. When considering what it actually does it is quite impressive, but for a large volume of data, results cannot be delivered in real-time. The Goodlife case study can be used to demonstrate this. It takes about four seconds to calculate a shortest path with this algorithm. This sound great, but if there are 1,150 consumers (2016 census tracts) and 60 centres (Goodlife locations), then a distance matrix of 69,000 cells is required. Multiplying the 69,000 shortest path distance calculations by four seconds each yields 276,000 seconds. That translates to 4,600 minutes, or 76 hours and 40 minutes, or about 3.25 days. This estimate also matches closely with the earlier experience (in Step 12) of using the script. The sixty by sixty network distance matrix that was produced to test the road network took four hours and twenty minutes to complete. That works out to 260 minutes to complete 3,600 distance calculations or 4.33 seconds per distance calculation.

The script was run to produce a network distance matrix for the Goodlife Fitness locations in the Toronto CMA. The actual time to complete the process was closer to 3.5 days. A screen capture of the results in the attribute table is shown below.



The values in this screen capture may be compared to those that appear in the earlier one for the euclidean distance method; the same census tracts and Goodlife Fitness clubs have been presented here.

Part III: The Huff Model

The Huff Model allows questions to be asked about market areas. The formula considers a consumer at a location “i” and a centre (often a shopping centre) at location “j”. The consumer is often represented as the population of a region such as a census tract. The centre is usually thought of as a shopping centre but could also be a city, attraction, or other type of destination. The formula is concerned with identifying the probability that consumer i will shop or travel to centre j. Influencing the formula are the distance “d” between the consumer and the centre and some measure of utility or attractiveness associated with the centre. Utility usually takes the form of size “s” and may measure square feet, number of employees, number of stores, or number of courses. Temperature could also be used – warmer weather attracts tourists. As with other gravity models, a greater size will draw greater attention from consumer i.

The Huff Model from the Consumer's Perspective

$$P_{ij} = \frac{\frac{S_j}{d_{ij}^b}}{\sum_j \left(\frac{S_j}{d_{ij}^b} \right)}$$

The Huff Model is typically viewed from the perspective of the consumer. The question may be asked: “What is the probability that consumer i will shop at centre j?”. To answer this question, a study area is considered and a probability that consumer i will shop at any centre j in the study area is calculated. If there are five centres, then five probabilities will be calculated for consumer i and the sum of the five probabilities will equal 1.

When considering a map that would present the study area of the consumers (census tracts) and centres (shopping centres), only one centre's set of probabilities would be displayed. If there are five centres, the map would show each census tract's probability for just the one shopping centre that is being considered. If a map of probabilities is produced for each of the five centres, then the five values for a census tract could be read across the five maps and they will sum to 1. In contrast, the sum of the values for all the consumers (census tracts) on the individual map will not equal 1.

The consumer perspective of the Huff Model is defined by the j under the Sigma in the formula above. If a grid is established to calculate the probabilities, consumers (census tracts) can be put into rows and centres (shopping centres) can be put into columns. The j under the Sigma indicates that the denominator of the Huff Model formula will only include the sum of the values for one row's (consumer) data. If the denominator represents all of one consumer's spending or attention, then the numerator represents that portion assigned to a given centre j.

The consumer perspective is important when considering the possible applications. It is often desirable to understand how consumers allocate their resources to the market. This perspective of the Huff Model allows for the likely distribution of consumer resources (disposable income, time, etc.) to be predicted. This estimate may be compared with reality and performance by a given centre may be questioned. If the centre is attracting less than their expected share of consumer resources, then action may be

required. If the centre is attracting more than their share of resources, then they are likely doing well.

An exponent “b” appears in the formula. This is used to adjust for the friction of distance associated with the goods or services made available at the centre. A number between or equal to the values of 1 and 3 may be used for this exponent.

“What if” scenarios may also be completed. For example, what would be the impact of adding or removing or changing a centre in the study area? These types of experiments can be completed with relative ease.

A Python Script for the Huff Model (RyersonGeo_-_Huff_model.py)

A Python script has been written for the Huff Model as presented above. The code for the script is presented below.

```
##RyersonGeo - Huff model=name

##Consumer_Layer_with_Distance_Matrix=vector
##Consumer_Layer_ID_Field=field Consumer_Layer_with_Distance_Matrix
##Centre_Layer=vector
##Centre_Layer_ID_Field=field Centre_Layer
##Centre_Layer_Attractiveness_Field=field Centre_Layer
##Huff_Exponent_Value=selection 1; 2; 3
##Output_Layer=output file

# Script: RyersonGEo - Huff Model
# Author: Michael Morrish
# Date: January 13, 2017
#
# This script takes in two input shapefiles, three field specifications, and
# one numeric value to produce Huff model probabilities.



# Imports.
from qgis.core import *
from PyQt4.QtCore import *

# Get the layers.
lyrConsumer = processing.getObject(Consumer_Layer_with_Distance_Matrix)
lyrCentre = processing.getObject(Centre_Layer)

# Get the fields.
fldConsumerID_index = lyrConsumer.fieldNameIndex(Consumer_Layer_ID_Field)
fldCentreID_index = lyrCentre.fieldNameIndex(Centre_Layer_ID_Field)
fldCentreAttract_index = lyrCentre.fieldNameIndex(Centre_Layer_Attractiveness_Field)

# Use dropdown list index to specify the Huff model exponent.
if Huff_Exponent_Value == 0:
    expHuff = 1
elif Huff_Exponent_Value == 1:
    expHuff = 2
elif Huff_Exponent_Value == 2:
    expHuff = 3
```

```
# Need to prepare output layer and add new field.
# New field is "Hi" plus the ID of the Centre.
# Loop through each Centre to construct new field names.
lyrOutput = processing.getObject(Output_Layer)
provider = lyrOutput.dataProvider()

for centreFeature in lyrCentre.getFeatures():

    # Capture value of fldCentreID_index (current ID).
    currentCentreID = centreFeature[fldCentreID_index]

    # Add and name the field.
    new_field_name = 'Hi' + currentCentreID
    provider.addAttributes([QgsField(new_field_name, QVariant.Double)])
    lyrOutput.updateFields()

# Set the output layer for editing.
lyrOutput.startEditing()

# Loop through each Consumer feature.
for consumerFeature in lyrConsumer.getFeatures():

    # Capture value of fldConsumerID_index (current ID).
    currentConsumerID = consumerFeature[fldConsumerID_index]

    # Create a total variable for the sumJ of Sj/dij values for use in the nested loop.
    sumJ_sjdivdij = 0.0

    # Huff Formula: [(sj/(dij)^b)/(SUMj(sj/(dij)^b))] for a given consumer i and centre j.
    # sumJ_sjdivdij is the denominator of this formula and is first loop below.
    # Exponent b is for friction of distance of the product or service.
    # Second loop below calculates numerator and completes Huff calc for a given ij.

    # Loop through each Centre feature to calculate a consumer's sumJ_sjdivdij.
    # This is the Huff formula denominator.
    for centreFeature in lyrCentre.getFeatures():

        # Capture value of fldCentreID_index (current ID).
        currentCentreID = centreFeature[fldCentreID_index]

        # Capture value of fldCentreAttract_index (current Centre Attractiveness).
        currentCentreAttract = centreFeature[fldCentreAttract_index]

        # Capture distance value for this Centre and this Consumer.
        # currentCentreID should match to field name in attrib table.
        currentDistance = consumerFeature[currentCentreID]

        # Calculate Centre Attractiveness / Distance^b >> (Sj/dij**b)

        # If statement to manage computing cost of exponent calculation.
        if expHuff == 1:
            sjdivdij = currentCentreAttract / currentDistance
        else:
            sjdivdij = currentCentreAttract / (currentDistance**expHuff)
```

```
            # Add new Sj/dij^b to sumJ_sjdivdij.
            sumJ_sjdivdij = sumJ_sjdivdij + sjdivdij

        # Loop through each Centre a second time to calculate Huff proportion.
        for centreFeature in lyrCentre.getFeatures():

            # Capture value of fldCentreID_index (current ID).
            currentCentreID = centreFeature[fldCentreID_index]

            # Capture value of fldCentreAttract_index (current Centre Attractiveness).
            currentCentreAttract = centreFeature[fldCentreAttract_index]

            # Capture distance value for this Centre and this Consumer.
            # currentCentreID should match to field name in attrib table.
            currentDistance = consumerFeature[currentCentreID]

            # Calculate Centre Attractiveness / Distance^b >> (Sj/dij**b)

            # If statement to manage computing cost of exponent calculation.
            if expHuff == 1:
                sjdivdij = currentCentreAttract / currentDistance
            else:
                sjdivdij = currentCentreAttract / (currentDistance**expHuff)

            # Complete the Huff formula calculation.
            calcHuffI = sjdivdij / sumJ_sjdivdij

            # Set the value of the new Hi field for the current Consumer and Centre.
            current_Huff_field = 'Hi' + currentCentreID
            Huff_field_index = lyrOutput.fieldNameIndex(current_Huff_field)
            lyrOutput.changeAttributeValue(consumerFeature.id(), Huff_field_index,calcHuffI)

# Commit the changes to the layer.
lyrOutput.commitChanges()
```

Step 15: Using the Huff Model Python Script in QGIS

Using the Huff Model Python script in QGIS is straightforward. The output of the first fourteen steps above provides for most of the requirements to make use of the this script. A few additional steps are also required and these are discussed below.

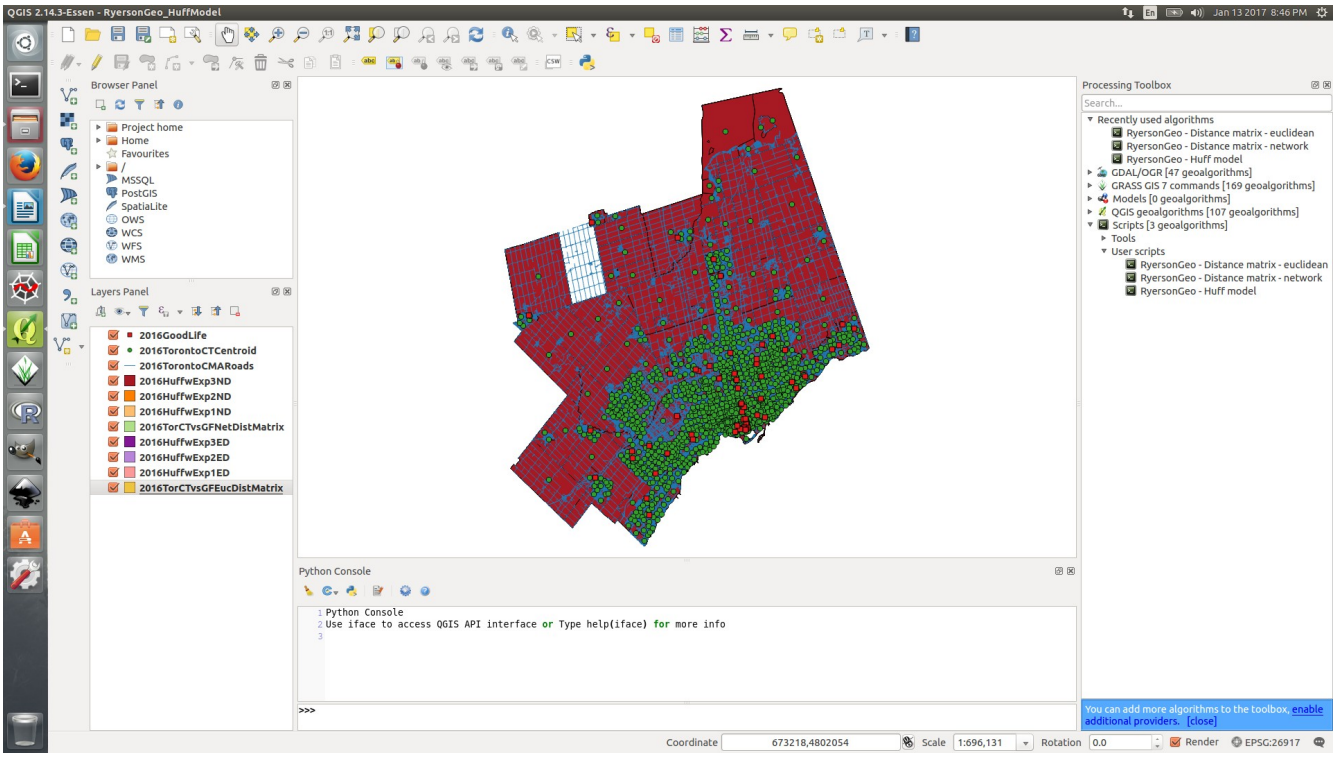
An output file is needed for the script. The output file is actually an existing file that is modified by the script – fields are added to it along with new data. It is the user's choice as to what they provide for this file. As presented in the discussions for the distance matrix scripts, an “empty” shapefile may be used for this purpose. Depending on the number of columns remaining unused (out of a total of 255) in the shapefile generated by the distance matrix scripts, that file could also be used as both an input and an output file. New columns will be appended after the distance matrix columns. This method would use the output shapefile produced in either Step 13 or 14. If an existing distance matrix file is used, a backup/copy of those files should be completed first

