TomboloDigitalConnector Documentation

Release 1.

Future Cities Catapult

Nov 30, 2017

Contents

The Tombolo Digital Connector is an open source tool that allows data enthusiasts to efficiently connect different data sets into a common format. It enables the **transparent** and **reproducible** combination of data which exists in different domains, different formats and on different spatio-temporal scales. The Tombolo Digital Connector makes it easier to generate models, indexes and insights that rely on the combination of data from different sources.

There are three particularly important parts to the Tombolo Digital Connector:

- Importers
  - Built-in importers harvest a range of data sources into the centralised data format. Examples include data from ONS, OpenStreetMap, NOMIS, the London Air Quality Network and the London Data Store. **We welcome the creation of additional importers**.
- Centralised data format
  - All data imported into the Tombolo Digital Connector adopts the centralised data format. This makes it easier to combine and modify data from different sources.
- Recipes
  - Users generate recipes with a declarative 'recipe language' to combine the data in different ways. This combination can generate new models, indexes and insights. For example, existing recipes can generate models of social isolation, calculate the proportion of an area covered by greenspace and even generate an active transport index. **We welcome the creation of additional recipes**.

CHAPTER 1

Tombolo Digital Connector

The Tombolo Digital Connector is a software library for integrating urban models and datasets.

For further information see the wiki.

1.1 Table of Contents:

1.2 Quick start

1.2.1 Requirements

- JDK (1.8+)
- PostgreSQL (9.4+)
- PostGIS (2.1+)
- Gradle (2.12+)
- (Optional) Wercker (1.0+)

1.2.2 Configure the project

Copy and amend the example configuration file at `/gradle.properties.example` to `/gradle.properties`.

Copy and amend the example API keys file at `/apikeys.properties.example` to `/apikeys.properties`. If you're not using the services mentioned in the file you can leave it as-is.

1.2.3 Set up main database

Then run the following to set up your database:

```
# Create a user and database
createuser tombolo
createdb -O tombolo tombolo -E UTF8
psql -d tombolo -c "CREATE EXTENSION postgis;"
psql -d tombolo -c "SET NAMES 'UTF8';"


# Create DB tables and load initial fixtures
psql -d tombolo -U tombolo < src/main/resources/sql/create_database.sql
```

1.2.4 Set up test database

The test database is used by the tests and is cleared routinely. We use this to gain control over what is in the database when our tests are running and to avoid affecting any important data in your main database.

To set up the test user and database:

```
# Create a user and database
createuser tombolo_test
createdb -O tombolo_test tombolo_test -E UTF8
psql -d tombolo_test -c "CREATE EXTENSION postgis;"
psql -d tombolo_test -c "SET NAMES 'UTF8';"

# Create DB tables and load initial fixtures
psql -d tombolo_test -U tombolo_test < src/main/resources/sql/create_database.sql
```

1.2.5 Run tests

```
gradle test
```

If you use the IntelliJ JUnit test runner, you will need to add the following to your VM Options in your JUnit configuration (Run -> Edit Configurations -> All under JUnit, and Defaults -> JUnit):

```
-enableassertions
-disableassertions:org.geotools...
-Denvironment=test
-DdatabaseURI=jdbc:postgresql://localhost:5432/tombolo_test
-DdatabaseUsername=tombolo_test
-DdatabasePassword=tombolo_test
```

1.3 Local deploy

To deploy to your local Maven installation (`~/.m2` by default):

```
gradle install
```

1.4 Run Tasks

1.4.1 Run export

We use the Gradle task `runExport` to run exports. The parameters are as follows:

```
gradle runExport \
    -PdataExportSpecFile='path/to/spec/file.json' \
    -PoutputFile='output_file.json' \
    -PforceImports='com.className'
    -PclearDatabaseCache=true
```

For example, this calculates the proportion of cycle traffic received at a traffic counter relative to the total traffic in a given borough and outputs the results to the file `reaggregate-traffic-count-to-la.json`:

```
gradle runExport \
    -PdataExportSpecFile='src/main/resources/executions/examples/reaggregate-traffic-
↪count-to-la.json' \
    -PoutputFile='reaggregate-traffic-count-to-la_output.json'
```

1.4.2 Run data catalogue

We use the Gradle task `runCatalogue` to explore the data catalogue. The parameters are as follows:

```
gradle runCatalogue \
    -PimporterClassName='full.name.of.the.importer'
    -PdatasetId='dataset-id'
```

If the datasetId parameter is not present the program writes out all the datasets available from the importer. If the datasetId is specificed the program writes out all attributes available from that importer dataset pair.

For example, this lists all datasets available in the ONS Census importer:

```
gradle runCatalogue -PimporterClassName='uk.org.tombolo.importer.ons.ONSCensusImporter
↪'
```

For example, this lists all attributes available in the dataset QS102EW from ONS (Population density):

```
gradle runCatalogue -PimporterClassName='uk.org.tombolo.importer.ons.ONSCensusImporter
↪' -PdatasetId='QS102EW'
```

1.4.3 Export data catalogue

We us the Gradle task `exportCatalogue` to export a JSON file detailing the capabilities of the connector.

```
gradle exportCatalogue -PoutputFile=catalogue.json
```

1.5 Continuous Integration

We're using Wercker for CI. Commits and PRs will be run against the CI server automatically. If you don't have access, you can use the Wercker account in the 1Password Servers vault to add yourself.

If you need to run the CI environment locally:

1. Install the Wercker CLI
2. Run `wercker build`

The base image is generated with the very simple Dockerfile in the root of this project. To push a new image to DockerHub you will need access to our DockerHub account. If you don't have access, you can use the DockerHub account in the 1Password Servers vault to add yourself.

If you need new versions of PostgreSQL, Java, etc, you can update the image:

```
docker build -t tombolo .
docker images
# Look for `tombolo` and note the IMAGE ID
docker tag <IMAGE_ID> fcclab/tombolo:latest
docker push fcclab/tombolo
```

1.6 Wiki to PDF

To create a PDF version of the Wiki documentation clone the wiki respository and run the gradel build in the wiki repository root folder.

```
gradle build
```

The Digital Connector main features are:

- Importers for various open source data sources and transforming them into a centralised data format.
- Importers for proprietary data that has been transformed according to the centralised data format.
- Model recipe language for creating urban models by joining datasets.
- Aggregators and de-aggregators for joining spatial data of different spatial granularities.
- Export data and model output in CSV or GeoJson format to use further in tools such as GIS systems, Jupyter Notebooks, etc.

