# Tombolo Digital Connector Examples

*Exported from the Tombolo GitHub Wiki on 2017-12-04 15:45*

# Cycling and Air Quality

As a demonstrative example of how Tombolo Digital Connector works we present a story of a fictional user called Thomas, who works for a London based active transport lobby. Thomas wants to visualise the relation between bicycle friendly boroughs in London and air quality. He browses the Tombolo Digital Connector catalogue of importable data and finds traffic counts from Department for Transport and air quality measurements from the London Air Quality Network. In order to combine the two datasets, he writes a Tombolo data export recipe (also known as model recipe) that returns a GeoJson file with the following output:

- the geographic shape of each London borough,
- the name of the borough,
- the cycle traffic in the borough as a fraction of car traffic,
- and the average nitrogen dioxide concentration as a proxy for air quality.

After exporting the data, Thomas opens the file in QGIS and even if he is not a GIS expert he can create simple heat-maps if he is given the right GeoJson data.

## Step 1: Create model recipe

A Tombolo model recipe is a json file describing the dataset to be exported and the output format. An executable recipe can be found here, but below we we describe the building blocks.

```
{
  "dataset" : INSERT-YOUR-DATASET-RECIPE,
  "exporter" : "uk.org.tombolo.exporter.GeoJsonExporter"
}
```

```
We will describe the dataset recipe below, but in the example above we can see that Thomas
```

The dataset recipe has three parts

```
{
  "subjects" : [INSERT-SUBJECT-RECIPE],
  "datasources" : [INSERT-DATASOURCE-RECIPE],
  "fields" : [INSERT-FIELD-RECIPE]
}
```

The **subjects** are the entities for which you want to combine data. In Thomas' case these are London Boroughs. The **datasources** specify which datasets should be used. In Thomas' case this will be Department

of Transport traffic counts and London Air Quality Network. Finally, the **fields** are descriptions of how the data should be combined together in the output file. In Thomas' case it could be one field for the cycling count as a proportion of car traffic counts and average yearly $NO_2$ levels across each borough.

For more information on each of these components see the description of the local datastore.

## Subjects

Thomas will specify that he wants London Boroughs as follows:

```
{
  "subjectType" : "localAuthority",
  "provider":"uk.gov.ons",
  "matchRule": {
    "attribute": "label",
    "pattern": "E090%"
  }
}
```

Literally this means that the Tombolo Digital Connector will output each **local authority**, as provided by Office for National Statistics (**ONS**), that has a **label** starting with the string **"E090"**. It happens that the London Boroughs can be uniquely identified by that string prefix. For more information about how to specify subjects see the subject section in the recipe language.

## Datasources

Thomas will now specify the datasources as follows:

```
[
  {
    "importerClass" : "uk.org.tombolo.importer.ons.OaImporter",
    "datasourceId": "localAuthority"
  },
  {
    "importerClass" : "uk.org.tombolo.importer.dft.TrafficCountImporter",
    "datasourceId" : "trafficCounts",
    "geographyScope" : ["London"]
  },
  {
    "importerClass" : "uk.org.tombolo.importer.lac.LAQNImporter",
    "datasourceId": "airQualityControl"
  }
]
```

The first datasource refers to **local authorities** from the **Output Area Importer** from **ONS**. The second datasource refers to **traffic counts** from the Department for Transport (**DfT**). Since the Department for Transport provides data-files for each region separately, Thomas can specify that he is only interested in traffic counts within **London**. The third datasource refers to **air quality** data from the London Air Quality Network (**LAQN**) from King's College London.

For more information about specifying data-sources see the data-source section of the recipe language.

At the moment we are relying on users knowing a lot about the data they want to import. This will be resolved in late 2017 with a user interface for supporting recipe generation.

