Applied Data Science Master’s degree programme

Spatial Data Analysis and Simulation Modelling course

Instruction manual for Lab 3.3: **spatial networks and catchment area analysis**

Document version: 0.9

Document modified: 2020.10.06

Dr. Zhiyong Wang, Dr. Simon Scheider

Department of Human Geography and Spatial Planning

Faculty of Geosciences

z.wang2@uu.nl, s.scheider@uu.nl

Introduction

In lab 2.3, we learnt how to build a network and calculate the shortest route between 2 points. We can apply that technique for many different types of network-based analysis. One of these applications is to do catchment area analysis. A catchment area is the area from which an agency or institution attracts a population who uses its services. In this practical, we will learn how to compute an Origin-Destination (OD) Matrix to derive catchment areas, given a set of origin points and another set of destination points. We will calculate the shortest routes between each origin-destination pair and find out the travel distance/time between them. Such analysis is a basis for city planners and governments to define the areas that would have access to services like fire departments, police stations and hospitals.

The task of this assignment is to find the closest hospital for each postcode (PC4) area in the city of Amsterdam, and to generate shortest routes to the PC4 areas given a certain hospital. For this purpose, you can directly download the following datasets (in the zipped file *Lab 3.3 data.zip*) from blackboard:

- PC4 (shp)
- hospitals_ams_2007 (shp)
- roads_ams_2008 (shp)

Software to be used:

- QGIS
- QGIS Network Analysis Toolbox (QNEAT3) plugin

The learning goals of this assignment are:

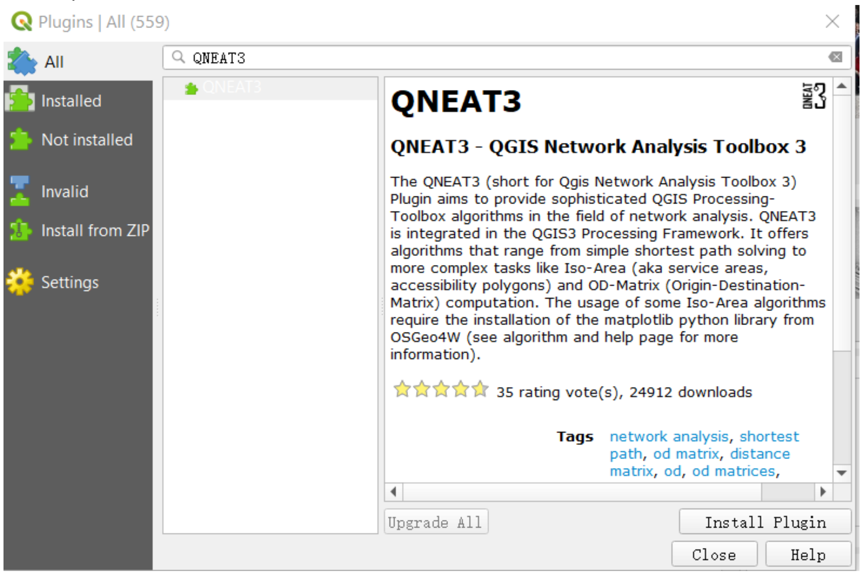
- Learn to extract centroids from a polygon layers
- Learn to calculate OD-matrix in QGIS
- Learn to perform network analysis in QGIS
- Lean to use Virtual Layers to run SQL query on a QGIS layer
- Learn to Use Python Console Editor to run a python script.

Setup

Install the QGIS plugin **QNEAT3**

1. Open QGIS, click **Plugins**, and select **Manage and Install plugins.**

2. Search for **QNEAT3** and install it



Plugins | All (559)

QNEAT3

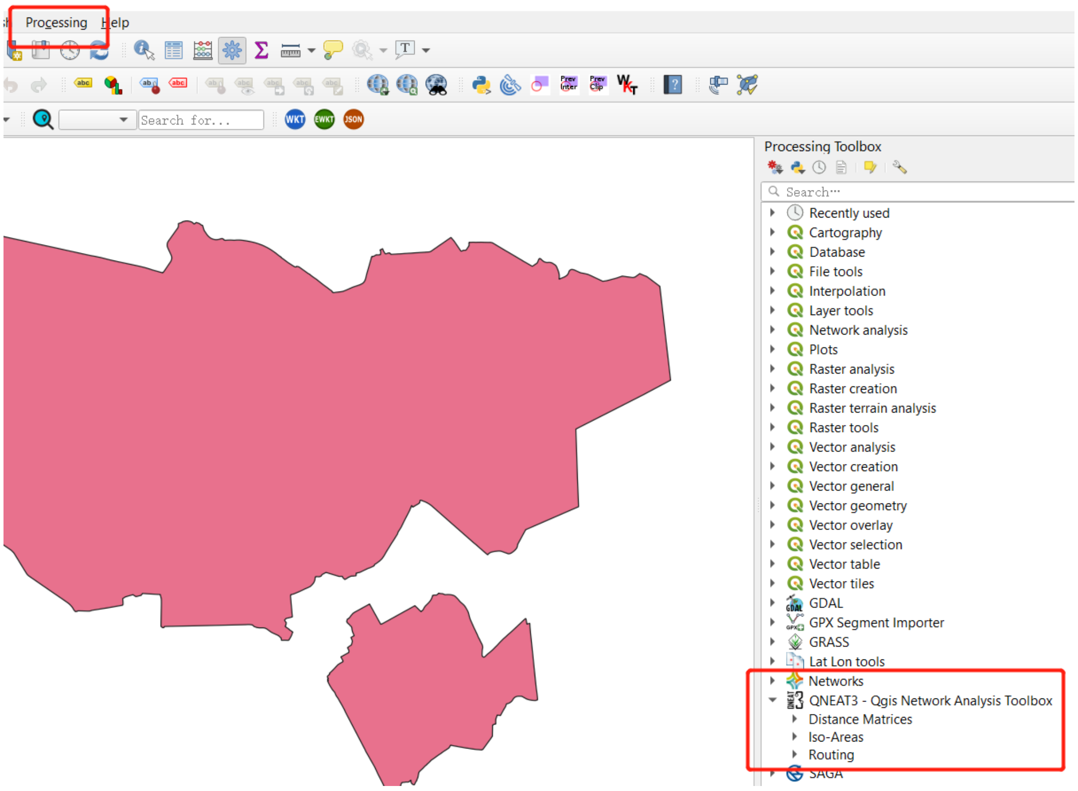
QNEAT3 - QGIS Network Analysis Toolbox 3

The QNEAT3 (short for Qgis Network Analysis Toolbox 3) Plugin aims to provide sophisticated QGIS Processing-Toolbox algorithms in the field of network analysis. QNEAT3 is integrated in the QGIS3 Processing Framework. It offers algorithms that range from simple shortest path solving to more complex tasks like Iso-Area (aka service areas, accessibility polygons) and OD-Matrix (Origin-Destination-Matrix) computation. The usage of some Iso-Area algorithms require the installation of the matplotlib python library from OSGeo4W (see algorithm and help page for more information).