The main advantages of using the Digital Connector are:

- No need to parse and understand the file format of numerous open data sources.
- Simple aggregation and de-aggregation of different spatial granularities.
- Reusability of model recipes.
- Simplified data gathering from complex APIs, such as Open Street Map.

In a nutshell, using the Digital Connector involves four steps:

1. After ensuring that you have installed the pre-requisites outlined here, install the Digital Connector by cloning the GitHub repository.

2. Write a recipe file that describes the desired data output from the Digital Connector. This recipe includes which data-sources to use and how to mix the data together.
3. Run the export. In this step the Digital Connector will connect to the relevant data sources, download the necessary data, join the data as per the user recipe and export the data in the requested output format.
4. Work with the data in tools such as QGIS, Jupyter Notebooks, R, etc.

CHAPTER 2

Example: Correlation between deprivation and childhood obesity

Rosie wants to study the correlation between deprivation and childhood obesity. She browses the importable data and finds that MSOA level childhood obesity is available from Public Health England and LSOA level deprivation score is available from DCLG. She writes in her recipe file that the output should be a CSV file with three columns.

- The first column is the MSOA identifier.
- The second column contains the percentage of obese children in that MSOA.
- The third column contains the median LSOA level deprivation value for the MSOA.

After exporting the data, Rosie loads it to a Jupyter Notebook where she uses R to calculate correlation coefficients and render a scatter-plot.

CHAPTER 3

Example: Cycling and pollution

Thomas wants to visualise the relation between bicycle friendly boroughs in London and air quality. He browses the importable data and finds traffic counts from Department for Transport and air quality measurements from the London Air Quality Network. He writes a recipe saying that he wants the the shape of each borough as a feature. He specifies that each borough should have three attributes:

- a name attribute;
- an attribute showing cycle traffic as a fraction of total traffic;
- and an attribute quantifying the air quality.

After exporting the data, Thomas opens the file in QGIS where he visualises it.

For a detailed use case see Use Case on Cycling and Air Quality.

CHAPTER 4

Example: Active Transport Index

As part of the Tombolo project we have built an application called City Index Explorer. The application functions as a demonstrator of the possibilities for combining various urban data sources into an urban model. One of the indices we have developed for demonstration is the Active Transport Index. The index combines various data sources to assign a score to each LSOA representing the use and potential for active transport.

For details, see Use Case on Active Transport Index

Figure 1 shows a high level system architecture for the Tombolo Digital Connector. The figure shows the main system components and the interlinking between them. We also show the links to external data and systems.

- At the core of the digital connector is the **Local Datastore** where all incoming data is stored in a centralised data format.
- **Importers** are responsible for interpret the incoming data and code it in the centralised data format. We have both built-in importers for **publicly available datasources** as well as support for users to write their own importers for their **proprietary data**. If the proprietary data is available simple shape-files or simple csv files, it may be importable using a generic data importer without any additional implementation.
- **Exporters** are used to export data from the system. The output data can simply be the data values as they originated in the input data, or as a modelled combinations of various data-sources.
- For the modelling part we both have pre-defined **built-in model recipes** as well as support for the user to specify and export their own **proprietary model recipes**.

`images/high-level-system-architecture.png`

Fig. 4.1: High level system architecture

Figure 1: *Overview of the Tombolo Digital Connector components and data workflow.*

At the core of the Digital Connector there is a centralised data-format for urban data. We refer to the physical implementation of the data-format as the Local Datastore. In the current implementation the data is stored in a PostGIS database instance. Figure 1 shows the Entity Relationship Diagram for the Local Datastore.

images/tombolo-erd.png

Fig. 4.2: Entity Relationship Diagram

Figure 1: *ERD for the Tombolo Digital Connector centralised data-format (Local Datastore).*

In short, the datastore contains a collection of **subjects** (e.g. traffic-counters, road segments or neighbourhoods) and **attributes** (e.g. bicycle-count, population, etc.). For each subject and each attribute, there is associated a time-series of zero or more values. For attributes that do not change over time we also have the notion of fixed values (e.g., road names).

Below we describe the centralised data format in more detail. Database field names are written in italics and primary keys as boldface.

CHAPTER 5

Provider

Provider is a data object representing sources of data. A provider could be a governmental organisation, such as ONS; or a private entity supplying a publicly available data source, such as Twitter.

A provider consists of:

- ***label:*** a unique label for the provider
- *name:* a name of the provider

CHAPTER 6

Subject Type

Subject Type is the type of subjects imported into the system (see description of subjects below). Subject types can be various such as traffic counter, IoT sensor, street segment, building, lsoa, msoa, local authority, etc.

A subject type consists of:

- ***provider:*** a provider of the subject type. I.e., the organisation responsible for defining and providing digital version of the subject.
- ***label:*** a label for the subject type. The label is unique for each provider.
- *name*: a human readable name of the subject type.

CHAPTER 7

Subject

Subject is a data object representing any type of subject or entity, which will in most cases refer to geographic objects, although we could support other subject types such as individuals or businesses. We support a both physical geographic subjects, such as street segments and buildings, as well as logical geographic subjects such as a geo-coded tweet or image.

A subject consists of:

- ***subject type:*** type of the subject (See description above).
- ***label:*** a unique label for the subject. The labels are unique within a subject type.
- *name:* a human readable name of the subject.
- *shape:* the geometric shape of the subject, in case it has one.

CHAPTER 8

Attribute

Attribute is a data object representing anything that could be measured or calculated, such as population density (for an LSOA), CO2 concentration (for an air quality sensor), obesity rate (for a local authority), etc.

An attribute consists of:

- ***provider:*** refers to the organisation or source of the data values for this attribute.
- ***label:*** a label of the attribute, unique for each provider.
- *name:* a human readable name of the attribute (e.g. Population density, Obesity rate, CO2 concentration, etc.)
- *description:* a further description of the attribute, e.g. describing the methodology used to generate the attribute values.

CHAPTER 9

Fixed Value

Fixed Value is a fixed attribute that can be assigned to a Subject. Examples can be a the value of a road-type Attribute assigned to a road Subject.

A fixed value consists of:

- ***subject:*** a foreign key reference to a Subject.
- ***attribute:*** a foreign key reference to an Attribute.
- *value:* the actual value of the attribute for the subject, most often a string.