## Fields

To finish his model recipe Thomas needs to specify the two data fields he wants to associate to each borough. Since both Department for Transport and London Air Quality Network sensor report readings for points (but not boroughs), Thomas needs to aggregate the sensor values to the borough level. Since he has already specified that he wants London Boroughs as outputs, the aggregation can be done automatically by the built-in **Geographic Aggregation Field**.

That is, the first field outputs the average nitrogen dioxide level for each borough, aggregated over all air quality sensors in the borough, and is specified as follows:

```
{
  "fieldClass": "uk.org.tombolo.field.aggregation.GeographicAggregationField",
  "label": "NitrogenDioxide",
  "subject": {
    "provider": "erg.kcl.ac.uk",
    "subjectType": "airQualityControl"
  },
  "function": "mean",
  "field": {
    "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
    "attribute": {
      "provider" : "erg.kcl.ac.uk",
      "label" : "NO2 40 ug/m3 as an annual me"
    }
  }
}
```

I.e. the Geographic Aggregation Field looks up all LAQN air-quality-sensors within each borough, extracts the latest value for each sensor and computes the mean over all values.

The latter field is slightly more complicated and outputs the bicycle count from all traffic counters across each borough and divides it by the total traffic count in the borough. The division is performed using the built in **Arithmetic Field**, applied on the output of two **Geographic Aggregation Fields** where traffic counts have been aggregated using a sum. The first field aggregates bicycle (pedal cycle) counts and the latter aggregates car traffic.

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField",
  "label": "BicycleFraction",
  "operation": "div",
  "field1": {
    "fieldClass": "uk.org.tombolo.field.aggregation.GeographicAggregationField",
    "label": "BicycleCount",
    "subject": {
      "provider": "uk.gov.dft",
      "subjectType": "trafficCounter"
    },
    "function": "sum",
    "field": {
      "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
```

```
      "attribute": {
        "provider" : "uk.gov.dft",
        "label" : "CountPedalCycles"
      }
    }
  },
  "field2": {
    "fieldClass": "uk.org.tombolo.field.aggregation.GeographicAggregationField",
    "label": "CarCount",
    "subject": {
      "provider": "uk.gov.dft",
      "subjectType": "trafficCounter"
    },
    "function": "sum",
    "field": {
      "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
      "attribute": {
        "provider": "uk.gov.dft",
        "label": "CountCarsTaxis"
      }
    }
  }
}
```

## Step 2: Exporting data

Now that Thomas has put his model recipe together, the next step in the process is to run the Tombolo Digital Connector on the recipe. We use the Gradle build tool to do that from the command line.