35 rating vote(s), 24912 downloads

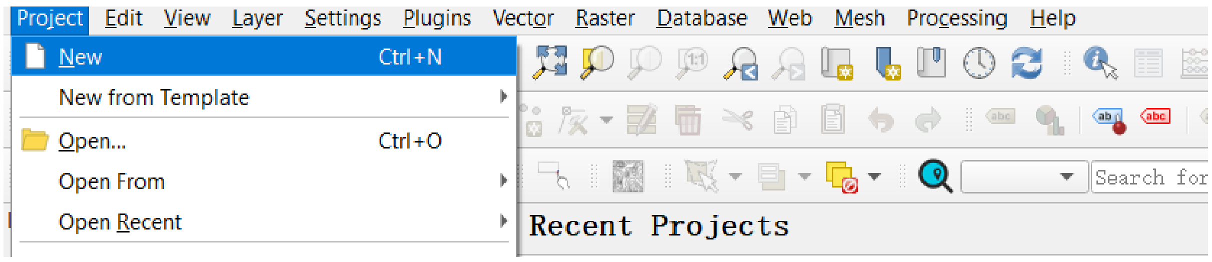
Tags network analysis, shortest path, od matrix, distance matrix, od, od matrices,

3. After the QNEAT3 is installed, click **Processing** and you can find the QNEAT3 plugin in the **Processing Toolbox**

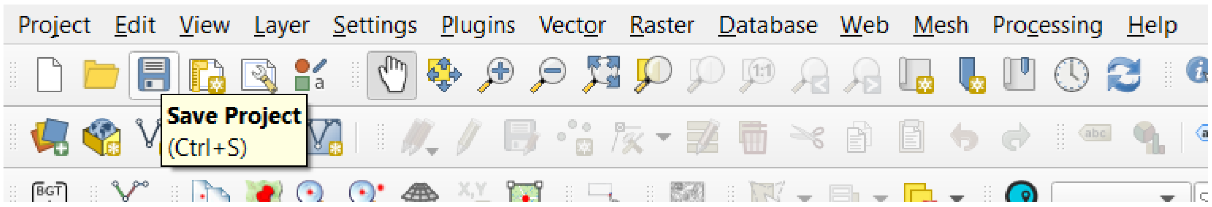


Prepare the software and load the data

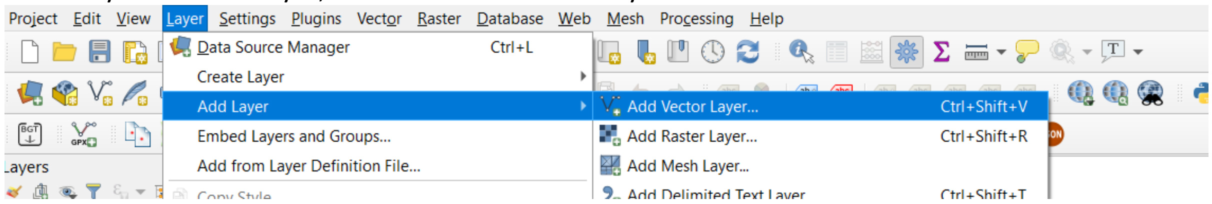
1. Create a new project in QGIS



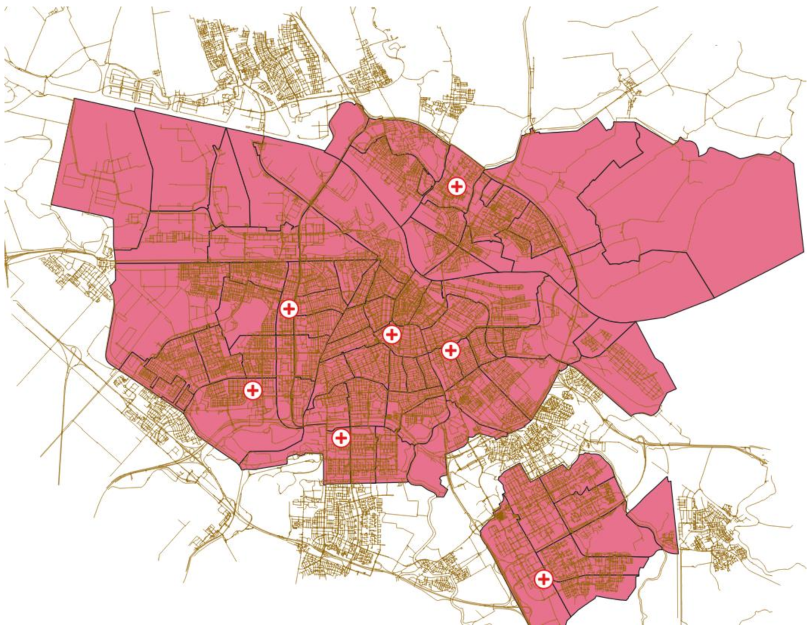
2. Save the new project on your local computer



3. Go to Layer and Add Layer, and select Add Vector Layer


Project
Edit
View
Layer
Settings
Plugins
Vector
Raster
Database
Web
Mesh
Processing
Help
Data Source Manager Ctrl+L
Create Layer
Add Layer
Embed Layers and Groups...
Add from Layer Definition File...
Copy Style
Add Vector Layer... Ctrl+Shift+V
Add Raster Layer... Ctrl+Shift+R
Add Mesh Layer...
Add Delimited Text Layer Ctrl+Shift+T
Layers

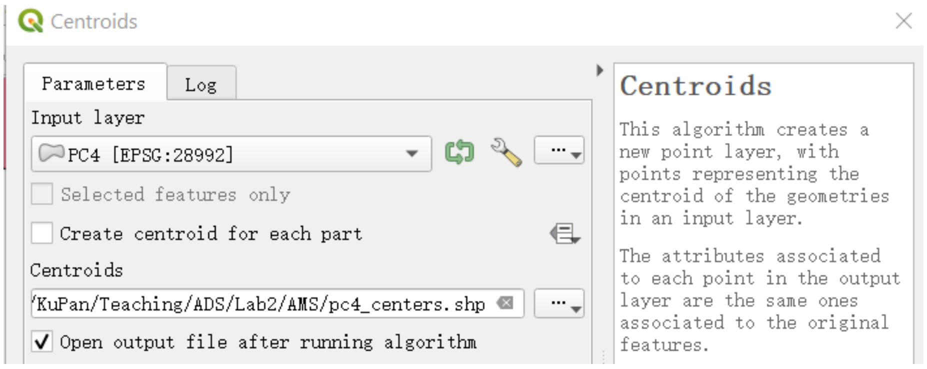
4. Locate the downloaded data file, and add **PC4.shp, hospitals_ams_2013.shp** and **roads_ams_2008.shp** to the canvas. You will see three layers in QGIS. Each point represents a fire station. Each polygon corresponds to a 4-digit postcode area. Each line represents a street in the city of Amsterdam. Make sure that you are using the EPSG:28992 Projected coordinate system for the Netherlands in your project settings



Calculate OD-Matrix

To compute the OD matrix, we will need a set of origins and a set of destinations. For simplicity reasons, here we assume that people travel from the centroids of PC4 areas towards hospitals. So in this practical, we use the centroids of PC4 areas as origin points, and use hospitals as destination points. Note that for other services, such as ambulances, police stations, and fire stations, they travel in the opposite direction, towards the incident places.

First, we extract centroids of the PC4 areas. Go to the menu -> **Vector** -> **Geometry tools** -> **Centroid** and select **create a centroid point layer**, and save it as a shp file (**pc4_center.shp**).


Centroids

Parameters | Log

Input layer: PC4 [EPSG:28992]

☐ Selected features only

☐ Create centroid for each part

Centroids: 'KuPan/Teaching/ADS/Lab2/AMS/pc4_centers.shp

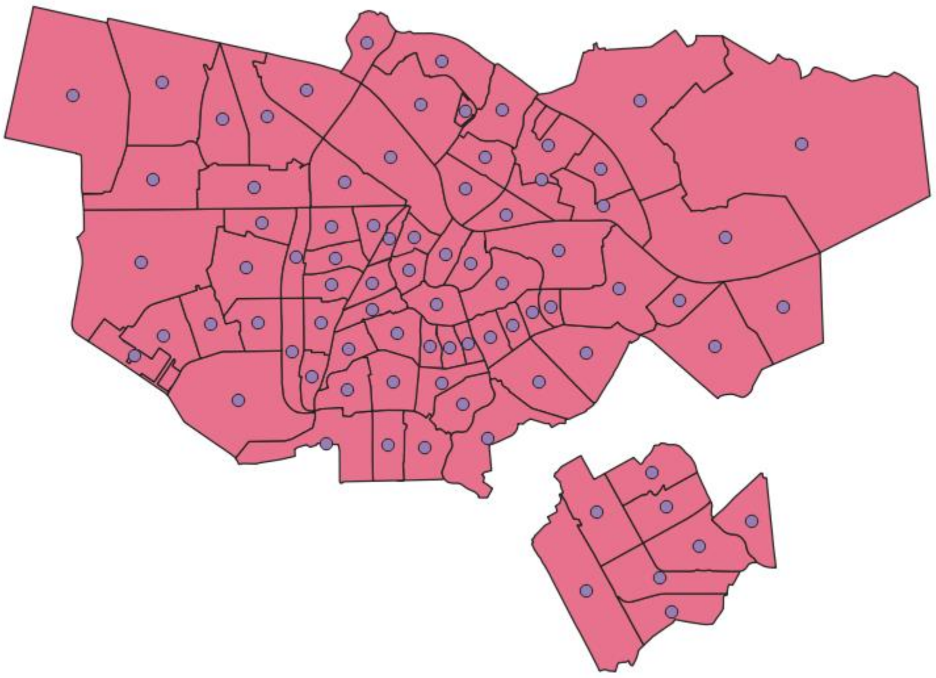
✓ Open output file after running algorithm