CHAPTER 10

Timed Value

Timed Value is a data object representing the value of an attribute for a certain subject, taken at a certain time point.

A timed value consists of:

- ***subject:*** a foreign key reference to a Subject.
- ***attribute:*** a foreign key reference to an Attribute.
- ***timestamp:*** time point to which the value refers. (E.g. 2015-03-04T10:33:44Z)
- *value:* the actual value for the attribute, subject and timestamp, most often a numeric value.

In this section we describe the Tombolo Digital Connector importers. The role of importers is to connect to external or local data sources and import the appropriate data by reformatting it into the centralised data format.

Importers are of three types:

- **Subject Importers** import only Subjects. Examples include the LSOA and MSOA importers for the ONS Geographies Portal and the Health Organisation importer for NHS Choices data.
- **Timed Value Importers** import only Timed Values for an externally defined set of Subjects. Examples include the ONS Census importer that import census values for geographies such as LSOAs, MSOAs and Local Authorities.
- **Mixed Importers** import both Subjects and Timed Values. Examples include the Traffic Counts importer which imports both the locations of traffic counts as Subjects and the actual traffic counts as Timed Values.

We have built-in importers for a range of mostly public and open datasets. Users can use these importers directly to import the data they need in the processing. Additionally we have support for users to extend the Digital Connector with their own user defined custom importers for their local proprietary datasets. In the code base we have support tools such as Excel and Shapefile data extraction tools to make it easier to write custom importers.

CHAPTER 11

Built-in importers

Below you can find a list of built-in importers that were available at the time of writing. For an up to date list look at the codebase.

- Department for Communities and Local Government (DCLG)
  - Indices of multiple deprivation (value importer)
- Department for Education (DfE)
  - List of all schools (subject importer)
- Department for Transport (DfT)
  - Accessibility of output areas (value importer)
  - Traffic counts (mixed importer)
- London Air Quality Network (LAQN)
  - Air quality data from Environmental Research Group Kings College London (mixed importer)
- London Datastore
  - Borough profiles (value importer)
  - Public Health Outcomes Framework (PHOF) indicators for London (value importer)
  - Walking and cycling information of London boroughs (value importer)
- NHS Choices
  - Health Organisations (subject importer)
- Office of National Statistics (ONS)
  - Output area geometries (subject importer)
  - Census data (value importer)
  - Claimants data (value importer)
  - Wages data (value importer)

- Open Street Map (OSM)
  - Cycling infrastructure (subjects importer)
  - Greenspace data (subjects importer)
  - Land-use data (subjects importer)
  - Road infrastructure (subjects importer)
- Public Health England (PHE)
  - Adult Obesity data (value importer)
  - Childhood Obesity data (value importer)
- Space Syntax
  - Open Map (mixed importer)
- Transport for London (TfL)
  - Transport stations importer (mixed importer)
- Twitter
  - Geo-coded tweet importer (subject importer)
- Generic Importers
  - Generic CSV Importer (value importer / subject importer / mixed importer)

In the development of importers we have taken a pragmatic approach where we implement importers on need-to-use basis where the city challenges have been in the driver seat. As the project progresses we will be constantly supporting more data sources.

In this section we describe the Tombolo Digital Connector fields and how modelling can be seen as fields. As with importers we have a set of built-in fields, recipes and models, together with a field specification language where users can define their own custom fields and models.

See also FAQ about fields and model recipes

CHAPTER 12

Value Fields

Value fields are the most basic fields. Their purpose is to being able to export existing raw imported data values or to feed them to one of the nested fields introduced later on in this section.

Currently we have implemented the following value fields:

- **Fixed Annotation Field:** Returns a fixed value for annotation purposes.
- **Fixed Value Field:** Returns the Fixed Value of a specified Attribute for a given Subject.
- **Latest Value Field:** Returns the latest Timed Value for a particular Attribute on the given Subject.
- **Subject Latitude Field:** Returns the latitude of the centroid of the Subject.
- **Subject Longitude Field:** Returns the longitude of the centroid of the Subject.
- **Subject Name Field:** Returns the name of the Subject.
- **Values By Time Field:** Returns all Timed Values on an Attribute for a given Subject.

For an up-to-date list, see here: value fields

CHAPTER 13

Transformation Fields

Transformation fields take as input a set of one or more fields and produce a new field by transforming the values of the input fields.

- **Arithmetic Field:** Takes as input an operation, and two fields. It returns for a given Subject the value resulting from applying the operation on the two field values. Supported operations are addition, subtraction, multiplication and division.
- **Percentiles Field:** Field that returns for a subject and an attribute the percentile in which its value falls, compared to all the values for the same attribute for a set of subjects. Percentiles can be calculated either over the output subjects or any other specified set of subjects. E.g. we could output the quartiles value for the population density of all LSOAs in Leeds where the quartiles boundaries are calculated using population density values for all LSOAs in England.
- **Field Value Sum Field:** Takes a list of fields as input and returns a field consisting of the sum of the input fields (To be replaced by generalised Arithmetic Field).
- **Fraction Of Total Field:** For a subject, returns the sum of its Timed Values for a list of dividend attributes divided by a divisor attribute (To be replaced by generalised Arithmetic Field).

For an up-to-date list, see here: transformation fields

CHAPTER 14

Aggregation and Disaggregation Fields

The aggregation and disaggregation fields are types of transformation fields where the transformation is not only a combination of other fields but also either aggregates values from fine granularities to a coarse granularity or de-aggregates values from coarse granularities to finer granularities.

- **Geographic Aggregation Field:** A field that calculates the value of a field for every other Subject within its geometry and performs some aggregation function on the result. Currently we support sum and mean value aggregations. E.g. a cycle-traffic value for a local authority can be calculated by averaging the cycle-counts from all traffic counters located within that local authority.
- **Map To Containing Subject Field:** This field will find a subject of a given type that contains the provided subject, and then associate a field value with that new subject. For example, if the containing Subject Type is a 'City' and it is given a subject representing a building, it will assign the containing city's field value to the building subject.
- **Map To Nearest Subject Field:** A field that finds the nearest subject of a given Subject Type and then evaluate the field specification with that new subject. For example, if the nearest Subject Type is 'Street' and it is given a subject representing a building, it will evaluate the field specification with a subject representing the Street that building is on.

