TMTplus Introduction to Scientific Programming

Mahdi Farnaghi, Mahdi Khodadadzadeh & Robert Ohuru

March 2021

Note from the teaching staff

The materials that we use for this course are somewhat new and have been heavily reworked. We will likely have made mistakes in this work or have glossed over issues that deserved better or different treatment. Where you find issues worth noting please do report these to us as it will allow to repair and improve.

By the way, welcome to the Exercise book for Introduction to Scientific Programming! We believe this book is self-explanatory, so go ahead and practice.

Chapter 11

Gdal vector operations

In many geospatial projects, vector data is common and their operational functionality is fundamental to achieve success.

The exercises

The goal of this practical session is to learn using the *ogr* package and handle vector data.

For the next exercises, we will work in part with vector data on the island of Texel (pronounce 'Tessel'), the largest inhabited island of The Netherlands. You will need data which is in a zipped folder containing the following files:

Texel_Overview.qgs A QGIS map that shows the point and polygon features that you will use in this exercises. You will see point and polygon features:

Texel Points The pink dots show the locations of people visiting the island.

Texel Polygons
- The green polygon shows a patch of forest.
- The grey polygon shows the main village in the island.
- The yellow polygon shows a recreation space

Texel_Points.shp This contains the locations of the visitors to the island. It holds 179 point features.

Texel_Polygons.shp This file contains the three polygons described above.

NL_provinces.shp This file contains the provinces of The Netherlands.

11.1 Access to vector data and simple spatial analysis

In this block of exercises you follow the steps taken in the theory slides. You will open a vector data source, and extract its layer and features. You will also perform simple spatial analysis, and in the end save the results to disk.

Ex 11.1

Using the *ogr* library, open file `NL_provinces.shp` and print the following information:

- Driver name
- Data source metadata
- Number of layers

Ex 11.2

With file `NL_provinces.shp` access the first layer data source using `.GetLayer()` and print the following information:

- The number of attribute table fields
- Iterate over the attribute table fields, and print all their names
- Layer extents
- Number of features
- Spatial reference system

Ex 11.3

Now that we have accessed the data source and the layer, it is time to access features inside the first layer. Iterate over the features, and print their value for the field `NAME_1`. You can access such value with the command `.GetFieldAsString()` method inside the loop.

Ex 11.4

With a filter on the layer, using the method `.SetAttributeFilter()`, extract the feature for the province of Overijssel (mark correct spelling) and print the following for this feature:

- NAME_1 value
- Type of geometry
- Geometry in well-known text format
- Area
- Extent

Ex 11.5 Access the feature geometry of Overijssel province and then create a buffer of 5,000 meters for this province. Print its geometry in WKT and also as a JSON object.

Ex 11.6 Save the calculated buffer as a new shapefile, and name it properly. Next, open it in either QGis or ArcGis. Perform a visual inspection.

Ex 11.7 Extract the feature geometry of Drenthe province, and with the previous buffer from the Overijssel geometry, check whether this it intersects Drenthe. If they do, calculate the intersection geometry, and save it too as a new shapefile. Open the shapefile in QGis or ArcGis. Perform a visual inspection again.

11.2 Calculation of distances between vector features

In the next exercises, you will learn to perform a basic operation in the field of GIS: the distance between a point and a polygon. For this purpose, you will use the vector feature located in the island of Texel, the westernmost of the Wadden Islands along the northern coast of The Netherlands.

Ex 11.8 To work with geospatial layers in Python, one first needs to load them into memory. Here, you will load two vector layers and then perform a distance operation.

```
open_polygons = ogr.Open(path_polygons)
target_polygons = open_polygons.GetLayer (0)
open_points = ogr.Open(path_points)
target_points = open_points.GetLayer (0)
```

The above script does the following:

1. Open the polygon data source
2. Load the polygon layer. From here on, we have a Python iterable
3. Open the points data source
4. Load the points layer

There are several handy layer methods to apply to opened vector layers, which may help to obtain information about the layer you are handling:

`.GetFeatureCount()` which returns the number of vector features in a layer.

`.GetSpatialRef()` which returns a text description of the reference system used to create a layer.

Write a Python script that opens two datasets (`Texel_Polygons.shp` and `Texel_Points.shp`) which we will use below, and print their respective number of features and spatial reference identifiers. Make sure you get the same number of features in the Python console as you saw in your GIS.

Ex 11.9

Once a layer is loaded, you can fetch features from it using method

`.GetFeature(X)`,

where `X` is an integer sequence number (starts counting at 0). You can obtain a geometry with the feature method `.GetGeometryRef()`. Fetch the 25th point feature and the 2nd polygon feature from the vector layers and then print their geometry.

To calculate distance, you can apply the geometry method `.Distance()` to your point feature, passing as a parameter the polygon.

```
polyGeometry = poly.GetGeometryRef ()
pointGeometry = point.GetGeometryRef ()
pointGeometry.Distance(polyGeometry)
```

This will calculate the distance between the point and polygon features. Distance comes in the same unit as associated with the used reference system, in our case, EPSG:28992 (RD_New). It is meters. Print the calculated distance.

Ex 11.10

In Exercise 8, we mentioned that once a layer is loaded with

`datasource.GetLayer()`

it becomes an iterable. This means that you can use a for loop over the layer. Applying the method `.GetField()` to each feature in the polygon dataset will

provide the contents of the given field for each feature. Now, use a for loop to iterate over the polygon layer and print field 'Hoofdgroep'. (This means: Main group.) You should see three records: 'Bos' (forest), 'Recreatie' (recreation) and 'Bebouwd' (built-up).

Ex 11.11

Now, using the 25th point used in Exercise 9, calculate distances between this point and each land use feature on the island of Texel. To do so, you will need to use `.GetGeometryRef()` at each iteration step and apply a distance operation between the 25th point and the feature. The yielded distances should be close to 4 km, 16 km and 5 km.