Centroids

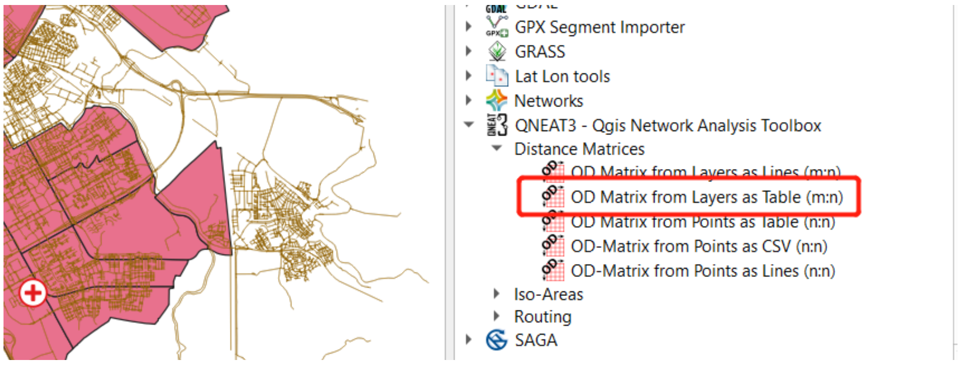
This algorithm creates a new point layer, with points representing the centroid of the geometries in an input layer.

The attributes associated to each point in the output layer are the same ones associated to the original features.

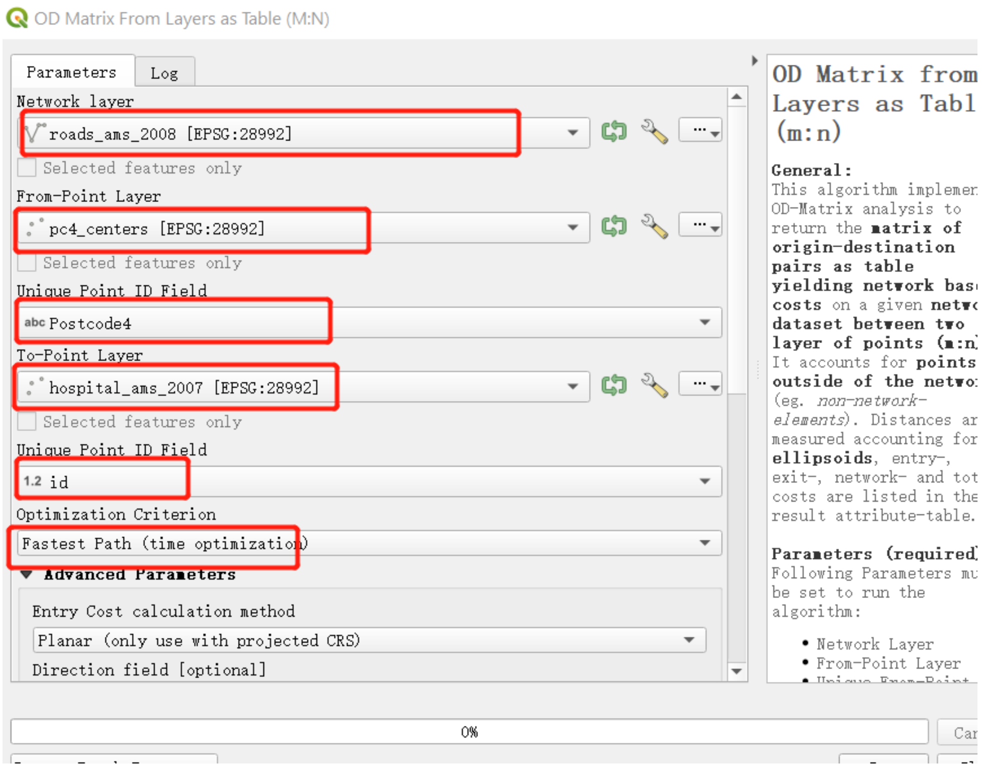
Click **Run** and you will see a new layer with the centroids of PC4 areas.



Go to **QNEAT3** --> **Distance matrices** and select **OD Matrix from Layers as Table (m:n)** algorithm. If you do not see this algorithm in the toolbox, make sure you have installed the QNEAT3 plugin.


GDAL
GPX Segment Importer
GRASS
Lat Lon tools
Networks
QNEAT3 - Qgis Network Analysis Toolbox
Distance Matrices
OD Matrix from Layers as Lines (m:n)
OD Matrix from Layers as Table (m:n)
OD Matrix from Points as Table (n:n)
OD-Matrix from Points as CSV (n:n)
OD-Matrix from Points as Lines (n:n)
Iso-Areas
Routing
SAGA

This OD Matrix algorithm finds the distances along the network between selected origin and destination points. Select **roads_ams_2008** as the Network layer. Select **pc4_centers** as the From-Points layer and **Postcode4** as the Unique Point ID field. Similarly, set **hospitals_ams_2013** as the To-Points Layer and **id** as the Unique Point ID field. Set the Optimization Criterion as **Fastest Path.**



Network layer
roads_ams_2008 [EPSG:28992]
Selected features only
From-Point Layer
pc4_centers [EPSG:28992]
Selected features only
Unique Point ID Field
abc Postcode4
To-Point Layer
hospital_ams_2007 [EPSG:28992]
Selected features only
Unique Point ID Field
1.2 id
Optimization Criterion
Fastest Path (time optimization)
Advanced Parameters
Entry Cost calculation method
Planar (only use with projected CRS)
Direction field [optional]

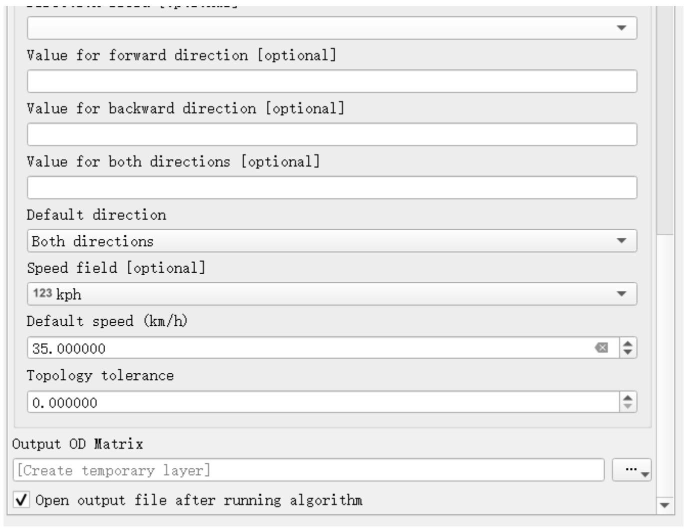
OD Matrix from Layers as Tabl (m:n)

General:
This algorithm implemen OD-Matrix analysis to return the **matrix of origin-destination pairs as table yielding network bas costs** on a given **netw dataset between two layer of points (m:n** It accounts for **points outside of the netwo** (eg. *non-network-elements*). Distances ar measured accounting for **ellipsoids**, entry-, exit-, network- and tot costs are listed in the result attribute-table.