For an up-to-date list, see here: aggregation fields

CHAPTER 15

Modelling Fields and Built-in Models

Modelling Fields and Built-in models are an important part of the Tombolo Digital Connector. It allows users to share definitions of custom fields and models. They are also used to encode built-in models that have been developed within the Tombolo project to address the City Challenges. Basic Modelling Field: A field that takes as input a specification (recipe) of a potentially complex field and returns a value that is calculated according to the specification. The recipe/specification can also be seen as a model.

At the time of writing we include the following Built-in models:

- **'Active Transport Index <Use-Case-on-Active-Transport-Index>'__:** A modelling field combining traffic counts from Department for Transport, cycling infrastructure from Open Street Map and travel to work information from the UK Census.
- **Social Isolation Score:** A modelling field for applying the Age UK model described in the city challenges description below.

These models are them selves combinations of sub models, e.g. for calculating the fraction of households renting in an LSOA. For browsing the available built-in models see modelling fields

CHAPTER 16

Exporters

Since the main goal of the Tombolo Digital Connector is to connect urban data and urban models, there is a large set of urban analytics and model building that takes place outside of the connector. To allow for connections with external systems the Digital Connector provides support for exporting data and model output. Currently there are two data formats supported.

- **GeoJson** is one of the most common data format for geographic data. It allows for easy integration between the Tombolo Digital Connector and Geographic Information Systems such as QGIS.
- **CSV** is one of the most common data format for relational data. It allows for easy integration between the Connector and various data processing and analytics tools.

The workflow of exporting data is core functionality of the current state of the Tombolo Digital Connector. The user creates a recipe file where they describe the output data they would like to get. The recipe file consists of four parts:

- **Subjects:** The user can specify the set of subjects for which data and models are to be exported. As an example, subjects can be all spatial network segments for a specific geographic area, all LSOAs within a certain geographic area, etc.
- **Data-sources:** A list of data-sources needed to be imported in order to export the data. As an example, data-sources can be the Space Syntax Open Space Map (SSx OSM) for the Royal Borough of Greenwich, traffic counts for London from Department for Transport (DFT), etc.
- **Fields:** A list of fields that are to be returned for each subject. As an example, for a set of spatial network segments the user could specify to export the connectivity of each segment according to SSx OSM, the nearest DfT traffic counts for that segment (if available) and a deprivation value for that segment disaggregated from the LSOA level deprivation scores from Department for Communities and Local Government (DCLG). In case a field is a transformation or a modelling field, the needed computation is performed at the time of exporting.
- **Exporter:** The name of the exporter to be used. E.g. GeoJson or CSV.

Note that in the case of built-in model fields, the user does not need to specify the data-sources since they are already included in the built-in model recipe.

The core means of using the Tombolo Digital Connector is to create a data export recipes, describing the data or models that we want to get out of the system, without describing how to generate the data. The recipe language is expressed in the well-known JSON format.

CHAPTER 17

Data export recipe

A data export recipe consists of two parts:

- **Dataset**: is a description of the data to be exported. The format of the dataset recipe is explained in the *Dataset recipe* section below.
- **Exporter**: is the canonical name of the Java class to be used to export the data. At the time of writing there are two types of exporters, one for CSV output and one for GeoJson.

Example data export specification for GeoJson output:

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

CHAPTER 18

Dataset recipe

A dataset recipe is composed of three parts:

- **Subjects**: Specifies the Subjects for which we return data in the final dataset. I.e., return all LSOAs in Milton Keynes, return all traffic counters in Greenwich, or return all local authorities in England. See further details in the *Subject recipe* section.
- **Datasources**: A list of data-sources needed to be imported in order to export the data. As an example, data-sources can be the Space Syntax Open Space Map (SSx OSM) for the Royal Borough of Greenwich, traffic counts for London from Department for Transport (DFT), etc. See further details in the *Datasource recipe* section.
- **Fields**: A list of Fields that are to be returned for each subject. As an example, for a set of spatial network segments the user could specify to export the connectivity of each segment according to SSx OSM, the nearest DfT traffic counts for that segment (if available) and a deprivation value for that segment disaggregated from the LSOA level deprivation scores from Department for Communities and Local Government (DCLG). In case a field is a transformation or a modelling field, the needed computation is performed at the time of exporting. See further details in the *Field recipe* section.

Example dataset specification skeleton:

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

CHAPTER 19

Subject recipe

A subject recipe is has two mandatory parts:

- **Provider**: specifies the provider of the subject subject to be returned.
- **Subject type**: specifies the type of the subject to be returned. At the time of writing we support various types, such as, local authority, msoa, lsoa, trafficCounter, SSxNode (a node in a Space Syntax graph), gpSurteries, etc.

Example of a simple subject specification where we return all traffic counters that have been imported.

```
{
  "provider" : "uk.gov.dft",
  "subjectType" : "trafficCounter"
}
```

Additionally a subject recipe can have two types of so-called match rules where we can restrict further the set of Subjects returned:

- **match-rule**: is a filter where we can restrict the returned subjects based on name or label. The match-rule is further composed of:
  - **attribute**: is the attribute on which we would like to filter. E.g. name or label.
  - **pattern**: is a string pattern that the specified attribute should match. We use the SQL like syntax using % as the wildcard. E.g. for filtering all strings that start with the string 'Leeds' we use the pattern 'Leeds%'.
- **geo-match-rule**: is a filter where we can restrict the returned subjects based on geographic constraints.

Example where we return all LSOAs whose name starts with the string 'Leeds':

```
{
  "provider" : "uk.gov.ons",
  "subjectType" : "lsoa",
  "matchRule": {
    "attribute": "name",
    "pattern": "Leeds%"
  }
}
```

Example where we return all London boroughs based on filtering for the label prefix ‘E090’:

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

Example where we return all traffic-counters in Greenwich and Islington:

```
{
  "provider" : "uk.gov.dft",
  "subjectType" : "trafficCounter",
  "geoMatchRule" : {
    "geoRelation" : "within",
    "subjects" : [
      {
        "provider" : "uk.gov.ons",
        "subjectType" : "localAuthority",
        "matchRule" : {
          "attribute" : "name",
          "pattern" : "Greenwich"
        }
      },
      {
        "provider" : "uk.gov.ons",
        "subjectType" : "localAuthority",
        "matchRule" : {
          "attribute" : "name",
          "pattern" : "Islington"
        }
      }
    ]
  }
}
```

CHAPTER 20

Datasource recipe

The datasource recipe consists of two fields:

- **importer-class**: Which Java class is used to import the data.
- **datasource-id**: The identifier of the dataset to be imported. This value is unique within each importer class.