```
gradle runExport \
    -PdataExportRecipe='path/to/recipe/file.json' \
    -PoutputFile='path/to/output/file.json'
```

The command takes two parameters, one pointing to the model recipe that was built in the previous step and one pointing to the output file to be generated.

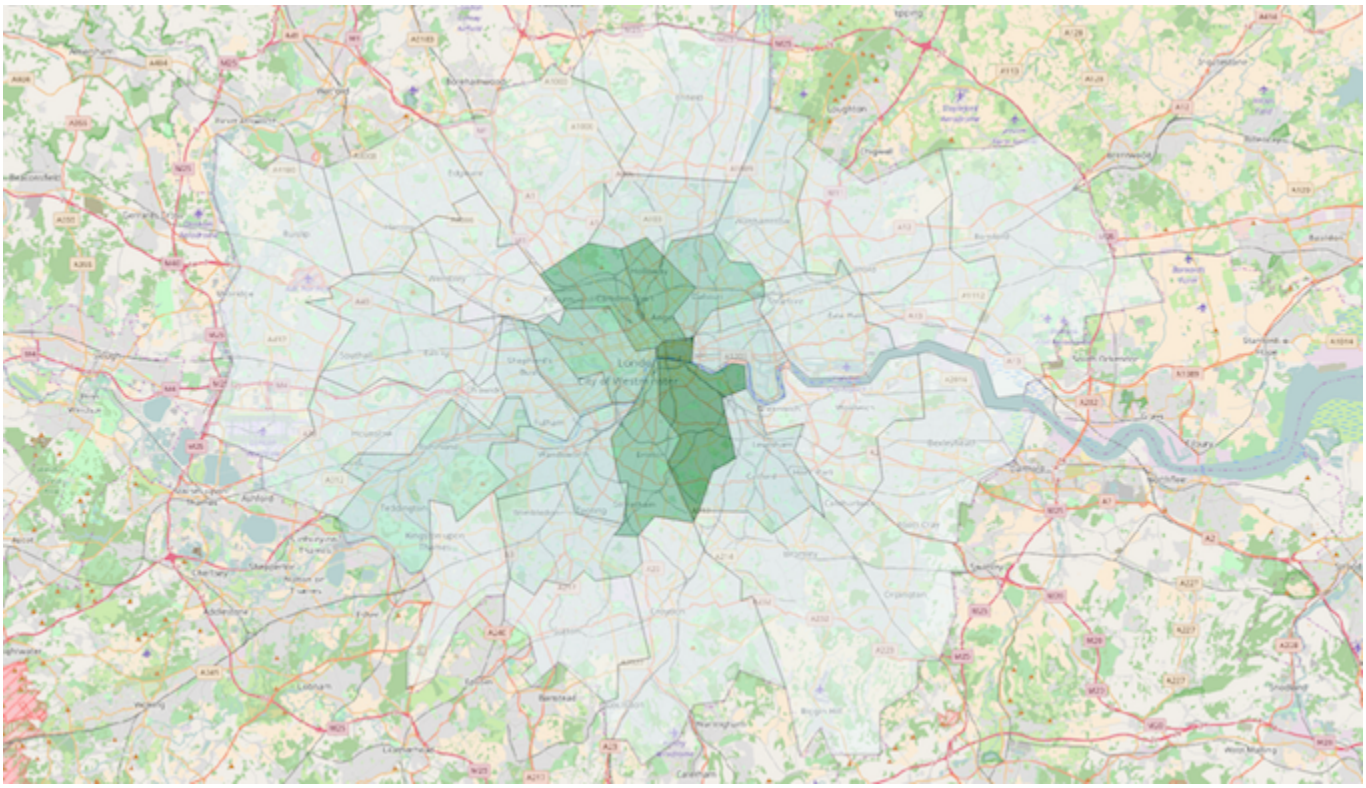
This will generate a GeoJson file with one geographic shape (know as feature in GeoJson lingo) for each London Borough.