Parameters (required
Following Parameters mu be set to run the algorithm:

- Network Layer
- From-Point Layer

Since we are using Dutch coordinate reference systems, we select **planar (only use with projected CRS)** to calculate the cost (length) of each street. For simplicity, we assume that all roads are available in both directions. Set **kph** as speed field and leave other options as default.


Value for forward direction [optional]

Value for backward direction [optional]

Value for both directions [optional]

Default direction

Both directions

Speed field [optional]

123 kph

Default speed (km/h)

35.000000

Topology tolerance

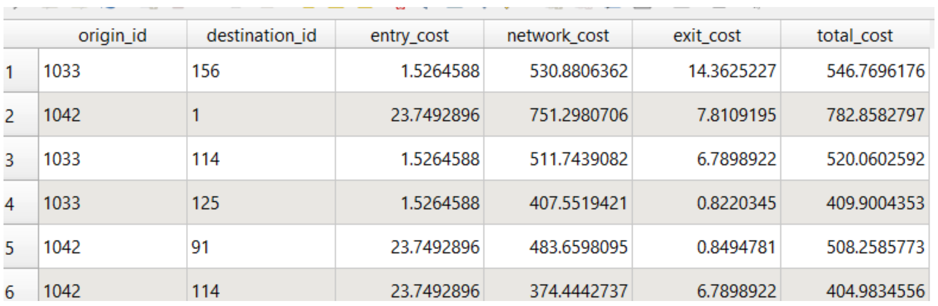
0.000000

Output OD Matrix

[Create temporary layer]

✓ Open output file after running algorithm

Click **Run**, and after a few minutes, you will see a new layer **Output OD matrix.** Right-click this new layer and select **Open Attributes Table**, you can see the distances between origins and the destination layer. The table contains 560 rows. We had 80 origin points (PC4s) and 7 destination points (hospitals), so the output contains 80x7 = 560. Because the centers of the postal areas are not coinciding with nodes in the road network, the algorithm also searches the road segment that comes closest to each center point as an entry point to the network. The **enry_cost** is the travel cost from the origin to the nearest road segment. Similarly, the **exit_cost** is the travel cost to the destination from the nearest road segment. The **total_cost** column contains the total cost (travel duration) between each origin point to every destination point.

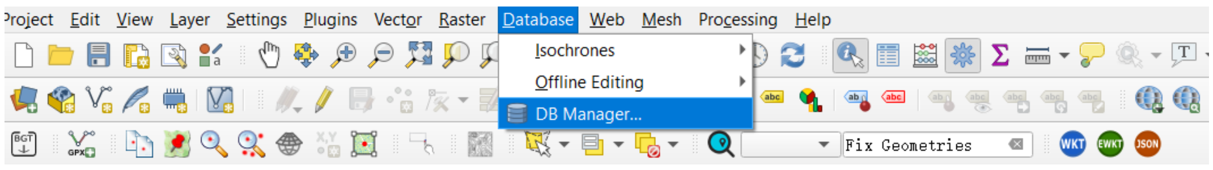


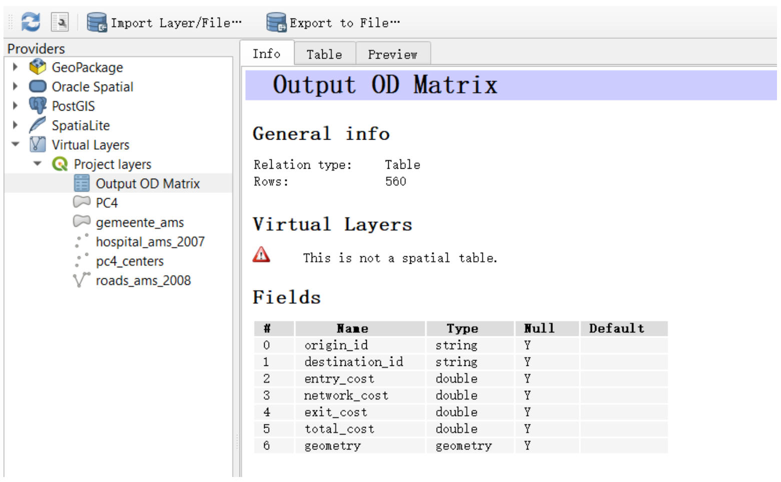
	origin_id	destination_id	entry_cost	network_cost	exit_cost	total_cost
1	1033	156	1.5264588	530.8806362	14.3625227	546.7696176
2	1042	1	23.7492896	751.2980706	7.8109195	782.8582797
3	1033	114	1.5264588	511.7439082	6.7898922	520.0602592
4	1033	125	1.5264588	407.5519421	0.8220345	409.9004353
5	1042	91	23.7492896	483.6598095	0.8494781	508.2585773
6	1042	114	23.7492896	374.4442737	6.7898922	404.9834556

Find the closest facility

In this practical, we are interested in only the destination point (hospital) with the shortest distance. We will use a SQL query to pick the destination with the least total cost among all destinations.

Go to Database --> DB Manager. In the DB Manager dialog, select the Virtual Layers –> Project layers --> Output OD Matrix from the left-hand panel.





Output OD Matrix

General info

Relation type:	Table
Rows:	560

Virtual Layers

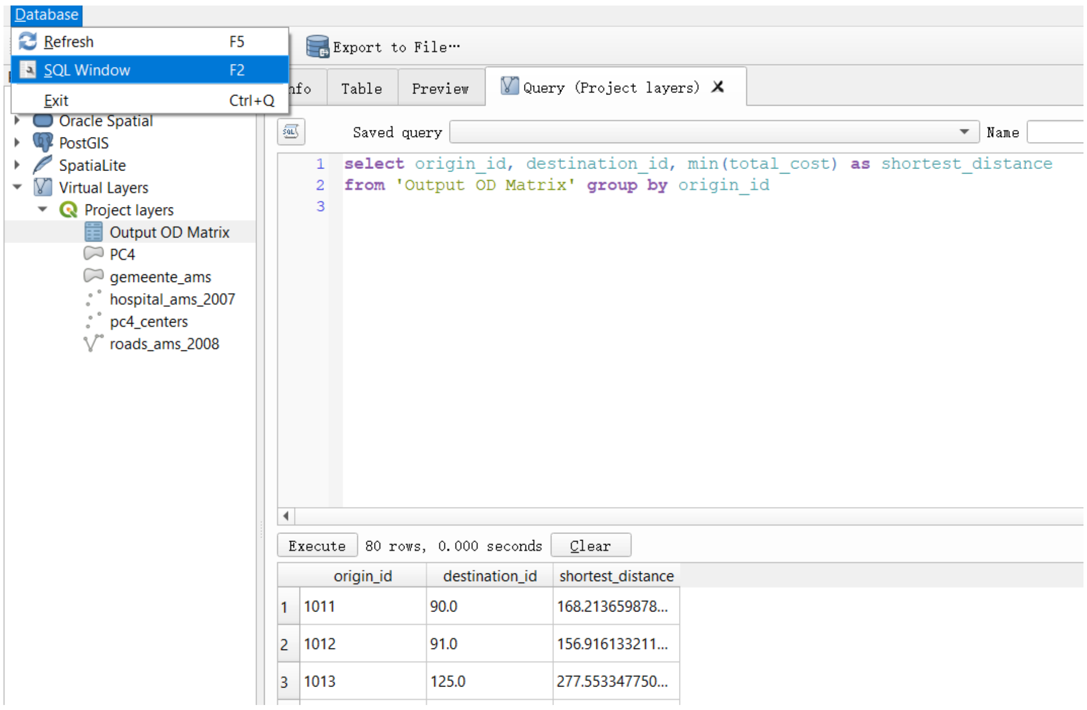
⚠ This is not a spatial table.