Example for LSOA geographies

```
{
  "importerClass": "uk.org.tombolo.importer.ons.OaImporter",
  "datasourceId": "lsoa"
}
```

Examples for importing the ONS Census dataset for overcrowding:

```
{
  "importerClass": "uk.org.tombolo.importer.ons.CensusImporter",
  "datasourceId": "QS408EW"
}
```

20.1 Datasource geography scope

Some data providers support customisable downloading of their data based on geographic area. E.g. Department of Transport provides traffic counts for individual regions or local authorities.

- **geographyScope**: is a list of geographic areas to import. For the supported values see the code for the corresponding importer.

Example for importing DfT traffic-counts for London:

```
{
  "importerClass" : "uk.org.tombolo.importer.dft.TrafficCountImporter",
  "datasourceId" : "trafficCounts",
  "geographyScope" : ["London"]
}
```

20.2 Datasource temporal scope

Some datasources support configurable downloads for some time periods.

- **temporalScope**: is a list of temporal references for which data can be imported. For the supported values see the code for the corresponding importer.

CHAPTER 21

Field recipe

The field recipe contains a list of fields that you want to output. Fields can be as simple as the latest value for a certain attribute, such as population density, or a potentially complex model, such as one for estimating the risk of social isolation among the elderly. All fields have two default fields (sic):

- **fieldClass**: is the Java class used to calculate the field value
- **label**: the label that the recipe creator chooses to associate with the field

In addition each field has additional fields (sic) depending on the type of the field. For example a latest-value-field you need to specify the attribute for which you want the latest value. E.g.,

```
"field": {
  "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
  "label": "Count of cars and taxis",
  "attribute": {
    "provider": "uk.gov.dft",
    "label": "CountCarsTaxis"
  }
}
```

As another example the arithmetic-field takes as an input the operation you want to apply and two fields

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField",
  "label": "An example of adding two fields",
  "operation": "add",
  "field1": INSERT-A-FIELD-RECIPE,
  "field2": INSERT-A-FIELD-RECIPE
}
```

Further examples can be found in the Fields and Models description page.

CHAPTER 22

Notes

- Introduce two types of scope.
  - Subject scope (optional)
  - Normalisation scope (optional)

This questionnaire is aimed at evaluating the effort required to implement a Tombolo importer for a data source. Before answering this questionnaire, please familiarise yourself with the Tombolo data model.

CHAPTER 23

Describe your importer

- What are the subjects related to this dataset?
- Are the subjects imported by this datasource or do you link to subjects in an external datasource?
- What are the fixed value attributes for the subjects?
- What are the timed value attributes for the subjects?

In this page we will give a tutorial of how to use the Tombolo Digital Connector.

CHAPTER 24

Install system

First of all you should follow the Quick Start guide to get the digital connector up and running by installing requirements, configuring the project and setting up both the main and test databases.

CHAPTER 25

Export data

Having installed the system, it is time to run a data export recipe. In the Use Case on Cycling and Air Quality we described an example recipe where we output, for every borough in London, information about NO2 concentration and the ratio between the bicycle traffic and car traffic (see recipe). To run the export recipe, run the following command from the root directory of the Tombolo Digital Connector:

```
gradle runExport \
  -PdataExportSpecFile='src/main/resources/executions/examples/london-cycle-traffic-
↪air-quality.json' \
  -PoutputFile='london-cycle-traffic-air-quality-output.json'
```

As mentioned in the use case description, this will give you a GeoJson file with the cycling and air quality indicators for each of the 33 London boroughs. Opening the Json file in QGIS and using a quantile-based colouring of the boroughs should give you an image similar to the one below (See QGIS tutorial for reference).

images/tutorial-london-cycle-counts-la.png

Fig. 25.1: London borough cycle to car count ratio

CHAPTER 26

Change granularity

Suppose you want to get the cycling information at a finer granularity, say at LSOA level. Copy the data export recipe into a new file called `london-cycle-traffic-air-quality-lsoa.json`.

Change the subjects clause from outputting all 33 London boroughs to outputting all LSOA geographies that fall inside the 33 London boroughs. I.e., change:

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

to:

```
"subjects" : [
  {
    "subjectType": "lsoa",
    "provider": "uk.gov.ons",
    "geoMatchRule": {
      "geoRelation": "within",
      "subjects": [
        {
          "subjectType": "localAuthority",
          "provider": "uk.gov.ons",
          "matchRule": {
            "attribute": "label",
            "pattern": "E090%"
          }
        }
      ]
    }
  }
],
```

and add the following datasource to the list of datasources:

```
{
  "importerClass" : "uk.org.tombolo.importer.ons.OaImporter",
  "datasourceId" : "lsoa"
},
```

What this does is that it tells the Digital Connector to output all LSOA geographies that fall inside the 33 London boroughs (We are here basing our work on the fact that London boroughs can be identified by a label staring with `E090`) (see the full recipe). Run the new recipe by executing the command:

```
gradle runExport \
  -PdataExportSpecFile='london-cycle-traffic-air-quality-lsoa.json' \
  -PoutputFile='london-cycle-traffic-air-quality-lsoa-output.json'
```

When looking at the output from the Digital Connector you will notice that you get very many warnings. This is because that there are very many LSOAs that do not have either a traffic counter in them or an air quality sensor, and hence there is no data to output. Yet, the Digital Connector does return a GeoJson file with the LSOA geographies. If you, again, use QGIS to create a quantile-based colouring of the LSOAs, you should get an image that looks like the one below.

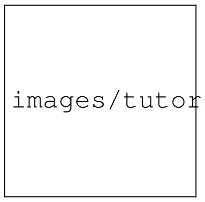

Fig. 26.1: London borough cycle to car count ratio

If you compare this image with the one from the first export you can see that it gives a finer granularity results, but the data is sparse due to the fact that not all LSOAs have a traffic counter. One way to overcome this sparseness of traffic counters and air quality sensors is to use a so-called back-off field. Back-off fields allow the user to output a different field value for a subject if none exists. In our case, if a traffic count value does not exist for an LSOA, we could instead output the traffic count value for the surrounding MSOA. If that value does not exist either, we could back-off to the local authority value.

Before we dive into back-off fields, it is better to introduce the the notion of modelling-fields.

CHAPTER 27

Modelling Fields

In the running example we have been using a combination of geographic aggregation fields and arithmetic fields to get the aggregated NO2 value for an area and the aggregated bicycle-to-car ratio for an area. These data aggregations and transformations are fairly basic and likely to be useful in city data analysis projects beyond this example. Hence we have wrapped them up as model recipes that can be used across different jobs without copying and pasting the entire code.

In order to demonstrate this, copy your lsoa data export recipe into a new file called `london-cycle-traffic-air-quality-lsoa-modelling.json`.

In this file you can replace the actual calculation with the corresponding pre-defined model recipe. I.e. replace:

```
{
  "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"
    }
  }
}
```

with the corresponding pre-defined model of aggregating NO2:

```
{
  "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
  "label": "NitrogenDioxide",
  "recipe": "environment/laqn-aggregated-yearly-no2"
}
```

Similarly the calculation of bicycle to car ratio:

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField",
  "label": "BicycleFraction",
  "operation": "div",
  "field1": {
    ...
  },
  "field2": {
    ...
  }
}
```

Can be replaced by the corresponding pre-defined model:

```
{
  "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
  "label": "BicycleFraction",
  "recipe": "transport/traffic-counts-aggregated-bicycles-to-cars-ratio",
  "datasources": [
    {
      "importerClass" : "uk.org.tombolo.importer.dft.TrafficCountImporter",
      "datasourceId" : "trafficCounts",
      "geographyScope" : ["London"]
    }
  ]
}
```

Note that in this case we additionally pass a list of datasources to the traffic counts model. This is because, by default, the model is set up for calculating the cycle to car ratio nationwide and will therefore import traffic counts for the entire country. However, since in this tutorial we are only interested in the data for London, we override the data-set import to only the London area.

This recipe is quite simpler than before (see full recipe). Now, run the new recipe by executing the command:

```
gradle runExport \
  -PdataExportSpecFile='london-cycle-traffic-air-quality-lsoa-modelling.json' \
  -PoutputFile='london-cycle-traffic-air-quality-lsoa-modelling-output.json'
```

If you look at the resulting file in QGIS, you will see that you get the same output as before but by utilising more simplified and re-usable code.

CHAPTER 28

Back-off Fields

Now that we have simplified the export recipe, we can go back to our intention to use back-off fields to overcome the sparseness of traffic counters and air quality sensors. A Back-off field takes as input an array of fields. It will try to calculate a value for each of the fields in order, and when it finds one it will output that one.

In order to demonstrate this, copy your modelling data export recipe into a new file called `london-cycle-traffic-air-quality-lsoa-backoff.json`.

Then replace the NO2 aggregation calculation:

```
{
  "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
  "label": "NitrogenDioxide",
  "recipe": "environment/laqn-aggregated-yearly-no2"
}
```

with a back-off field that first tries to aggregate the NO2 values for the output geography (LSOAs). If no air quality sensor exists within the LSOA, it tries to aggregate NO2 values for the surrounding MSOA. If no air quality sensor exists within the MSOA, it will aggregate NO2 values for the surrounding local authority (borough). The resulting back-off field looks like this:

```
{
  "fieldClass": "uk.org.tombolo.field.aggregation.BackOffField",
  "label": "NitrogenDioxide",
  "fields": [
    {
      "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
      "recipe": "environment/laqn-aggregated-yearly-no2"
    },
    {
      "fieldClass": "uk.org.tombolo.field.aggregation.MapToContainingSubjectField",
      "subject": {
        "provider": "uk.gov.ons",
        "subjectType": "msoa"
      },
      "field": {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "recipe": "environment/laqn-aggregated-yearly-no2"
      }
    },
    {
      "fieldClass": "uk.org.tombolo.field.aggregation.MapToContainingSubjectField",
      "subject": {
        "provider": "uk.gov.ons",
        "subjectType": "localAuthority"
      },
      "field": {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "recipe": "environment/laqn-aggregated-yearly-no2"
      }
    }
  ]
}
```

The back-off field for the ratio of bicycles to cars is exactly the same, only with changing the corresponding recipe (see full recipe). As before you run the recipe with the command:

```
gradle runExport \
  -PdataExportSpecFile='london-cycle-traffic-air-quality-lsoa-backoff.json' \
  -PoutputFile='london-cycle-traffic-air-quality-lsoa-backoff-output.json'
```

Using QGIS to visualise the back-off model in a similar manner as done above, we get a much less sparse data output as shown in the image below.

images/tutorial-london-cycle-counts-backoff.png

Fig. 28.1: London borough cycle to car count ratio

As part of the Tombolo project we have built an application called City Index Explorer. The application functions as a demonstrator of the possibilities for combining various urban data sources into an urban model. One of the indices we have developed for demonstration is the Active Transport Index. The index combines various data sources to assign a score to each LSOA representing the use and potential for active transport.

In particular, the index consists or three components: * **Cycle traffic**: The bicycle counts as a fraction of the total traffic count in the LSOA using traffic counter information from Department for Transport. * **Cycle infrastructure**: The fraction of the entire road infrastructure that is fitted with cycling infrastructure (such as cycle lanes) using information from Open Street Map. * **Active commute**: The fraction of total commuters that commute principally by either cycling or walking using information from the 2011 Census.

Below we will describe the generation of the index in detail, but see also the model recipe for the active transport index and recipe for exporting as GeoJson the active transport index, together with its components for all LSOAs in England and Wales.

CHAPTER 29

Export recipe

As with other export recipes, the active transport index export recipe has 4 main parts.

- **subjects**: Tells the digital connector to calculate values for each LSOA in the UK.
- **datasources**: Tells which datasources need to be imported. In this case only the LSOAs, since any additional datasources needed by the index are covered by the recipe for the respective index or index component.
- **fields**: Tells the digital connector to export 4 fields: the index itself, together with its 3 components. The last 3 fields are not necessary when exporting the index, but for the city index explorer application we do need this information for data visualisation purposes.
- **exporter**: Tell the digital connector to export the data as GeoGson.

```
{
  "dataset": {
    "subjects": [
      {
        "provider": "uk.gov.ons",
        "subjectType": "lsoa"
      }
    ],
    "datasources": [
      {
        "importerClass": "uk.org.tombolo.importer.ons.OaImporter",
        "datasourceId": "lsoa"
      }
    ],
    "fields": [
      {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "label": "index:active_transport",
        "recipe": "city-indices/active-transport/ActiveTransportIndex"
      },
      {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "label": "component:proportion_of_cycles_to_cars",
        "recipe": "city-indices/active-transport/CyclingCount-lsoa"
      },
      {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "label": "component:cycle_lane_score",
        "recipe": "city-indices/active-transport/CycleLaneCount-lsoa"
      },
      {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "label": "component:travel_to_work_score",
        "recipe": "city-indices/active-transport/ActiveTransportToWork"
      }
    ]
  },
  "exporter" : "uk.org.tombolo.exporter.GeoJsonExporter"
}
```

Below we will explain the active transport index and its components in more detail.