```
{
  "type":"FeatureCollection",
  "features":[
    {
      "type":"Feature",
      "geometry":{
        "type":"Polygon",
        "coordinates":[[[-0.0802,51.5069],[-0.1092,51.5099],[-0.1114,51.5098],
                        [-0.1116,51.5153],[-0.1053,51.5185],[-0.0852,51.5203],
                        [-0.0784,51.5215],[-0.0802,51.5069]]]
      },
      "properties":{
        "label":"E09000001",
        "name":"City of London",
        "Nitrogen Dioxide":81.3333333333333,
        "Bicycle Fraction":0.25473695591455
```

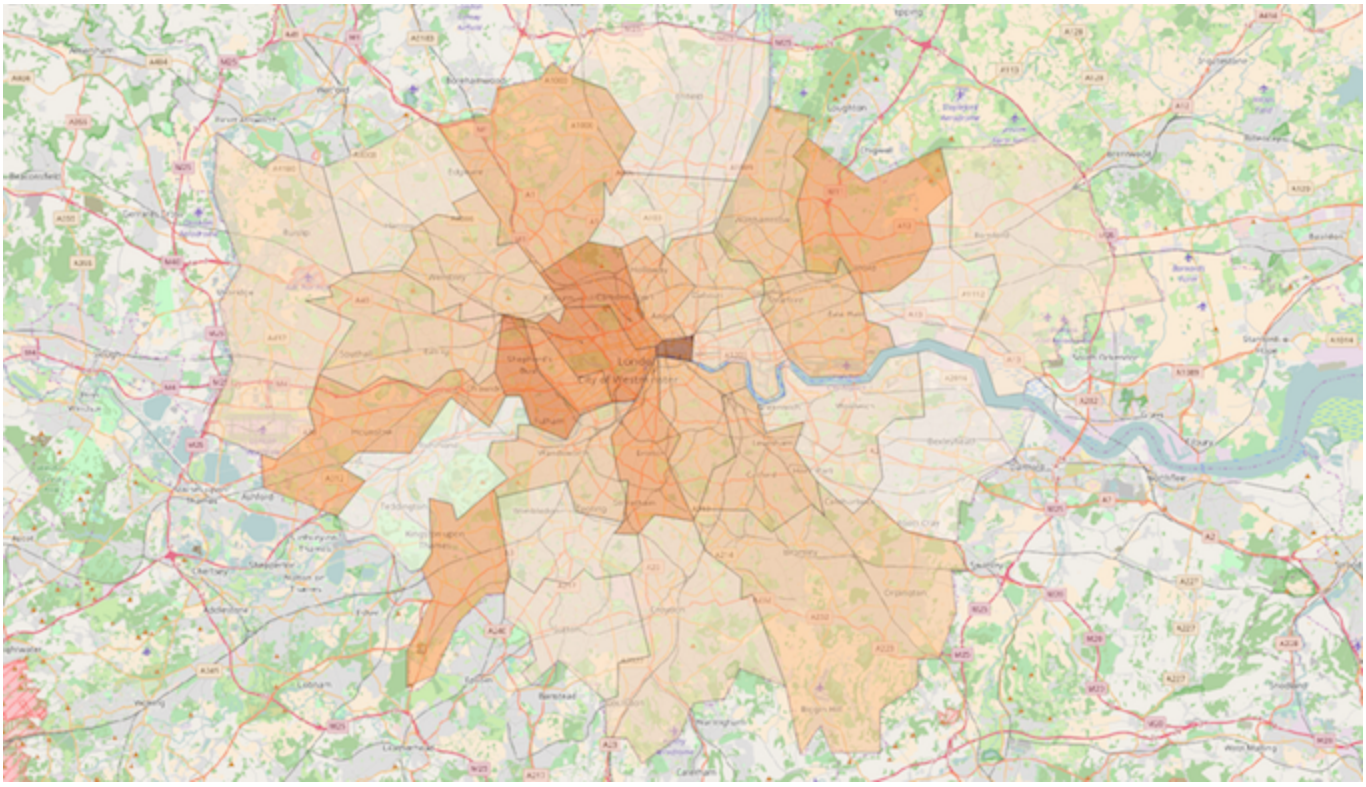
```
      }
    },
    {
      "type":"Feature",
      "geometry":{
        "type":"Polygon",
        "coordinates":[[[0.1468,51.5688],[0.1619,51.5616],[0.1852,51.5655],
                        [0.1902,51.5527],[0.1605,51.5123],[0.1272,51.5194],
                        [0.0981,51.515],[0.0925,51.5257],[0.0727,51.5293],
                        [0.0684,51.5444],[0.0935,51.5458],[0.119,51.5573],
                        [0.1316,51.5718],[0.1264,51.5867],[0.1482,51.599],
                        [0.1468,51.5688]]]
      },
      "properties":{
        "label":"E09000002",
        "name":"Barking and Dagenham",
        "Nitrogen Dioxide":34,
        "Bicycle Fraction":0.00615611326607704
      }
    },
    ...
  ]
}
```


## Step 3: Data visualisation

Having exported the data, Thomas can open the output file in QGIS and export simple maps of on one had the fraction of bicycle traffic and nitrogen dioxide levels on the other.



*Heat map of bicycle traffic counts as a fraction of car traffic counts.*



*Heat map of annual nitrogen dioxide levels.*