Fields

#	Name	Type	Null	Default
0	origin_id	string	Y	
1	destination_id	string	Y	
2	entry_cost	double	Y	
3	network_cost	double	Y	
4	exit_cost	double	Y	
5	total_cost	double	Y	
6	geometry	geometry	Y	

Click the SQL Window button. Enter the following query and click Execute. The results will be displayed as below.

```
select origin_id, destination_id, min(total_cost) as shortest_distance

from 'Output OD Matrix' group by origin_id
```

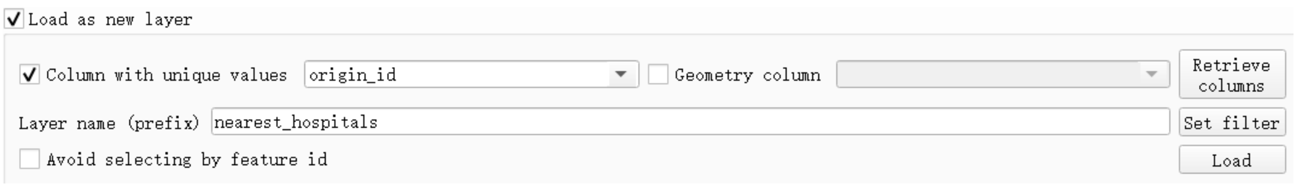


```
select origin_id, destination_id, min(total_cost) as shortest_distance
from 'Output OD Matrix' group by origin_id
```

80 rows, 0.000 seconds

	origin_id	destination_id	shortest_distance
1	1011	90.0	168.213659878...
2	1012	91.0	156.916133211...
3	1013	125.0	277.553347750...

Check and select Column with unique values as **origin_id**. Enter **nearest_hospitals** as the Layer name (prefix). Click **Load**. You will see a new virtual layer **nearest_ hospitals** added to the Layers panel. This table contains the result of our analysis, i.e., nearest hospital for each PC4 zone.



✓ Load as new layer

✓ Column with unique values: origin_id

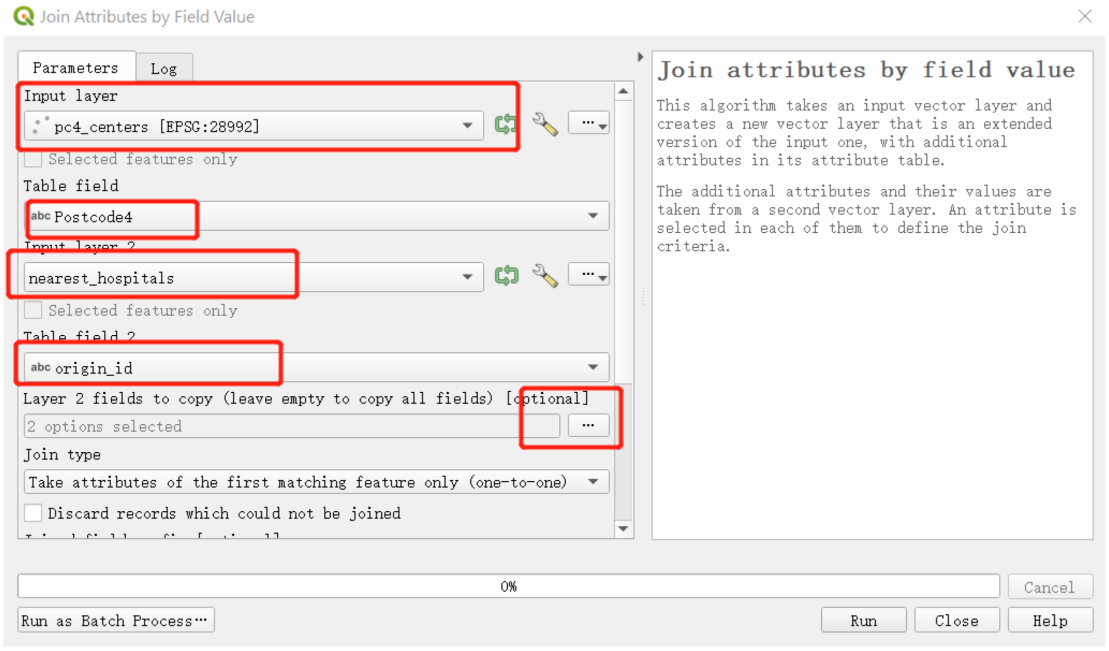
☐ Geometry column

Layer name (prefix): nearest_hospitals

☐ Avoid selecting by feature id

Generate hub lines to visualize and validate the results

To visualize the results, we can now create a hub-spoke visualization using the **nearest_ hospitals** layer. Go to Processing **-->**Toolbox. Search for and locate the **Join attributes by field value** algorithm. Double-click to launch it. Select pc4_centers as the Input layer and "**Postcode4**" as the Table field. Set **nearest_hospitals** as the Input layer 2 and **origin_id** as the Table field 2. Click the ... button next to Layer 2 fields to copy and select **destination_id** and **shortest_distance**. Click **Run**.


Join Attributes by Field Value

Parameters | Log

Input layer: pc4_centers [EPSG:28992]
Selected features only
Table field: Postcode4
Input layer 2: nearest_hospitals
Selected features only
Table field 2: origin_id
Layer 2 fields to copy (leave empty to copy all fields) [optional]: 2 options selected
Join type: Take attributes of the first matching feature only (one-to-one)
Discard records which could not be joined

Join attributes by field value

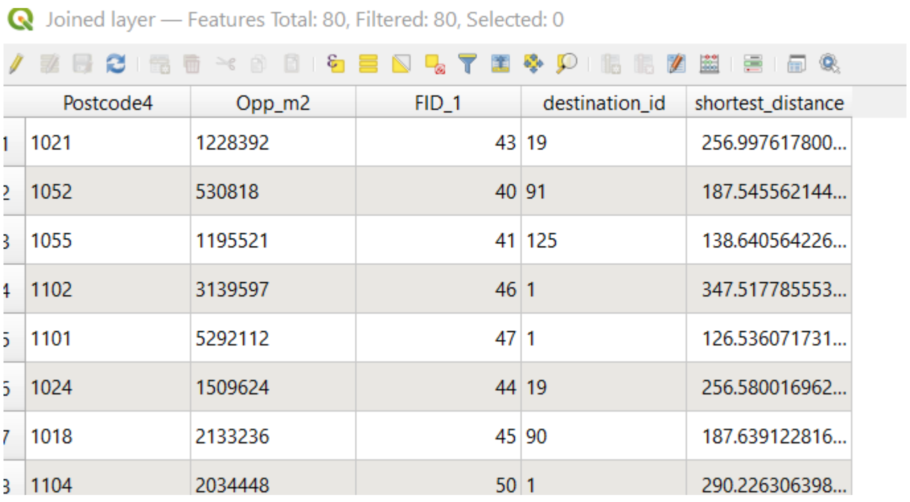
This algorithm takes an input vector layer and creates a new vector layer that is an extended version of the input one, with additional attributes in its attribute table.

The additional attributes and their values are taken from a second vector layer. An attribute is selected in each of them to define the join criteria.