CHAPTER 30

Active Transport Index

As described above, the active transport index is composed of three components. The index is simply the sum of those three components, implemented as a list-arithmetic-field using the addition operation.

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ListArithmeticField",
  "operation": "add",
  "fields": [
    {
      "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
      "recipe": "city-indices/active-transport/CyclingCount-lsoa"
    },
    {
      "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
      "recipe": "city-indices/active-transport/CycleLaneCount-lsoa"
    },
    {
      "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
      "recipe": "city-indices/active-transport/ActiveTransportToWork"
    }
  ]
}
```

In the future we might consider implementing the index as a “linear-combination-field”. Pending implementation of that field.

Each component is described below.

CHAPTER 31

Cycle traffic

The **‘cycle-traffic <>‘__** component uses the Department for Transport traffic counts to calculated ratio between cycle traffic count and the sum of both cycle and car traffic counts. The field is implemented as a back-off-field where we first try to calculate a value for the corresponding LSOA. If there is no traffic counter within the LSOA, we back-off to outputting the ratio based on all traffic counters in the surrounding MSOA. If no traffic counters exist in the MSOA, we back-off to the value for the surrounding local-authority. Finally, for local-authorities with no traffic counts, a default value of zero is returned.

The implementation of the back-off field is as follows:

```
{
  "fieldClass": "uk.org.tombolo.field.aggregation.BackOffField",
  "fields" : [
    {
      "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
      "recipe": "city-indices/active-transport/CyclingCount"
    },
    {
      "fieldClass": "uk.org.tombolo.field.aggregation.MapToContainingSubjectField",
      "subject": {
        "provider": "uk.gov.ons",
        "subjectType": "msoa"
      },
      "field": {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "recipe": "city-indices/active-transport/CyclingCount"
      }
    },
    {
      "fieldClass": "uk.org.tombolo.field.aggregation.MapToContainingSubjectField",
      "subject": {
        "provider": "uk.gov.ons",
        "subjectType": "localAuthority"
      },
      "field": {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "recipe": "city-indices/active-transport/CyclingCount"
      }
    },
    {
      "fieldClass": "uk.org.tombolo.field.value.FixedAnnotationField",
      "value" : "0.0"
    }
  ]
}
```

where cycling-count is defined using the built-in fields for aggregating traffic counts:

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField",
  "operation": "div",
  "field1": {
    "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
    "recipe": "transport/traffic-counts-aggregated-bicycles"
  },
  "field2": {
    "fieldClass": "uk.org.tombolo.field.transformation.ListArithmeticField",
    "operation": "add",
    "fields": [
      {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "recipe": "transport/traffic-counts-aggregated-bicycles"
      },
      {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "recipe": "transport/traffic-counts-aggregated-cars"
      }
    ]
  }
}
```

see further the built-in recipes for aggregating bicycle and car traffic counts.

CHAPTER 32

Cycling infrastructure

The cycling infrastructure component of the active transport index is also a back-off field. In essence it uses Open Street Map data to calculate how large fraction of the entire road network is marked as having cycling infrastructure. The back-off field is implemented in similar way as above where we first try to calculate a value for the LSOA, but if none is available we first back off to the surrounding MSOA and eventually to the surrounding local authority. It is implemented as follows:

```
{
  "fieldClass": "uk.org.tombolo.field.aggregation.BackOffField",
  "label": "Cycleway to Highway Count Ratio",
  "fields" : [
    {
      "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
      "label": "Cycleway to Highway Count lsoa",
      "recipe": "city-indices/active-transport/CycleLaneCount"
    },
    {
      "fieldClass": "uk.org.tombolo.field.aggregation.MapToContainingSubjectField",
      "label": "Cycleway to Highway Count msoa",
      "subject": {
        "provider": "uk.gov.ons",
        "subjectType": "msoa"
      },
      "field": {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "label": "Cycleway to Highway Count",
        "recipe": "city-indices/active-transport/CycleLaneCount"
      }
    },
    {
      "fieldClass": "uk.org.tombolo.field.aggregation.MapToContainingSubjectField",
      "label": "Cycleway to Highway Count localAuthority",
      "subject": {
        "provider": "uk.gov.ons",
        "subjectType": "localAuthority"
      },
      "field": {
        "fieldClass": "uk.org.tombolo.field.modelling.SingleValueModellingField",
        "label": "Cycleway to Highway Count",
        "recipe": "city-indices/active-transport/CycleLaneCount"
      }
    },
    {
      "fieldClass": "uk.org.tombolo.field.value.FixedAnnotationField",
      "label" : "default value",
      "value" : "0.0"
    }
  ]
}
```

where the CycleLaneCount model is an arithmetic-field where we divide the count of road segments with cycling infrastructure with the total count of road segments. In both cases the counts are calculated by using a attribute-matcher-field together inside a geographic-aggregation-field that aggregates over the corresponding geography (LSOA, MSOA or local-authority). It is implemented as follows.