To make use of the Huff Model Python script, that script also needs to be added to the appropriate

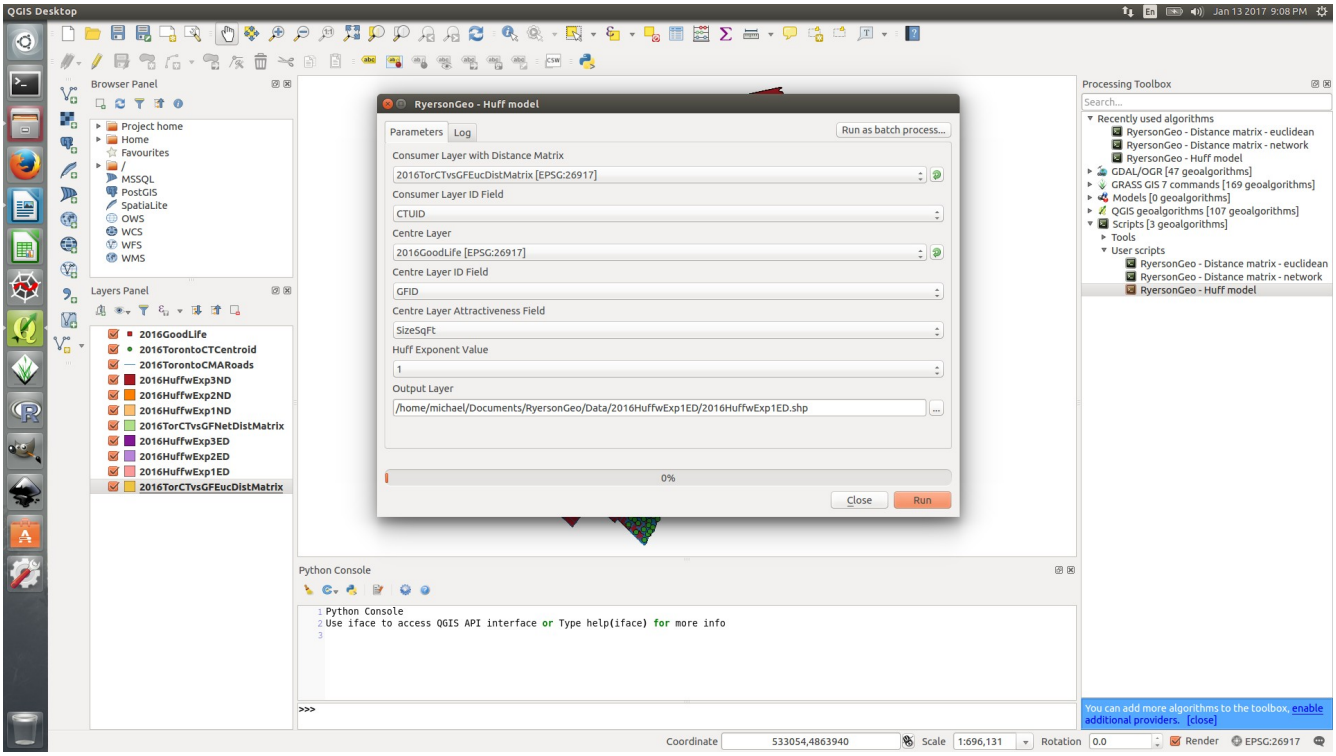
folder on the user's computer so that it is recognized by QGIS. Once there, it can be added to a toolbox (and may be recognized automatically) in the Processing Toolbox panel of QGIS. To execute the script, right-click on it and click “execute”.

The screen capture below illustrates the starting point that has been prepared for using the Huff Model script for this case study. In the Layers Panel on the left, eleven layers are present. Six of these are layers that have been prepared for Huff Model output and are “empty” files. The user may choose to run the Huff Model script with either a euclidean or network distance matrix. They may also select an exponent value of 1, 2, or 3 for use in the calculation of the Huff probabilities. These possibilities result in the need for these six output files. Layers with “ED” in the name are for output based on a euclidean distance while names with “ND” are for the network distance matrix input. The “Exp#” part of the layer name documents the exponent value that was used to generate the Huff Model results stored in the file. Different fields in the attribute table may be used as the variable for attractiveness or size and, if multiple experiments are completed, this could also be tracked in the layer name. A naming convention is important for managing the result files. Each user can use their own system, but they should have one to ensure they know what has been used to generate each set of results.

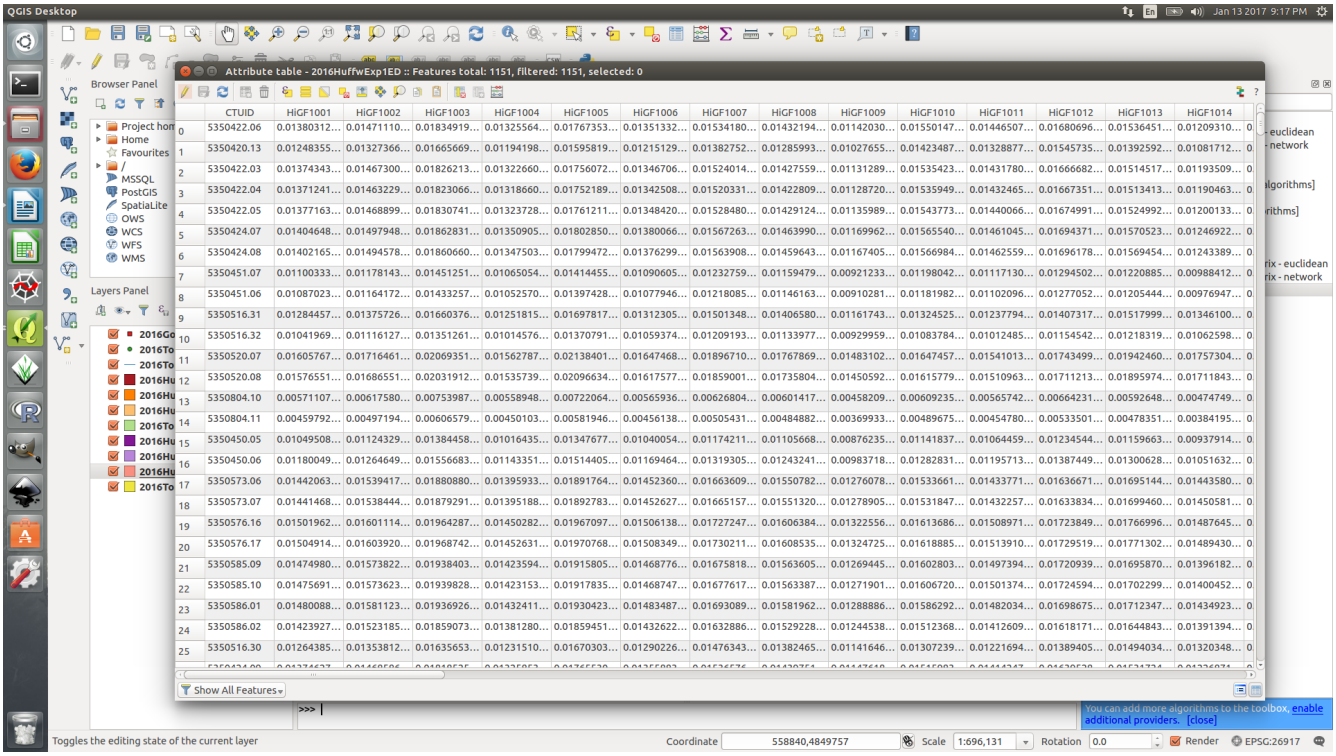
An output file should only be used with the Huff Model script once. The script generates new field names based on the unique identifiers of the data being used. If the script is run with the same output file twice, there is likely to be an error as the generated names have already been used. Due to this limitation, a new output file should be used for new experiments.



Once executed, the script starts off by requesting several parameters to be specified by the user. See the next screen capture below.