0%

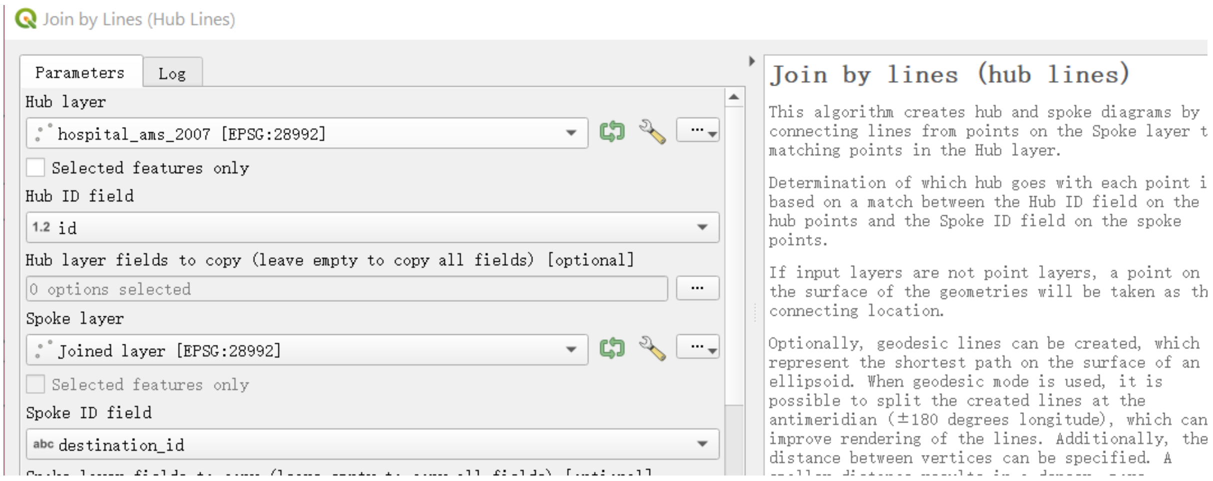
Cancel | Run as Batch Process… | Run | Close | Help

After the running is finished, you can see a new **Joined layer** added to the Layers panel. This layer contains the nearest hospital attributes for each origin point (pc4 center).


Joined layer — Features Total: 80, Filtered: 80, Selected: 0

	Postcode4	Opp_m2	FID_1	destination_id	shortest_distance
1	1021	1228392	43	19	256.997617800…
2	1052	530818	40	91	187.545562144…
3	1055	1195521	41	125	138.640564226…
4	1102	3139597	46	1	347.517785553…
5	1101	5292112	47	1	126.536071731…
6	1024	1509624	44	19	256.580016962…
7	1018	2133236	45	90	187.639122816…
8	1104	2034448	50	1	290.226306398…

We can now create a hub-spoke visualization using this layer. In the **Processing toolbox**, go to **Vector analysis** , and select **Join by lines (hub lines)** algorithm. Right-click to launch it. Select **hospitals_ams_2007** as the Hub layer and **id** as the Hub ID field. Select **Joined layer** as the Spoke layer and **denstination_id** (pc4) as the Spoke ID field. Click **Run**.



Join by Lines (Hub Lines)

Parameters | Log

Hub layer

hospital_ams_2007 [EPSG:28992]

Selected features only

Hub ID field

1.2 id

Hub layer fields to copy (leave empty to copy all fields) [optional]

0 options selected

Spoke layer

Joined layer [EPSG:28992]

Selected features only

Spoke ID field

abc destination_id

Join by lines (hub lines)

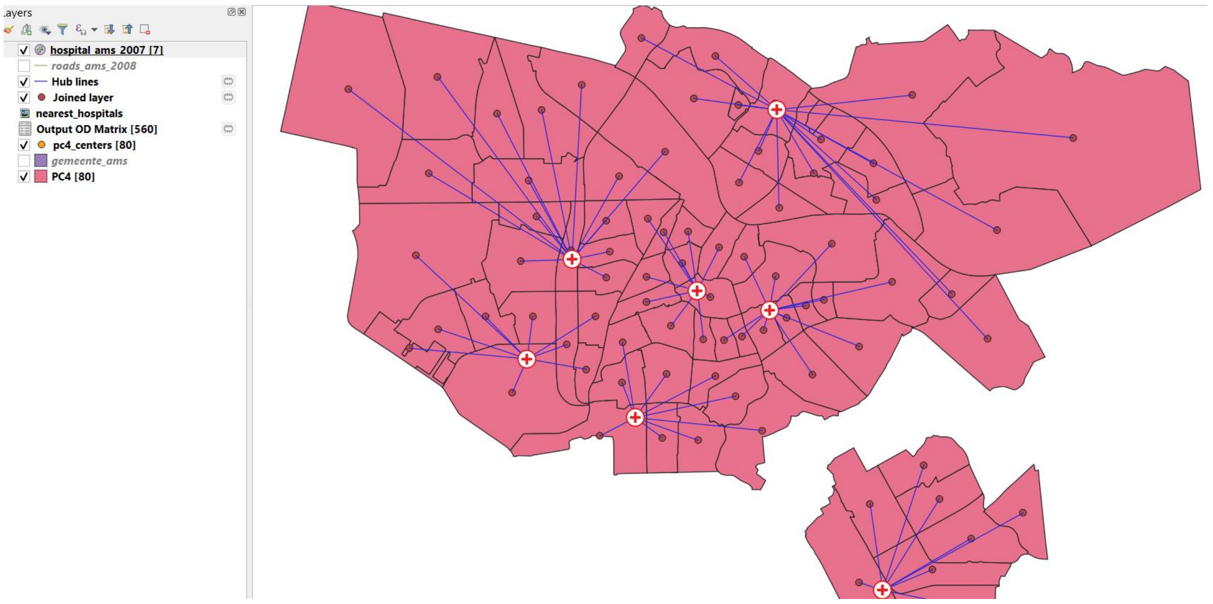
This algorithm creates hub and spoke diagrams by connecting lines from points on the Spoke layer t matching points in the Hub layer.

Determination of which hub goes with each point i based on a match between the Hub ID field on the hub points and the Spoke ID field on the spoke points.

If input layers are not point layers, a point on the surface of the geometries will be taken as th connecting location.

Optionally, geodesic lines can be created, which represent the shortest path on the surface of an ellipsoid. When geodesic mode is used, it is possible to split the created lines at the antimeridian (±180 degrees longitude), which can improve rendering of the lines. Additionally, the distance between vertices can be specified. A

Once the processing is finished, a new layer **Hub lines** will be included to the Layers panel. This layer shows the lines connecting each postcode zone with the nearest hospital.



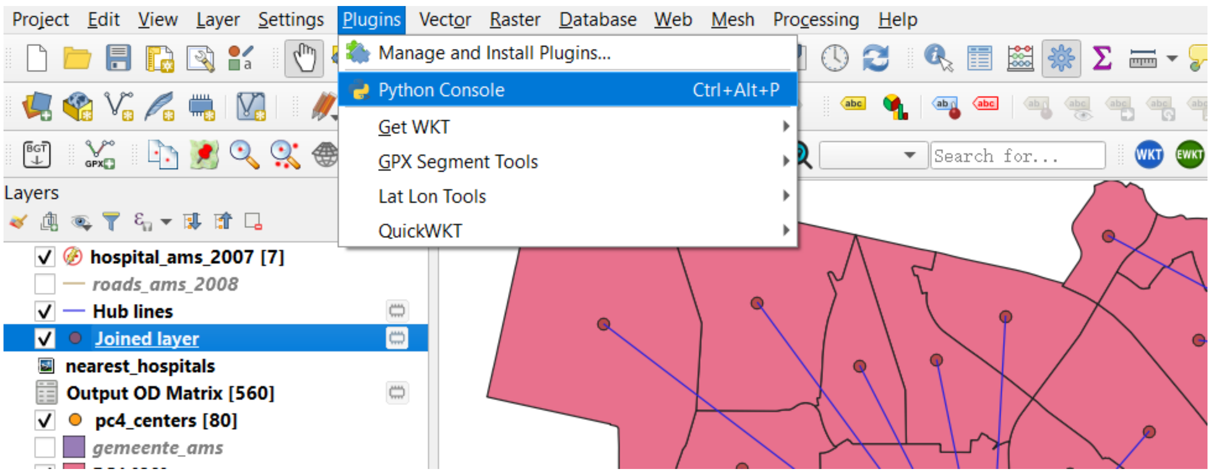
.ayers

- ✓ hospital_ams_2007 [7]
- roads_ams_2008
- ✓ Hub lines
- ✓ Joined layer
- nearest_hospitals
- Output OD Matrix [560]
- ✓ pc4_centers [80]
- gemeente_ams
- ✓ PC4 [80]