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField",
  "label": "Cycle to Road count Ratio",
  "operation": "div",
  "field1": {
    "fieldClass": "uk.org.tombolo.field.aggregation.GeographicAggregationField",
    "label": "Cycle Lane Count",
    "function": "sum",
    "subject" : {
      "provider": "org.openstreetmap",
      "subjectType": "OSMEntity"
    },
    "field": {
      "fieldClass": "uk.org.tombolo.field.assertion.AttributeMatcherField",
      "label": "Assert Cycle Lane",
      "attributes": [
        {
          "provider": "org.openstreetmap",
          "label": "cycleway"
        },
        {
          "provider": "org.openstreetmap",
          "label": "cycleway:left"
        },
        {
          "provider": "org.openstreetmap",
          "label": "cycleway:right"
        },
        {
          "provider": "org.openstreetmap",
          "label": "cycleway:oneway"
        },
        {
          "provider": "org.openstreetmap",
          "label": "cycleway:oneside"
        },
        {
          "provider": "org.openstreetmap",
          "label": "cycleway:otherside"
        }
      ],
      "field": {
        "fieldClass": "uk.org.tombolo.field.value.FixedAnnotationField",
        "value": "1"
      }
    }
  },
  "field2": {
    "fieldClass": "uk.org.tombolo.field.aggregation.GeographicAggregationField",
    "label": "Highway Count",
    "function": "sum",
    "subject" : {
      "provider": "org.openstreetmap",
      "subjectType": "OSMEntity"
    },
    "field": {
      "fieldClass": "uk.org.tombolo.field.assertion.AttributeMatcherField",
      "label": "Assert Highway",
      "attributes": [
        {
          "provider": "org.openstreetmap",
          "label": "highway"
        }
      ],
      "field": {
        "fieldClass": "uk.org.tombolo.field.value.FixedAnnotationField",
        "value": "1"
      }
    }
  }
}
```

CHAPTER 33

Active commute

The active-commute component uses data from the travel-to-work dataset from the UK census (qs701ew). It adds the number of people who either walk or cycle to work and divides by the total population size. The field code is as follows:

```
{
  "fieldClass": "uk.org.tombolo.field.transformation.ArithmeticField",
  "label": "Active Transport to Work",
  "operation": "div",
  "field1": {
    "fieldClass": "uk.org.tombolo.field.transformation.ListArithmeticField",
    "label": "Sum Active modes",
    "operation": "add",
    "fields": [
      {
        "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
        "attribute": {
          "provider": "uk.gov.ons",
          "label": "Method of Travel to Work: On foot"
        }
      },
      {
        "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
        "attribute": {
          "provider": "uk.gov.ons",
          "label": "Method of Travel to Work: Bicycle"
        }
      }
    ]
  },
  "field2": {
    "fieldClass": "uk.org.tombolo.field.value.LatestValueField",
    "attribute": {
      "provider": "uk.gov.ons",
      "label": "Method of Travel to Work: All categories: Method of travel to work"
    }
    }
}
```

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.

CHAPTER 34

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 chose to export the data as GeoJson.

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 NO2 levels across each borough.

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

34.1 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.

34.2 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 datafiles 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.

34.3 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"
```

```
            }
        }
    }
}
```

CHAPTER 35

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
      }
    },
    ...
  ]
}
```

CHAPTER 36

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.

images/london-cycle-counts.png

Fig. 36.1: London Bicycle Fraction

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

images/london-no2-concentration.png

Fig. 36.2: London Nitrogen Dioxide

Heat map of annual nitrogen dioxide levels.

A| B| C| D| E| F| G| H| I| J| K| L| M| N| O| P| Q| R| S| T| U| V| X| Y| Z| W|

CHAPTER 37

A

37.1 Attribute

A data object representing anything that could be measured or calculated, such as population density (for an LSOA), CO2 concentration (for an air quality sensor), obesity rate (for a local authority), etc. See further

CHAPTER 38

B

CHAPTER 39

C

CHAPTER 40

D

CHAPTER 41

E

CHAPTER 42

F

CHAPTER 43

G

CHAPTER 44

H

CHAPTER 45

I

CHAPTER 46

J

CHAPTER 47

K

CHAPTER 48

L

48.1 LSOA

Lower Level Super Output Area (LSOA) is a geography defined by the Office for National Statistics and commonly used form many statistical outputs.

CHAPTER 49

M

CHAPTER 50

N

CHAPTER 51

O

CHAPTER 52

P

CHAPTER 53

Q

CHAPTER 54

R

CHAPTER 55

S

CHAPTER 56

T

CHAPTER 57

U

CHAPTER 58

V

CHAPTER 59

X

CHAPTER 60

Y

CHAPTER 61

Z

CHAPTER 62

W

- General
- Data
- Fields
- Model recipes

CHAPTER 63

General

63.1 What is the difference between the Tombolo project and Tombolo Digital Connector?

The Tombolo Digital Connector is one of the pieces of software developed within the Tombolo project. The Tombolo project has a much wider scope with, among others, work packages around city engagement and creating a network of practitioners of urban modellers.

63.2 What is the difference between the Tombolo Digital Connector and a GIS system?

The Tombolo Digital Connector is a tool for downloading and combining urban data. It can be used to share data combination recipes. The Tombolo Digital Connector is not a data-storage system (although one of its components is a database where data is stored temporarily).

GIS systems are a more generic tools for manipulating data. The Tombolo Digital Connector can be used to download and combine datasets that can then be manipulated further in a GIS system.

63.3 What is the difference between the Tombolo Digital Connector and data analytics tools?

The Tombolo Digital Connector is a tool for downloading and combining urban data. It can be used to share data combination recipes. The Tombolo Digital Connector is not a generic purpose data analytics or data visualisation tool. The Tombolo Digital Connector can be used to download and combine datasets that can then be manipulated further in a data analytics or data visualisation tools.

63.4 Is it a middleware or API or platform?

It is a tool for joining heterogeneous datasets together.

CHAPTER 64

Data

64.1 Does Tombolo provide me with data?

No, it provides you a tool to get the data.

64.2 Can the digital connector solve the problem of missing values?

No, but you can write and importer for it or introduce a back-off field.

64.3 What is the difference between fixed values and timed values?

In the centralised data-format, subjects can be assigned two types of attributes: fixed and timed. Fixed attributes are the ones whose values do not have a time associated with them, e.g. name of a road segment, type of a road, category of an Open Street Map geography, etc. Timed attributes are the ones whose values have a time associated with them. e.g. number of deprived households in a certain LSOA according to the 2011 Census, NO concentration for an air quality sensor at 2pm on a certain day, etc.

CHAPTER 65

Fields

65.1 What is the difference between a Field and an Attribute?

Attributes are used to represent characteristics of Subjects as they are imported from the external datasources. Fields are used to represent the data that we want to export. In the simplest form a Field will simply specify which attribute we would like to export (see Value Fields below). However, in most cases the output is likely to be a more complicated, such as an arithmetic calculation over multiple attributes (see Transformation Fields below), aggregation of attribute values from fine grained geometries to coarse grained geometries (see Aggregation Fields below), etc.

In short, Attributes represent data input and Fields represent data output.

CHAPTER 66

Model recipes

66.1 What is the difference between a model and a recipe?

A recipe describes how to combine data to create a model. A model is the result of applying the recipe to a data-set.