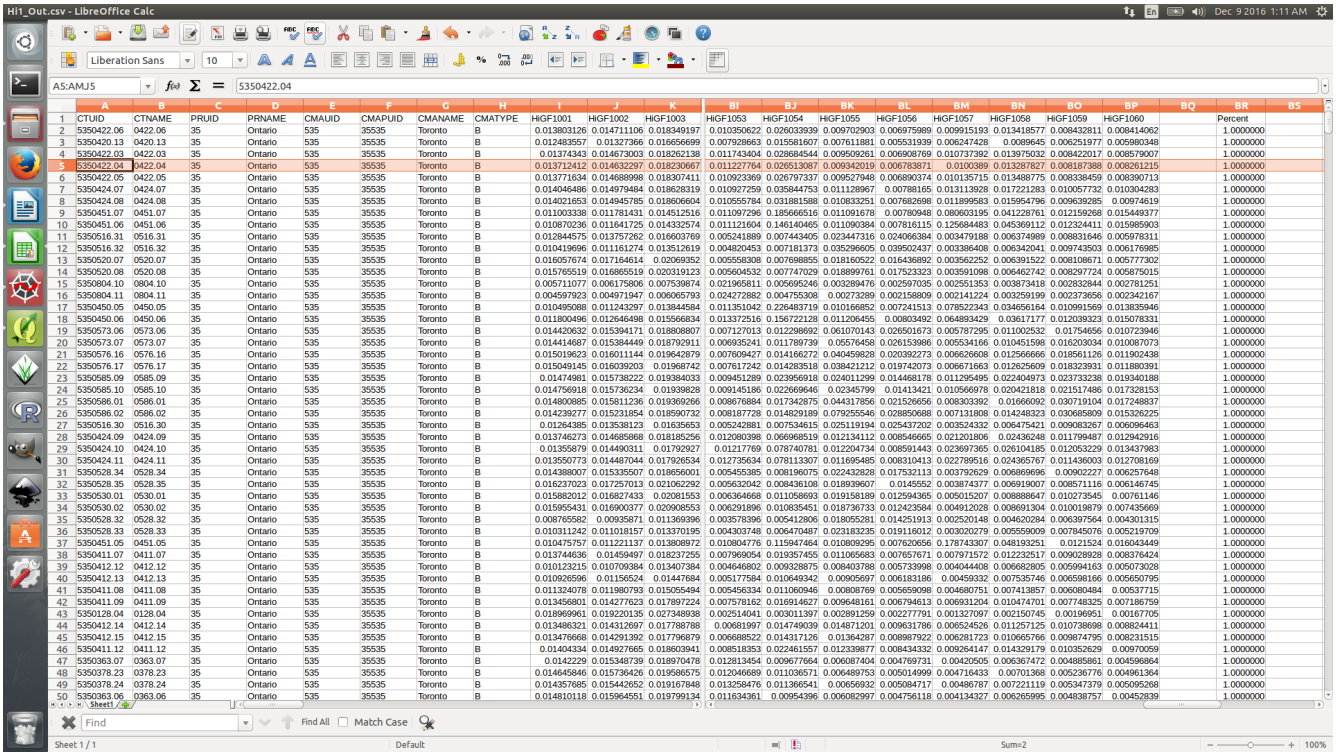


The first request is for a shapefile that contains the consumer information along with the distances between each consumer and each centre. The shapefile created in either Step 13 (euclidean distance matrix) or Step 14 (network distance matrix) provides for this need. Second, the ID field for the consumer layer needs to be specified. In this case, it is “CTUID”. The next request is for the point layer for the centres. A shapefile with the Goodlife Fitness locations along with the size parameter is specified. This file was created in Steps 1-4 and 7. Two parameters are specified next to identify the ID and Size fields for the specified centre layer. It should be noted that if more than one size variable is available, the desired one could be selected here and the script could be run multiple times to explore different measures of utility individually. The user needs to specify an exponent value for use in the Huff Model formula. A value of 1 is the default and the user may also select a value of 2 or 3 from the dropdown list. Lastly, the script needs to know where to put the results. An “empty” file prepared as indicated above has been specified. Different output files can be generated for different cases by changing the parameters and this has been done here. Running the script for each of the six configurations for this case study takes less than a minute each; this creates sixty new fields and writes sixty-nine-thousand pieces of data to each output file. The attributes table for the first results file is presented in the screen capture below. This output is for the euclidean distance matrix with the exponent set to a value of 1.



Each of the sixty new fields have been added to the output file. The prefix “Hi” is used in combination with the existing field names (which are the centre IDs). The “H” is for “Huff” and the “i” is for “consumer”. Caution needs to be taken with the choice of field names and their length. Shapefiles have limits on their table field names of less than 10 characters. With two for the prefix, this leaves 8 characters for the original name. The field names are used as keys by the script and it is important that they remain as expected (and not abbreviated by QGIS to correct for length limitations). This is more likely to be an issue with identifiers in the data for centres as these are the ones that end up as field names.

To review some of the characteristics of the data output for the script, an export file has been created and a new screen capture of a spreadsheet is presented below.

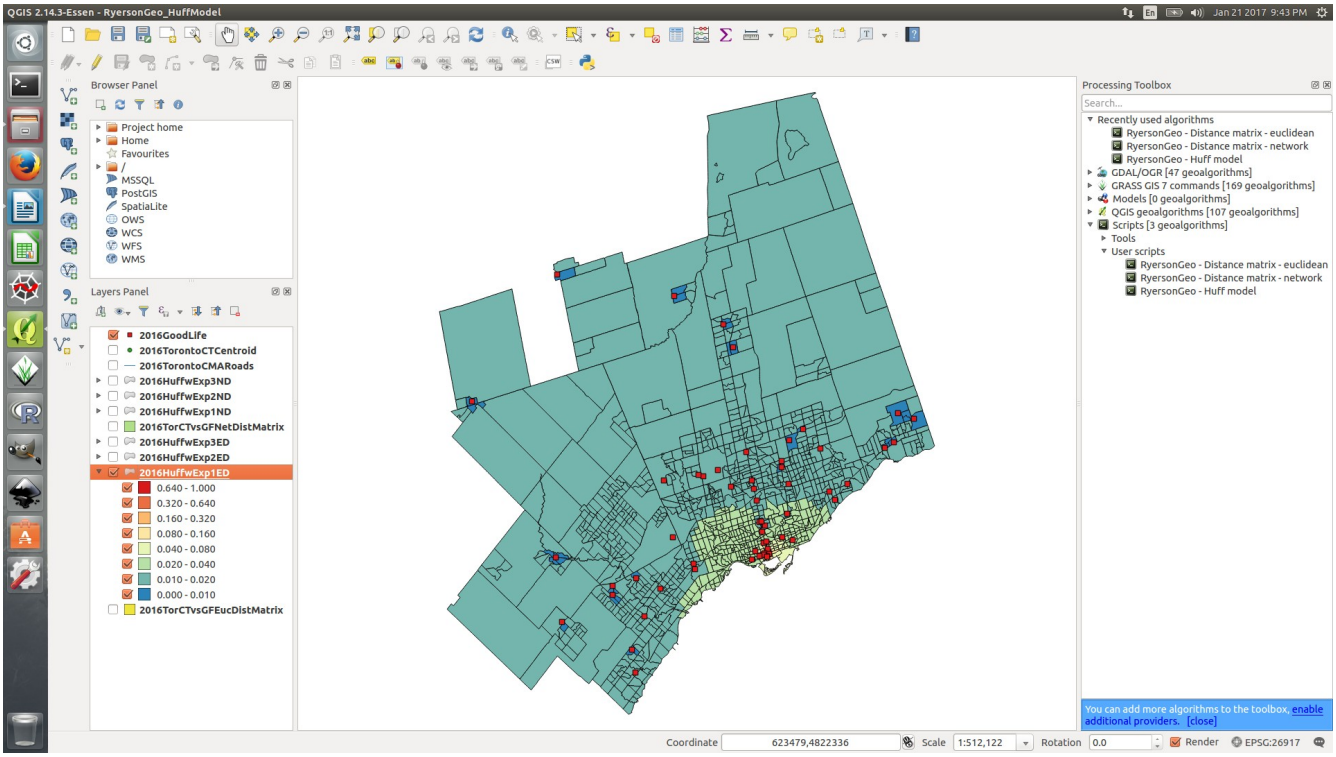


As discussed earlier, when considering the probabilities generated for the Huff Model, it is expected that the sum of the probabilities for a consumer shopping at all centres will sum to 1. Adding a totals column to the spreadsheet and summing the probabilities of a census tract across centres shows that this is true and that the script is performing as expected.

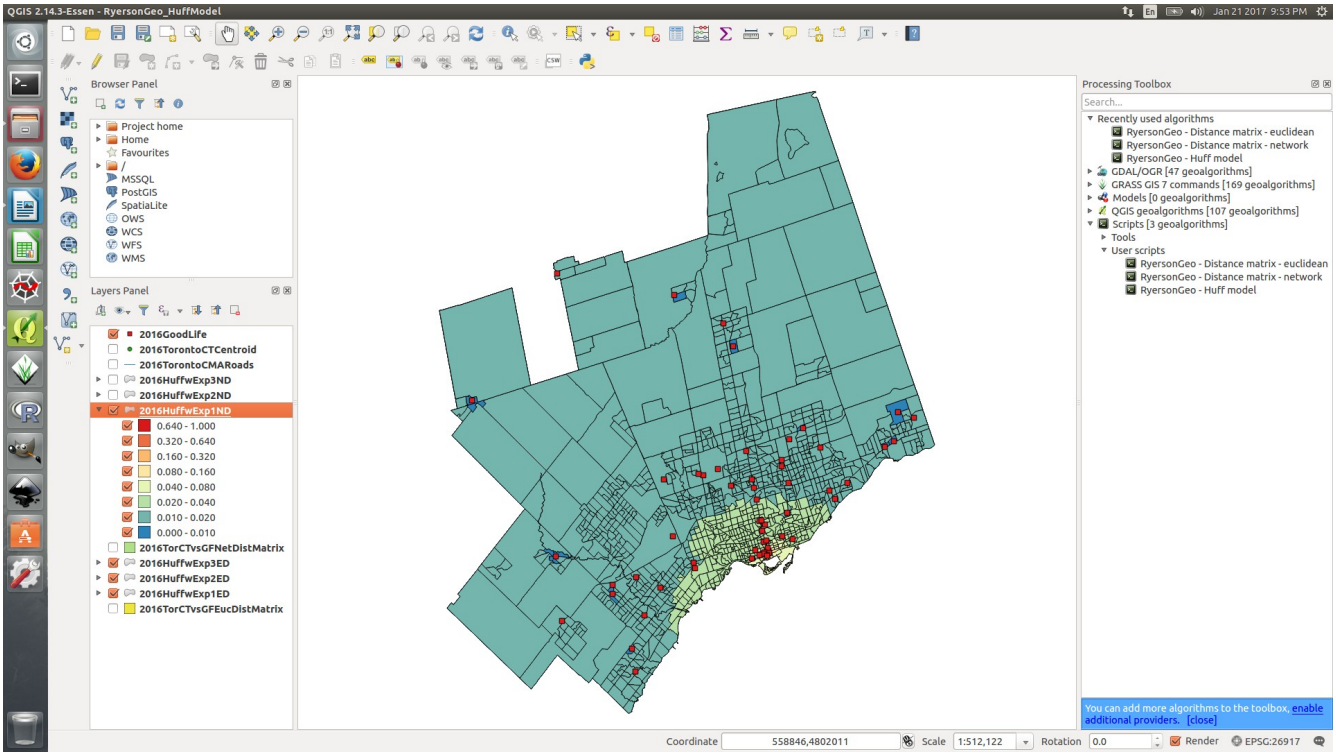
Visualizing Huff Model Output with Maps

A sample of maps for the Huff Model probabilities produced are presented below in screen captures. Each map uses the same scale to ensure each is comparable and may be interpreted in the same way. Eight classes are used with blues covering the lower values and reds covering the higher values. Having a consistent scale for all maps is a challenge as the probability values presented cover such a wide range of values and the detail can easily be lost if not enough classes are used. To keep the number of classes manageable, a non-linear scale has been applied. Two to the power of n divided by 100 has been used to set the upper bounds of each class. That is: 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, and 1.00 (1.00 should be 1.28, but the highest possible value for a probability is 100% or 1.00). This scale seems to work for this data. See the screen captures below for centres identified by GF1002 and GF1027. Each of the six Huff probability calculations have been mapped.