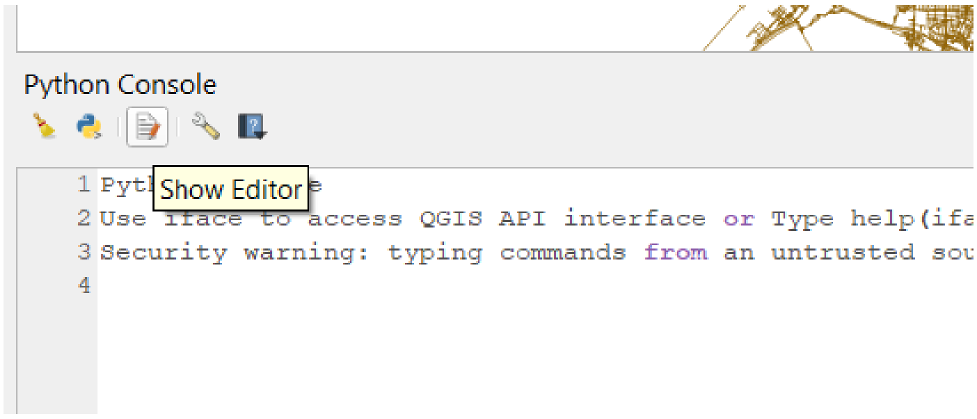
Generate the shortest route to the closest facility

As we can see, the lines connecting the origin and destination are straight-lines, the nearest facility was found using the distance along the road network. It may be more useful to show the actual shortest-path between each origin-destination. To generate the shortest-path between multiple origin-destination pairs, we will use some python scripting to generate the shortest route and visualize it.

Go to Plugins --> Python Console



Click the Show Editor button in the Python Console.



```
1 Pyt
2 Use iface to access QGIS API interface or Type help(ifa
3 Security warning: typing commands from an untrusted sou
4
```

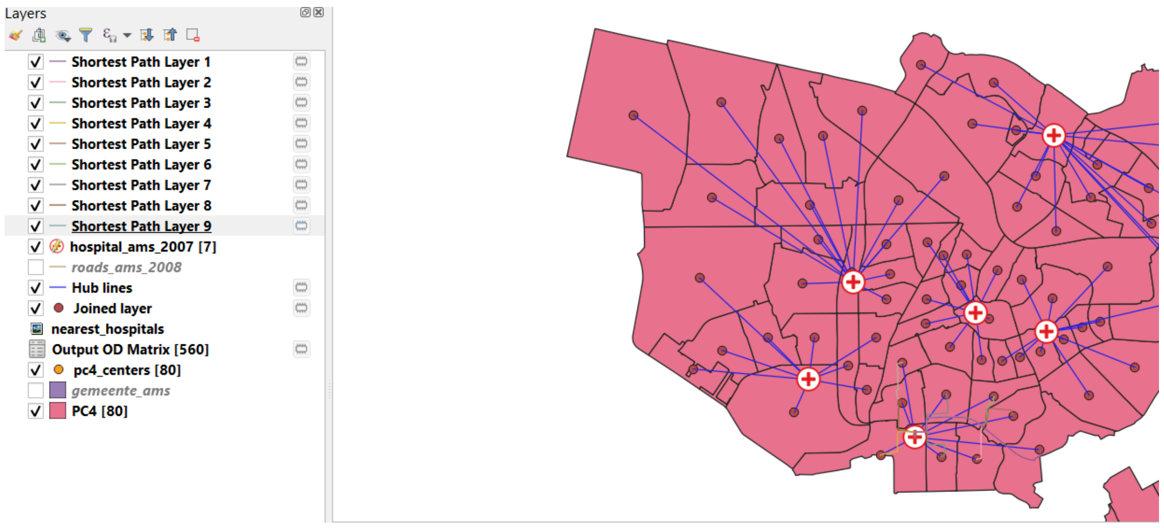
In the editor window, copy/paste the following script (you can also find it in file *py_hospitals.py* in the downloaded zip file). This script contains the parameter values used by the QNEAT3 one-to-one path planning. Look at the script and try to understand it. Click the Run Script button to start execution. For simplicity reasons, we only generate the shortest routes to the hospital with **id =156 (you can change the id if you like). Note that you have to change the corresponding road network entry to your local path (indicated in Red).**

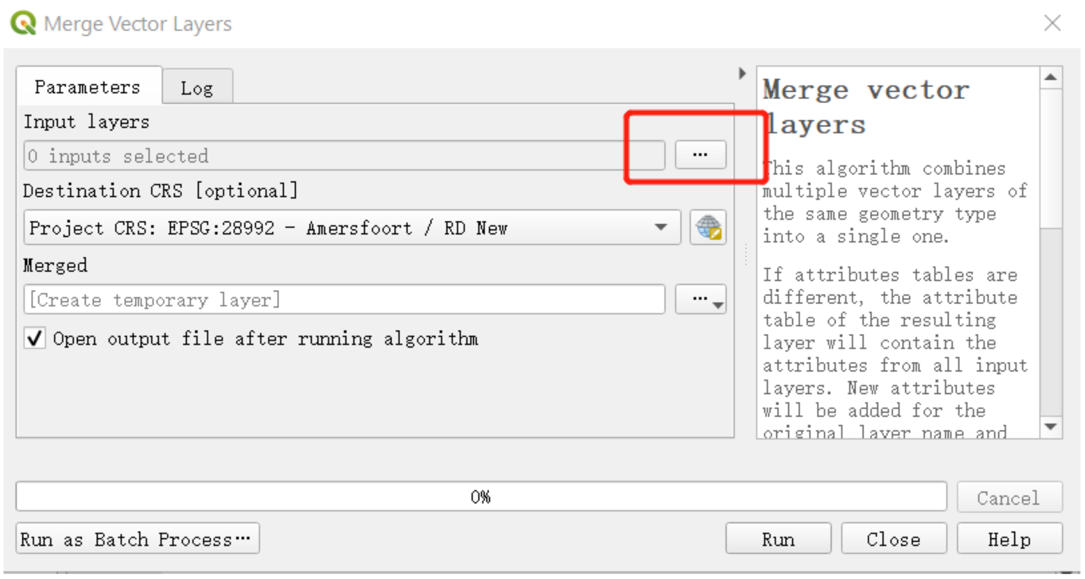
```
origin_layer = QgsProject.instance().mapLayersByName('pc4_centers')[0]

destination_layer = QgsProject.instance().mapLayersByName('hospital_ams_2007')[0]

matrix = QgsProject.instance().mapLayersByName('nearest_hospitals')[0]
```

```
destination_expr = QgsExpression('destination_id=156')


for f in matrix.getFeatures(QgsFeatureRequest(destination_expr)):
    origin_expr = QgsExpression('Postcode4={}'.format(f['origin_id']))
    destination_expr = QgsExpression('id={}'.format(f['destination_id']))
    origin_feature = origin_layer.getFeatures(QgsFeatureRequest(origin_expr))
    origin_coords =  [(f.geometry().asPoint().x(), f.geometry().asPoint().y())
        for f in origin_feature]
    print(origin_coords)
    destination_feature = destination_layer.getFeatures(QgsFeatureRequest(destination_expr))
    destination_coords =  [(f.geometry().asPoint().x(), f.geometry().asPoint().y())
        for f in destination_feature]
    params = {
        'INPUT':'F:\\KuPan\\Teaching\\ADS\\Lab2\\AMS\\roads_ams_2008.shp',
        'START_POINT':'{},{}'.format(origin_coords[0][0], origin_coords[0][1]),
        'END_POINT':'{},{}'.format(destination_coords[0][0], destination_coords[0][1]),
        'STRATEGY':1,
        'ENTRY_COST_CALCULATION_METHOD':1,
        'DIRECTION_FIELD':'DIRECTIONA',
        'VALUE_FORWARD':'One Way (Digitizing direction)\n',
        'VALUE_BACKWARD':'One way (Against digitizing direction)\n',
        'VALUE_BOTH':'',
        'DEFAULT_DIRECTION':2,
        'SPEED_FIELD':'kph',
        'DEFAULT_SPEED':5,
        'TOLERANCE':0,
        'OUTPUT':'memory:'}
    print('Executing analysis')
    processing.runAndLoadResults("qneat3:shortestpathpointtopoint", params)
```

The script may take a few minutes to run. Once it is done, you will see 9 new layers named Shortest Path layer. Let's rename these layers, assign each layer a number from 1-9, merge them to a single layer. Go to the **Vector general**, select **Merge vector layers** and click it. Select the 9 shortest path layers and **click Run.**

The figure below shows the 9 shortest Path layers before merging into a single layer.

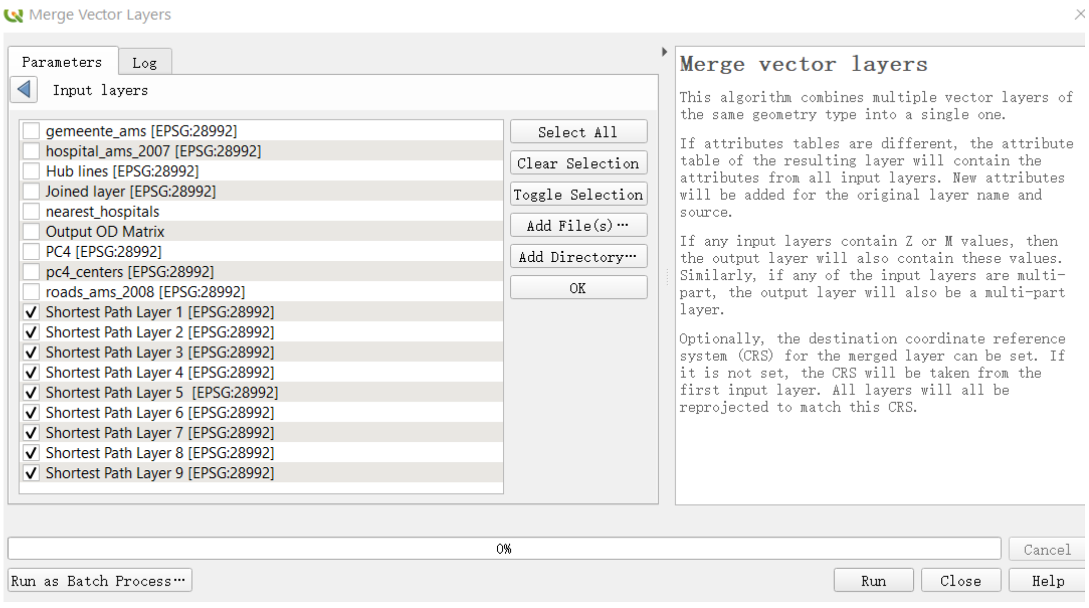




Merge vector layers

This algorithm combines multiple vector layers of the same geometry type into a single one.

If attributes tables are different, the attribute table of the resulting layer will contain the attributes from all input layers. New attributes will be added for the original layer name and


Merge Vector Layers
Parameters
Log
Input layers
gemeente_ams [EPSG:28992]
hospital_ams_2007 [EPSG:28992]
Hub lines [EPSG:28992]
Joined layer [EPSG:28992]
nearest_hospitals
Output OD Matrix
PC4 [EPSG:28992]
pc4_centers [EPSG:28992]
roads_ams_2008 [EPSG:28992]
✓ Shortest Path Layer 1 [EPSG:28992]
✓ Shortest Path Layer 2 [EPSG:28992]
✓ Shortest Path Layer 3 [EPSG:28992]
✓ Shortest Path Layer 4 [EPSG:28992]
✓ Shortest Path Layer 5 [EPSG:28992]
✓ Shortest Path Layer 6 [EPSG:28992]
✓ Shortest Path Layer 7 [EPSG:28992]
✓ Shortest Path Layer 8 [EPSG:28992]
✓ Shortest Path Layer 9 [EPSG:28992]
Select All
Clear Selection
Toggle Selection
Add File(s)…
Add Directory…
OK
0%
Cancel
Run as Batch Process…
Run
Close
Help

Merge vector layers

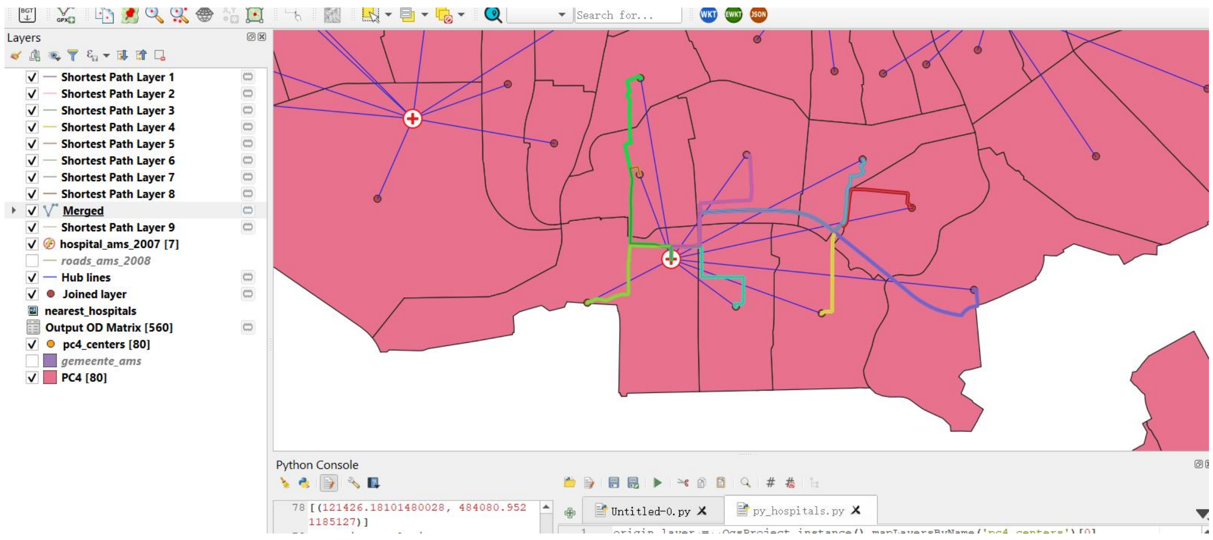
This algorithm combines multiple vector layers of the same geometry type into a single one.

If attributes tables are different, the attribute table of the resulting layer will contain the attributes from all input layers. New attributes will be added for the original layer name and source.

If any input layers contain Z or M values, then the output layer will also contain these values. Similarly, if any of the input layers are multi-part, the output layer will also be a multi-part layer.

Optionally, the destination coordinate reference system (CRS) for the merged layer can be set. If it is not set, the CRS will be taken from the first input layer. All layers will all be reprojected to match this CRS.

A new Merged layer will be created and contains the shortest path between the selected hospital (id = 156) and destinations (PC4 zones).



```
78 [(121426.18101480028, 484080.952
   1185127)]
```

References:

QGIS Network Analysis Toolbox 3, https://root676.github.io/

Catchment area, https://en.wikipedia.org/wiki/Catchment_area#:~:text=In%20human%20geography%2C%20a%20catchment,to%20attend%20a%20local%20school.

OD cost matrix analysis https://desktop.arcgis.com/en/arcmap/latest/extensions/network-analyst/od-cost-matrix.htm

Assignment: Catchment analysis for police stations in Amsterdam

In this exercise, you do a similar analysis to find catchment areas (PC4 areas) for the police stations in the city of Amsterdam. You may reuse certain datasets (PC4 and network datasets) and adapt your codes. For your convenience, you can download the dataset of police stations (Politiebureaus) of the Netherlands via the following link https://shorturl.at/tvQ48. Note that for police stations, police travels towards the incident places. So in the calculation of the OD Matrix, you have to select **pc4_centers** as the **To-Points layer** and set **police stations** as the **From-Points**. In your submitted document, please include the following: 1). A screenshot of the used datasets (only Amsterdam); 2). A screenshot of the OD matrix table. Also please explain what the **origin_id** is and what **destination_id** is in this exercise; 4) A screenshot of hub lines from police stations to PC4s; 4). Screenshots of the shortest routes from a selected police station to PC4s that would use its services.