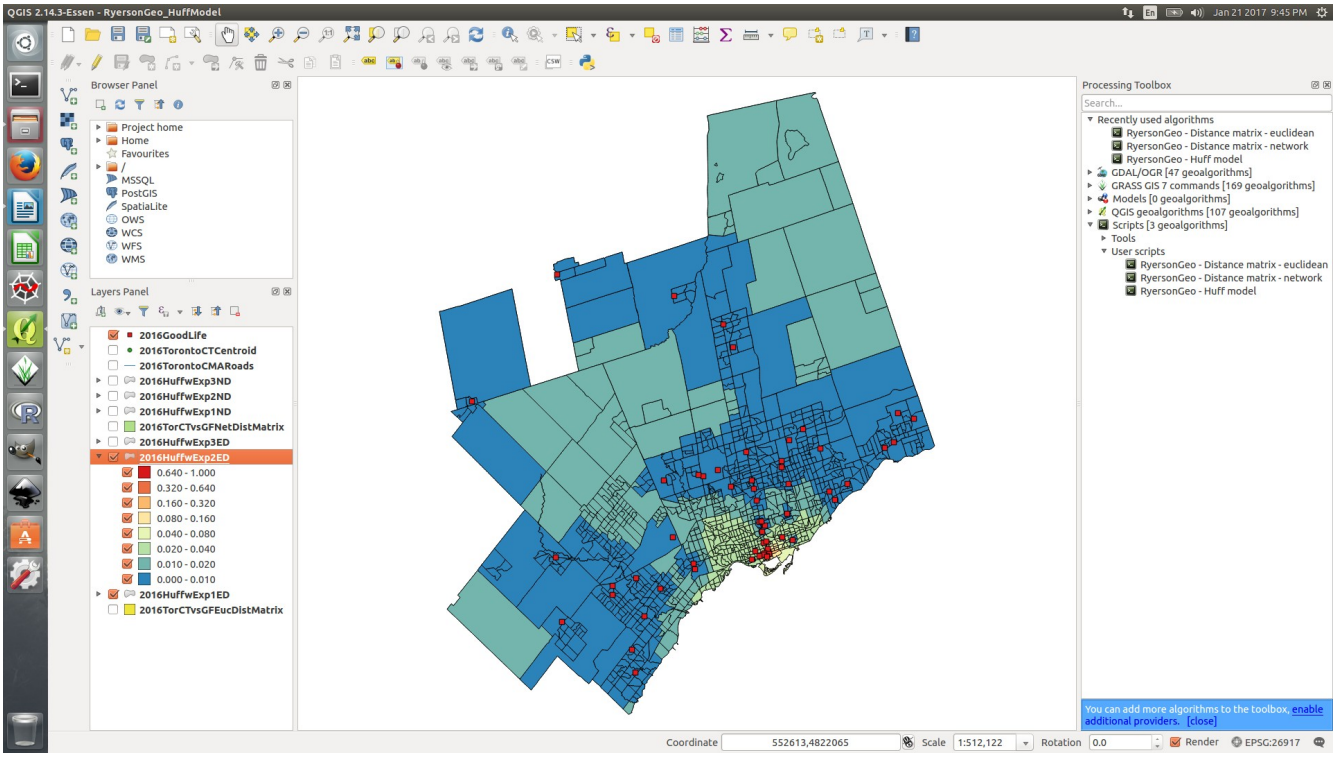
GF1002 – Euclidean Distance, Huff Exponent=1:



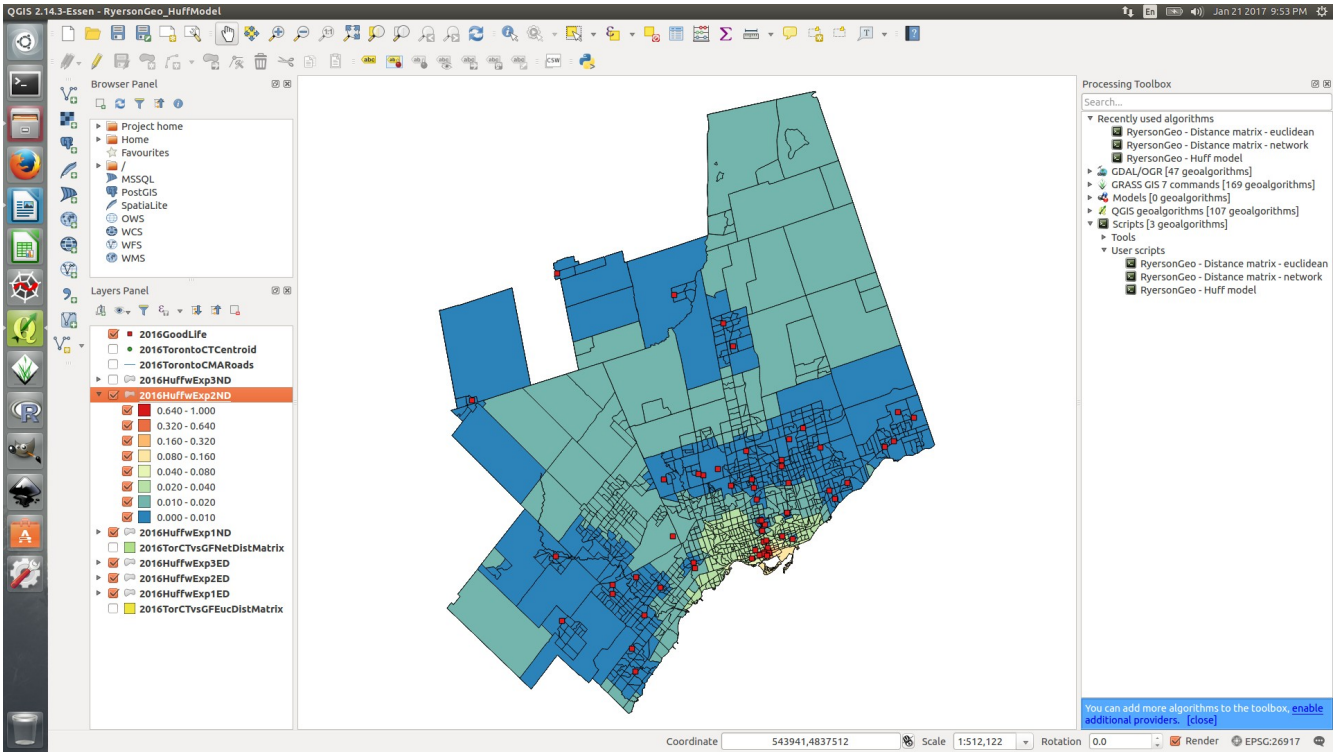
GF1002 – Network Distance, Huff Exponent=1:



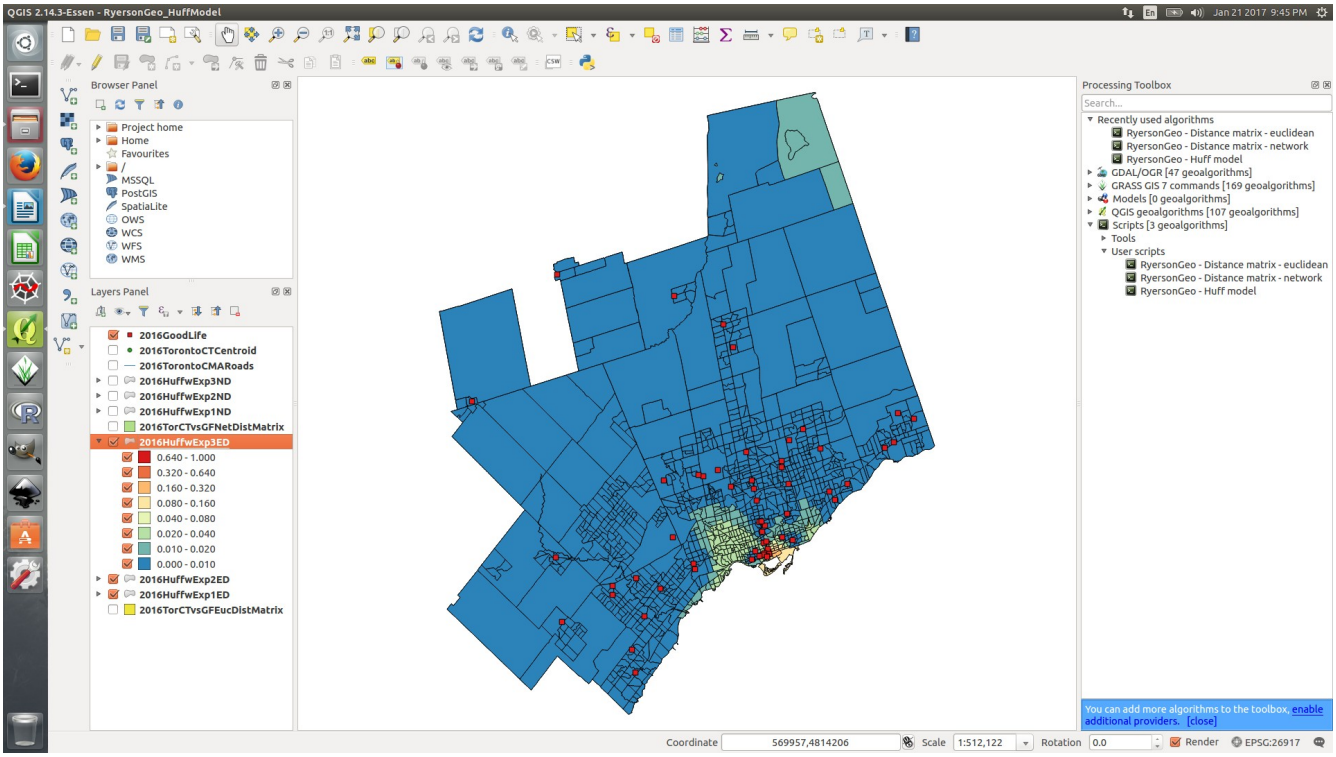
GF1002 – Euclidean Distance, Huff Exponent=2:



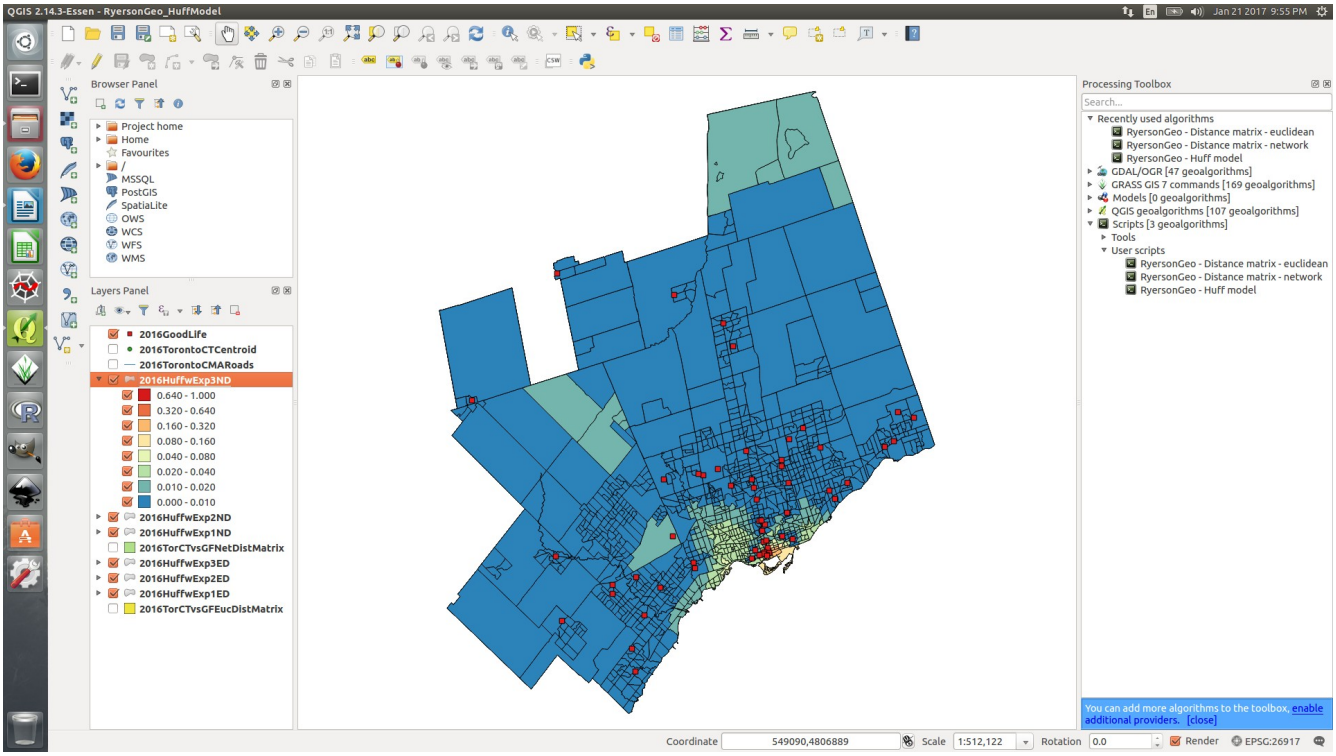
GF1002 – Network Distance, Huff Exponent=2:



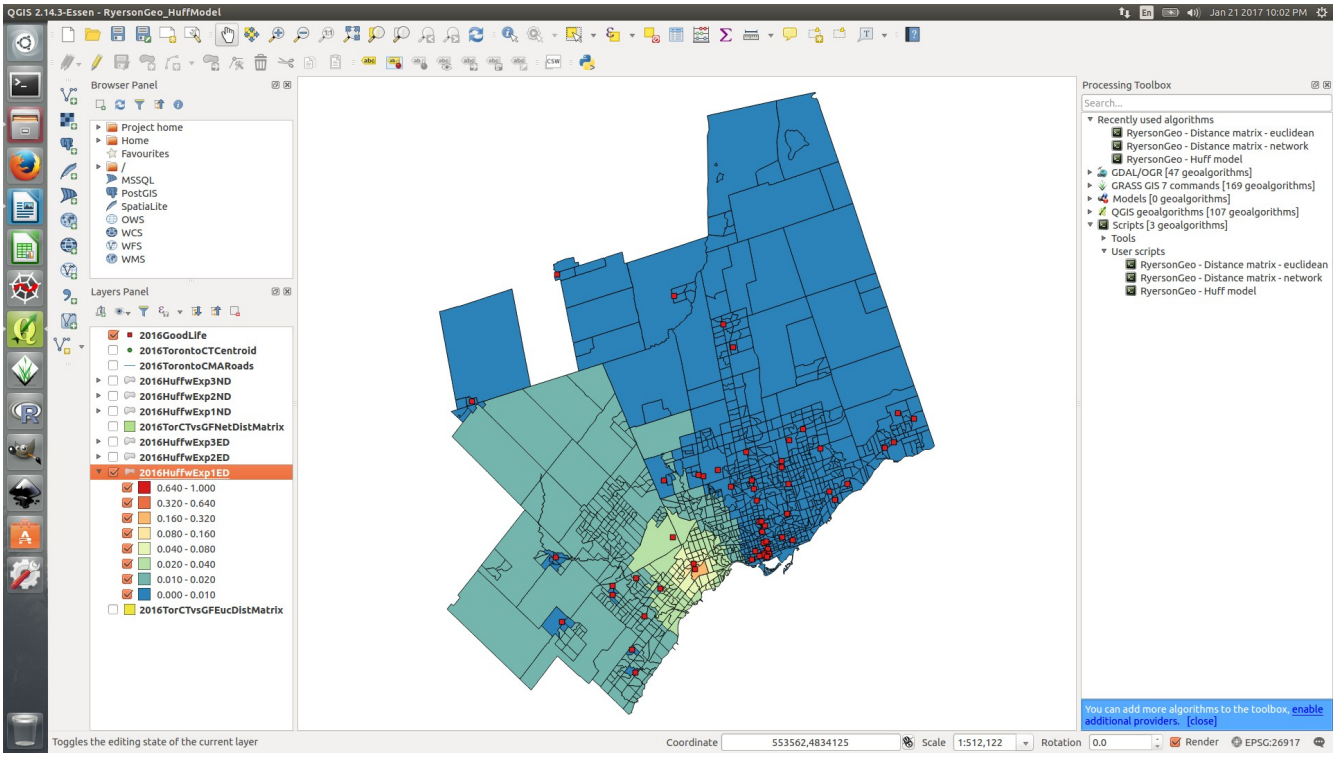
GF1002 – Euclidean Distance, Huff Exponent=3:



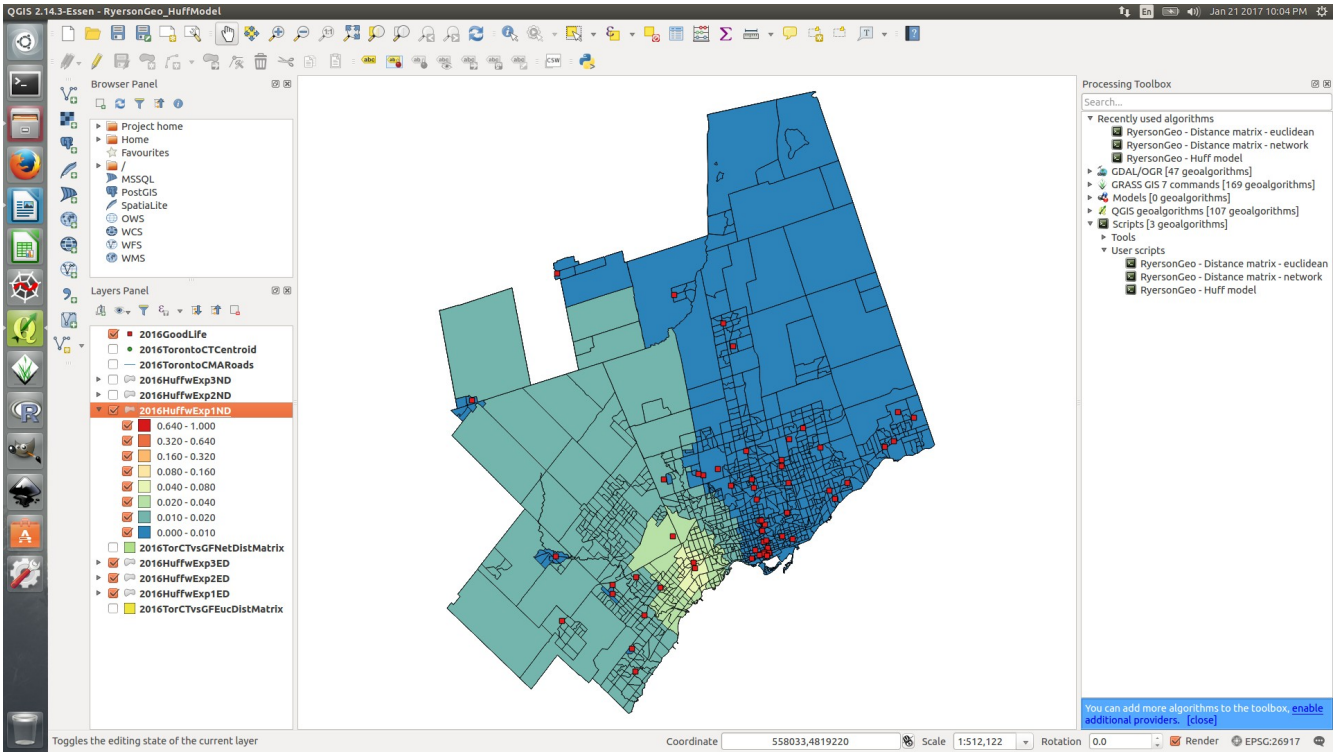
GF1002 – Network Distance, Huff Exponent=3:



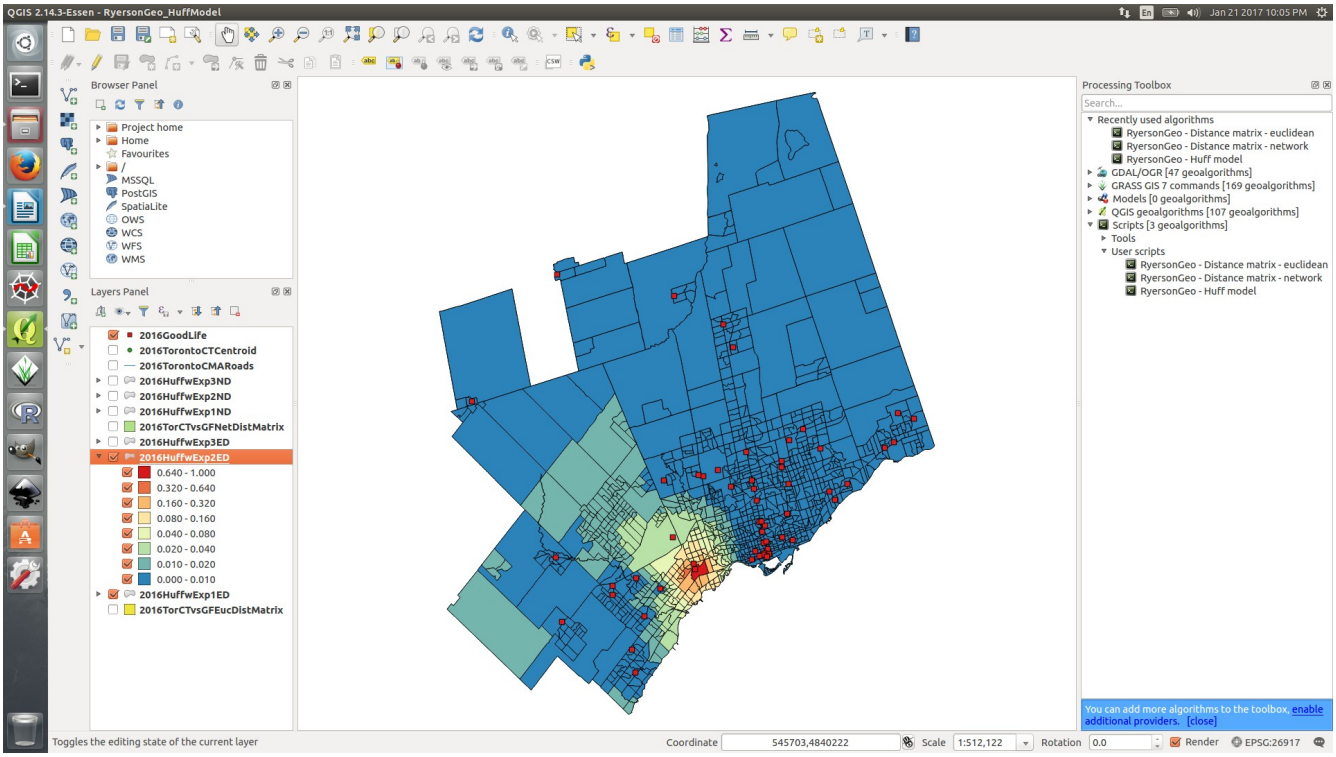
GF1027 – Euclidean Distance, Huff Exponent=1:



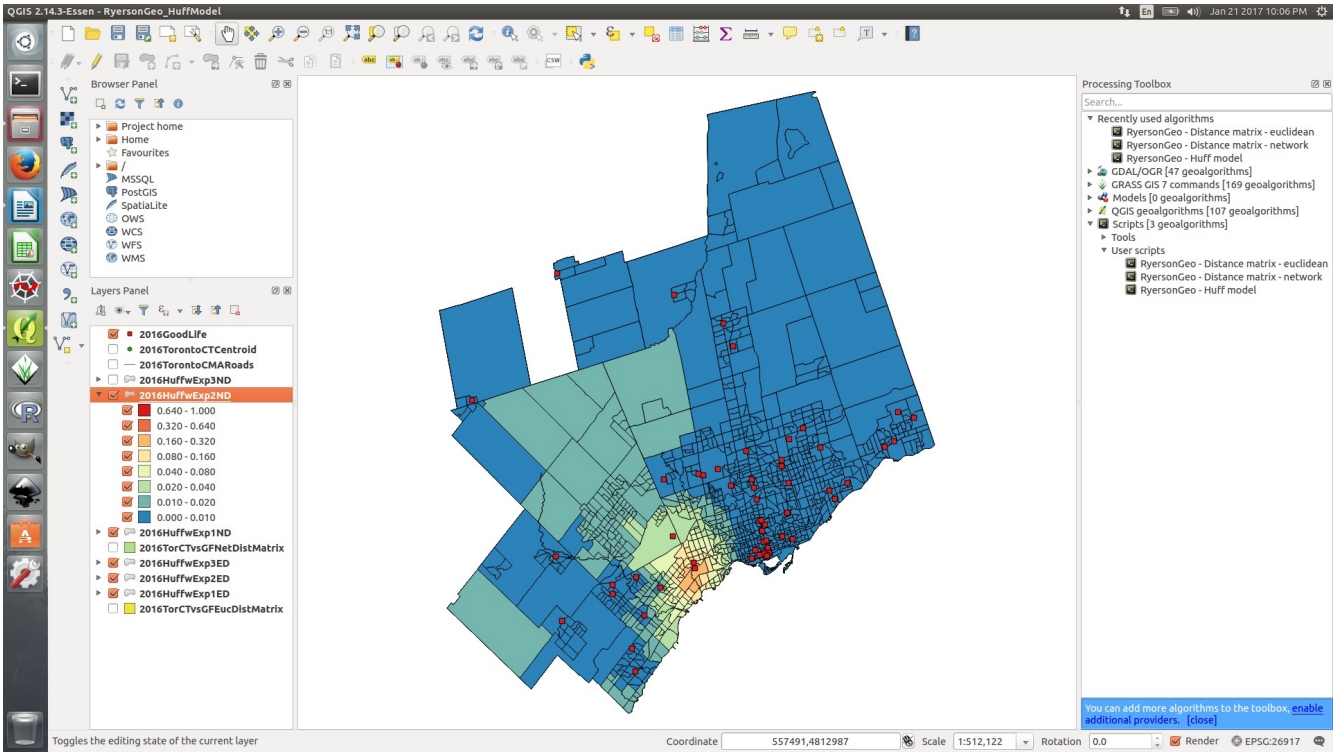
GF1027 – Network Distance, Huff Exponent=1:



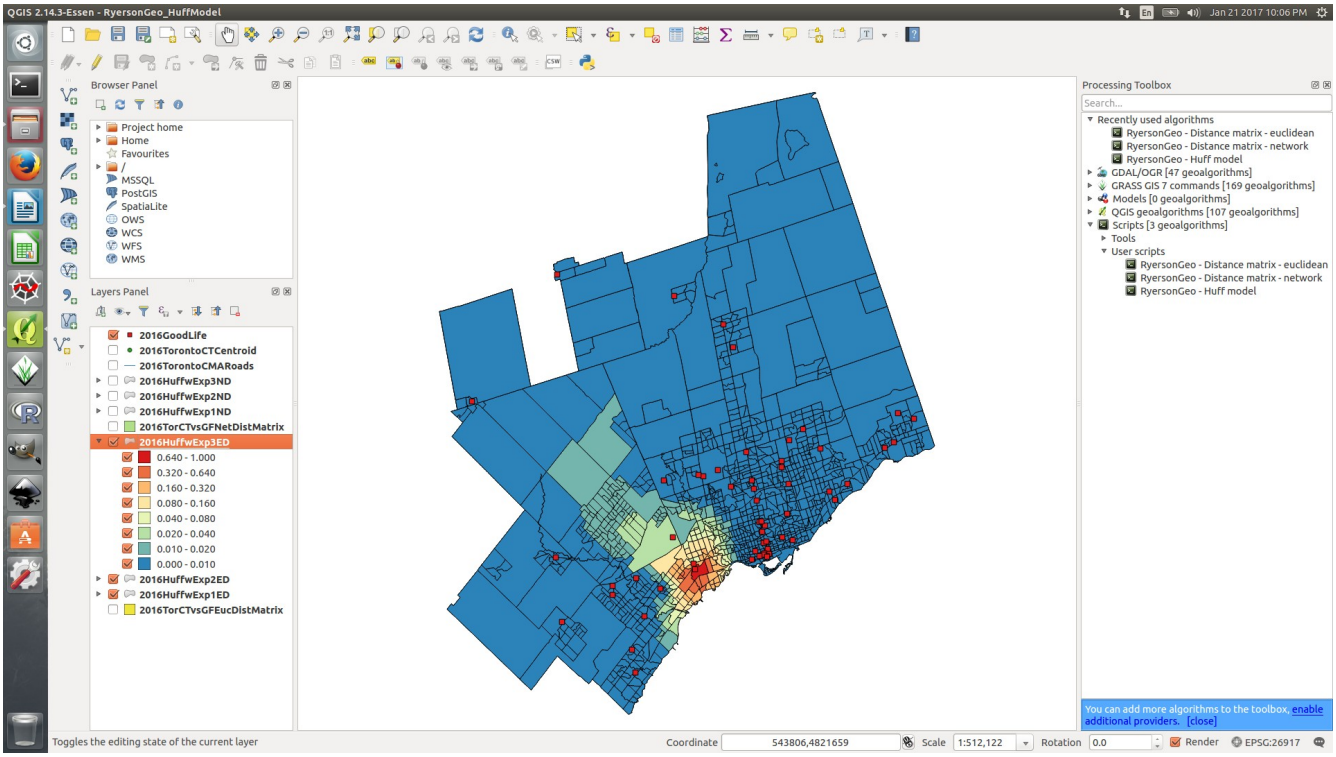
GF1027 – Euclidean Distance, Huff Exponent=2:



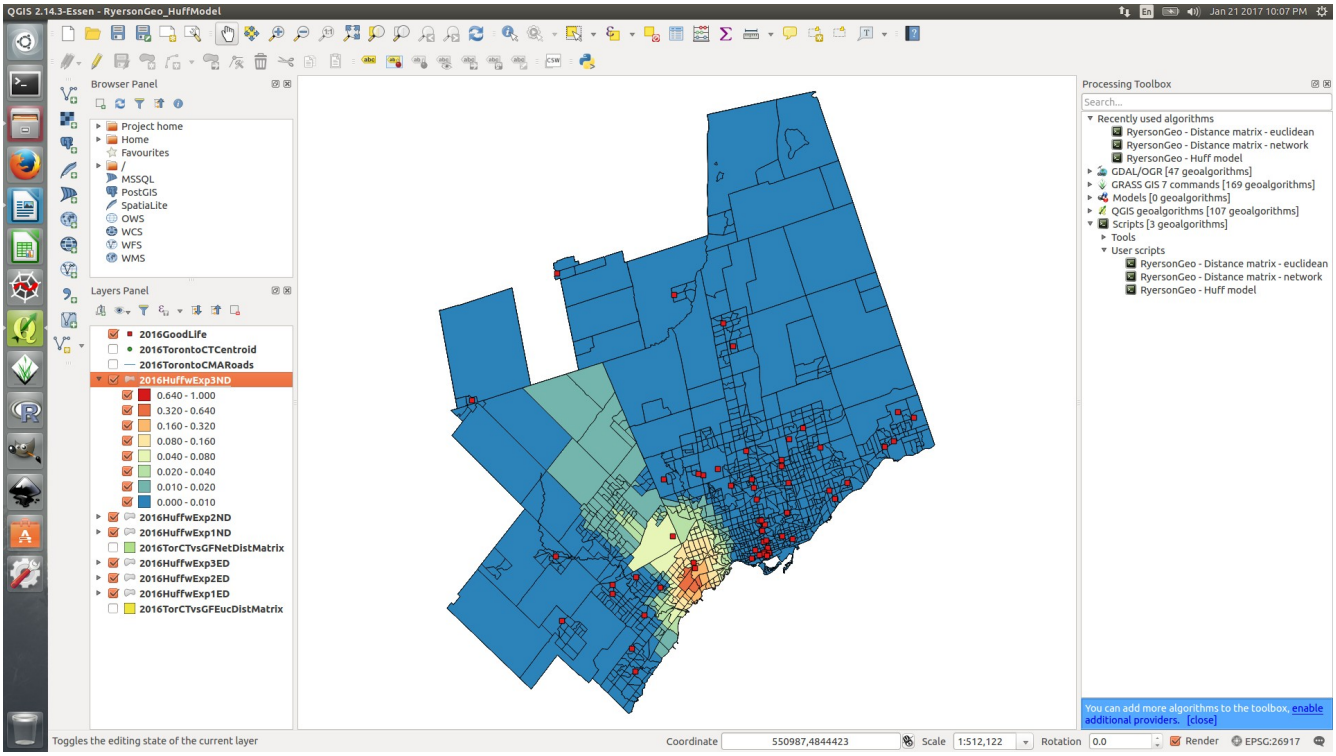
GF1027 – Network Distance, Huff Exponent=2:



GF1027 – Euclidean Distance, Huff Exponent=3:



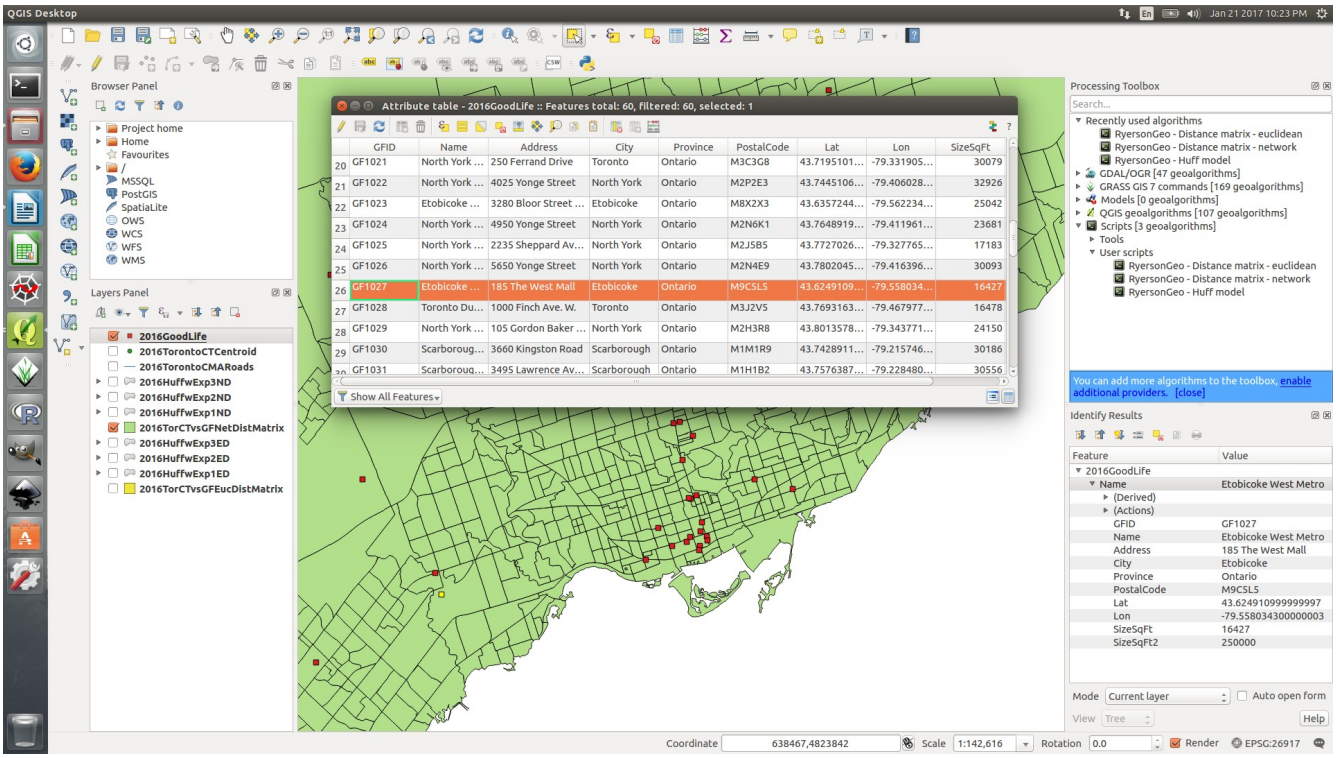
GF1027 – Network Distance, Huff Exponent=3:



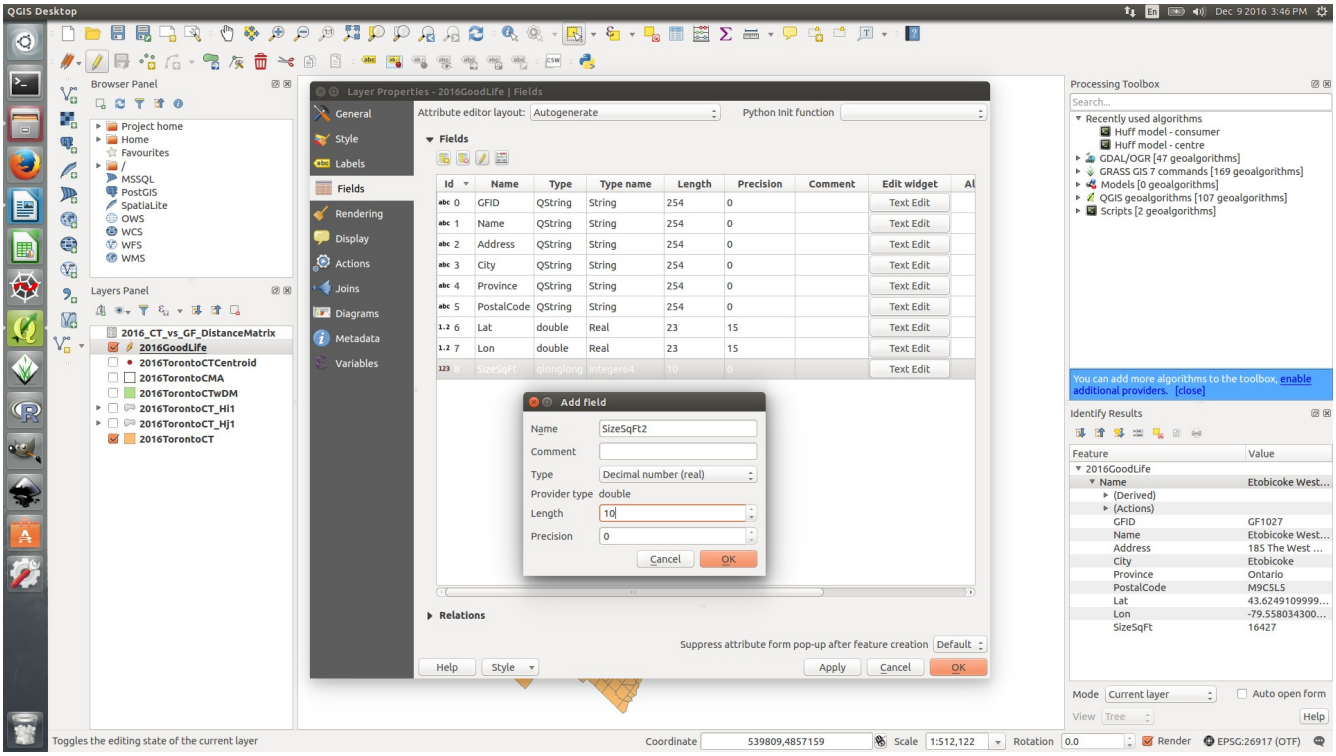
Step 16: Adding Utility Variables for “What-if” Scenario Exploration

Let's assume that Goodlife Fitness has decided to invest in the Etobicoke West Metro club, this is GF1027 (see the maps above). This location is shown as a yellow dot on the map in the next screen capture. Head Office wants to know what the impact will be on their other locations if they convert this location into a 250,000 sqft mega club. This is explored with the Huff Model Python script in QGIS.

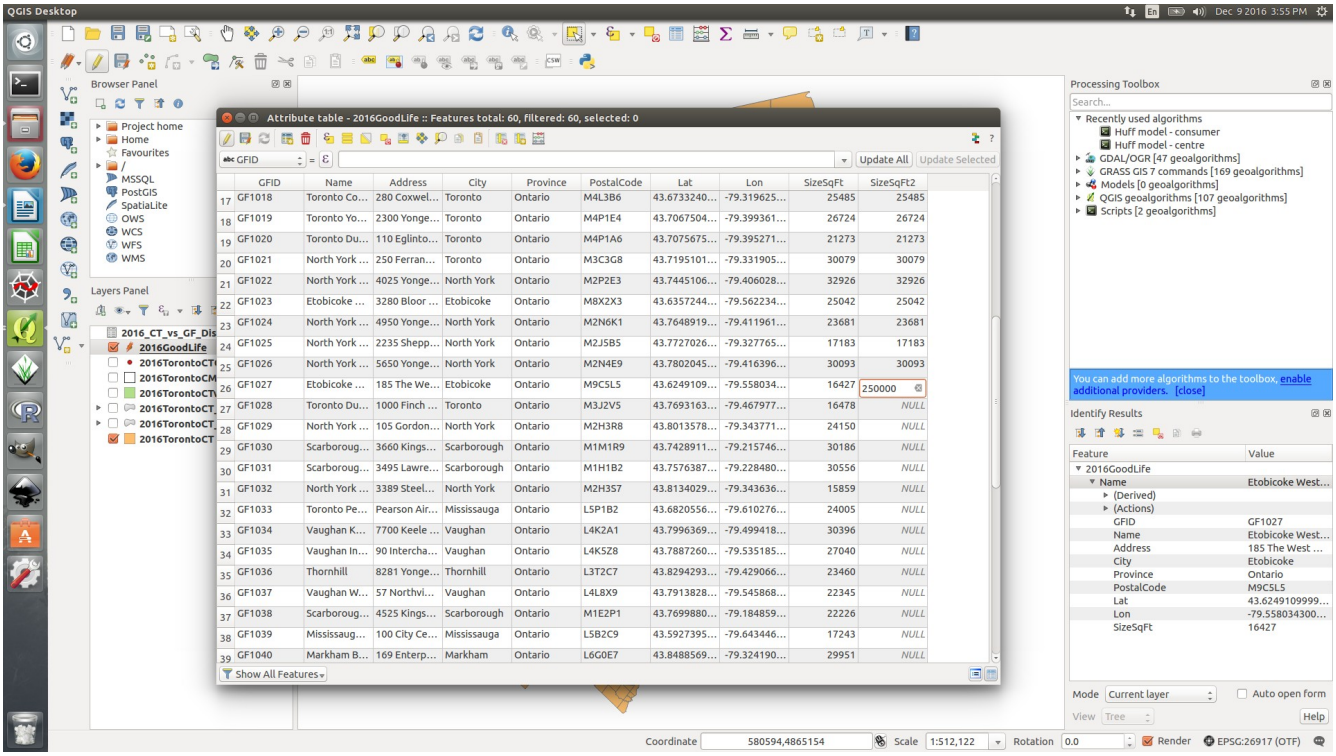
To consider this problem, not much more is required of the data that is already in place. The distance matrix shapefile (representing the consumer) prepared in either Step 13 or Step 14 does not need to change as it holds the census tract boundaries and the distance matrix values. With all the clubs remaining in their existing locations, this file is still valid. If new clubs are opened or if existing clubs are closed, then this file would need to be recreated to reflect the change. The second input for this Python script, the centres shapefile, required editing; this is the data for the Goodlife Fitness locations. As the original locations are being considered in place, this file can be modified from the current form with the editor in QGIS. If locations need to be added, removed, or relocated; then the file could either be recreated from a .csv file or edited with “heads-up” digitizing techniques. To complete the “what if”, a second size column has been added to the file with the same values for all clubs except for GF1027. This location's new size has been set to a value of 250,000. This editing process is reviewed below.



In the layer properties for the Goodlife Fitness locations, a new field is added on the “Fields” tab. See the next screen shot. The field type and length are matched to the exiting SizeSqFt field.

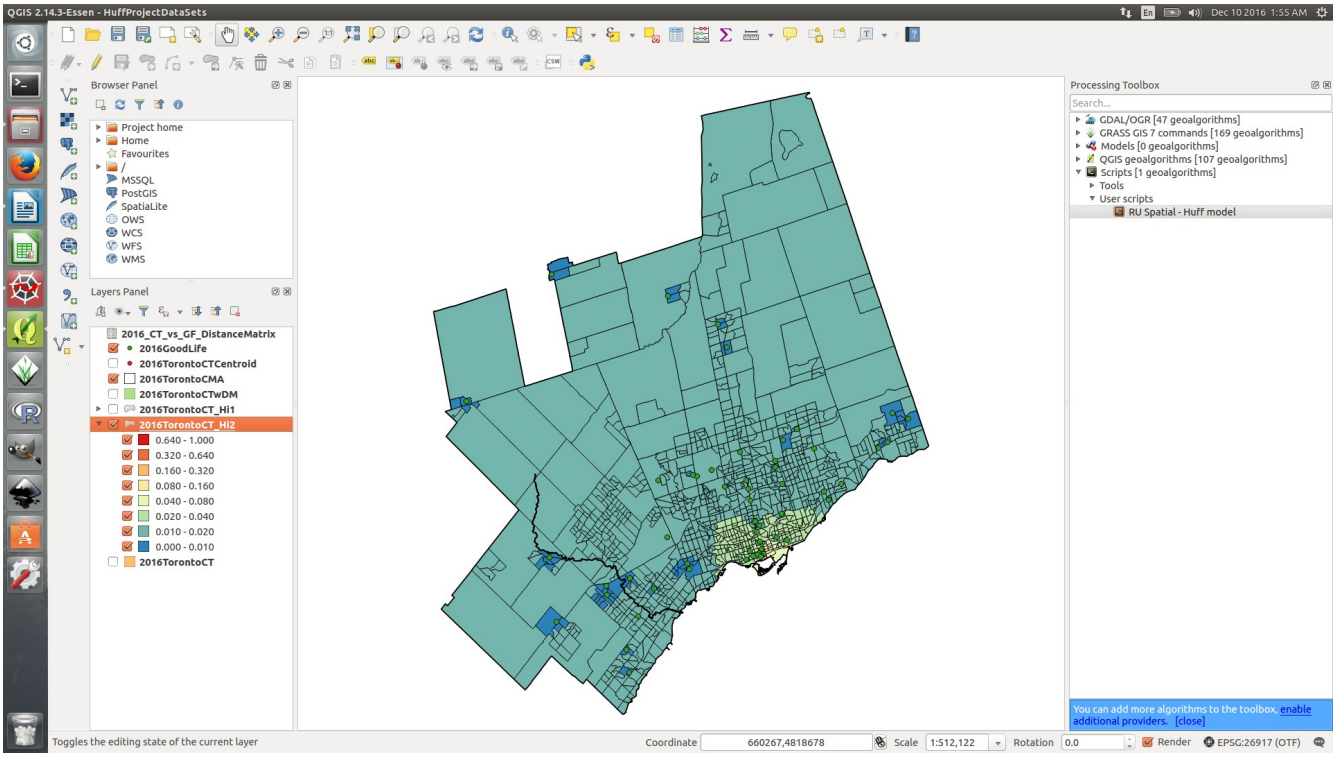


Using the editor in the attribute table, the values in the original field are re-keyed into the new column. There are likely other ways to do this, but for 60 values, this is probably quicker. The next screen capture shows the editing process.

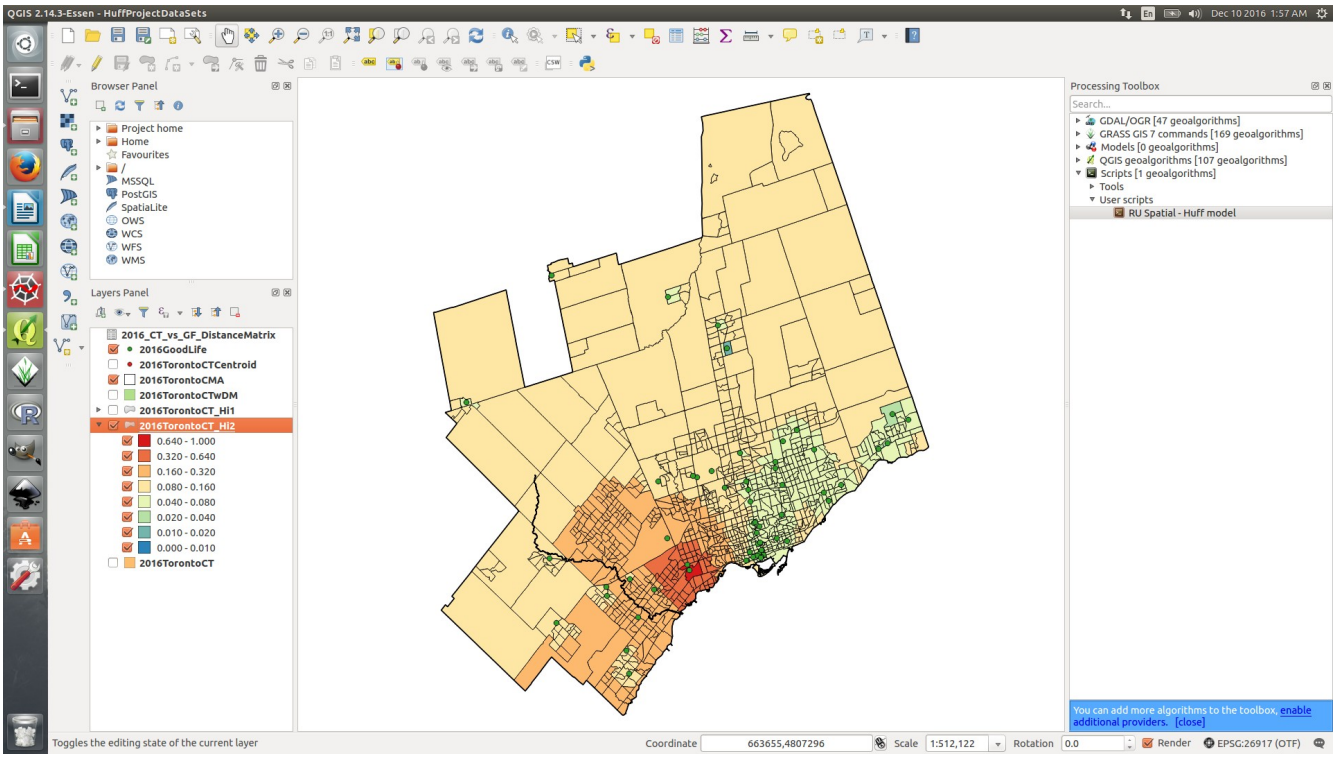


The Huff Model script may be run again to explore the new attractiveness variable. To do this six new "empty" output files should be created to record the results for each combination of distance matrix (euclidean or network) and choice of Huff exponent. Visualizations for GF1002 and GF1027 are presented below. These are for the euclidean distance matrix and the Huff exponent set to 1. These may be compared to the maps appearing earlier.

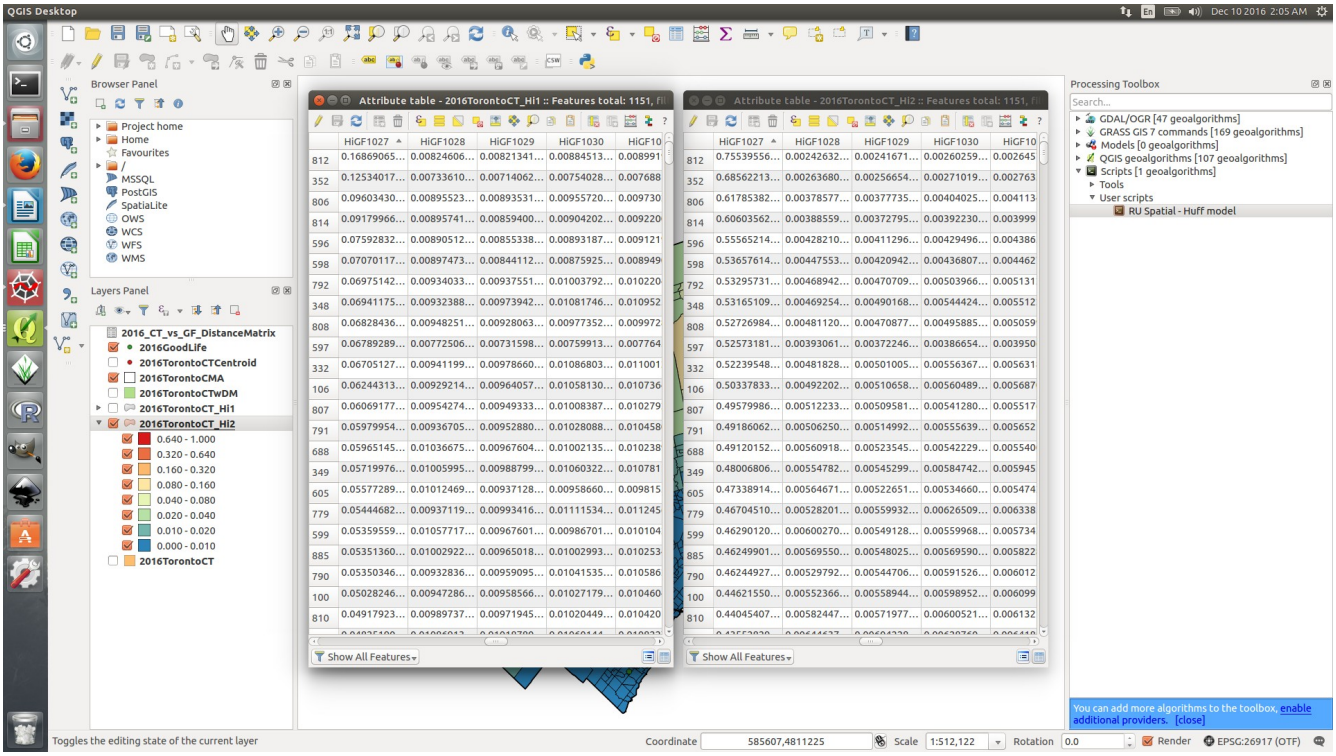
GF1002 – Euclidean Distance, Huff Exponent=1, Club Size Unchanged:



GF1027 – Euclidean Distance, Huff Exponent=1, Club Size Change to 250,000 sqft:



Attribute table values for GF1027:



As can be observed by reviewing the maps generated with the new size data, the mega club (GF1027) will have an impact on both its own status in the market and also on those clubs that surround it. The last screen capture above presents the attribute tables for GF1027 (see the left column in each table). The table on the left is for the original “Size” variable and the one on the right reflects the new 250,000 sqft facility. Scanning the numbers and reading down the GF1027 columns, it can be seen that this location has significantly strengthened it's position in the market. Reading across the rows, and comparing the two tables, it is observed that the improvement in GF1027 is at the expense of the other locations – it is harder for them to attract those close to GF1027 to their own locations. The probabilities for any given consumer (census tract) must sum to 1. If GF1027's share goes up, then the others must go down.

Part IV - Conclusions

QGIS provides an adequate platform for conducting Huff Model calculations and experiments. The majority of the functionality required for this work is available in a standard installation of QGIS 2.14 Essen and with a standard spreadsheet application. The functionality required to complete the Huff Model calculations themselves are not built-in standard and these have been provided for with the use of a custom-written Python script. To support the Huff Model Python script developed here, two additional Python scripts have been developed to generate distance matrices. All of these capabilities and the operation of the new scripts have been demonstrated here.

It was initially suggested that a Python plugin be used for the task of providing capabilities related to the Huff Model. This goal is being worked towards through the intermediate steps of Python scripts. Python plugins are a lot more complex and require the development of a graphical user interface (GUI); they also need to be compiled. There is limited documentation to assist navigating this more complex path to the desired solution. The goal of a plugin should be achievable with some time to locate suitable documentation and research plugin development for QGIS.

The provided Python scripts are designed to provide capabilities based on the Huff Model as presented in academic textbooks. There are other variations of the Huff Model including an Advanced Huff Model and implementations by Esri in Business Analyst. Results from this work appear to match textbook results.

This work should be considered in the development stage. While the developed scripts appears to work well, more comprehensive testing should be completed. The script should also be enhanced to “idiot proof” it as much as possible. There is also some concern over the amount of time scripts based on the Dijkstra algorithm take to complete.

Moving forward, this work may be expanded to create additional Python code to provide functionality similar to those found in other implementations of the Huff Model. This may include allowing for multiple, weighted variables to be used as a measure of utility or size in the Huff Model formula. Work may also be completed to simplify or enhance the process by including additional capabilities in the Python code. An example of this is the improvement made to the process for the euclidean distance matrix preparation. The review completed here has focused on generating Huff Model results. Functionality, through the use of Python code, could also be produced to automate the *use* of the Huff Model results. This could take the form of visualization production or further downstream analysis.