Appendix 2b – Data preparation: Building preparation

Why building preparation?

To create the Detour Index (DI) and the Local Significance (LS) index, we are only interested in the residential buildings.

By downloading all buildings from OpenStreetMap (OSM) that are located inside a certain area, we have also downloaded buildings with non-residential function, though.

In addition to filtering out non-residential buildings, we need population data attached to buildings to calculate the LS values.

These tasks will be executed during the building preparation process.

Building preparation

During the process of downloading the building data from the Overpass-API, we retrieved a plethora of polygons representing all buildings in a city.

Attached to the building data there is often some information on the building type or use.

Since the OSM contributors that create the building data have different levels of experience and different motivations, the annotation of these information is rather inconsistent, so we cannot solely rely on it.

To circumvent the inconsistent annotation, we will compare the OSM buildings with the urban atlas (UA) data that we have downloaded manually.

UA land-use data provides information on residential areas of cities on a European level.

We can utilize the information on residential areas to filter out all OSM buildings that are located outside of these areas.

This way we try to retrieve all residential buildings of a city.

A side-effect of this procedure is, that we can in the same step join the population data that is contained in the UA dataset to the OSM buildings.

Information on the population is mandatory for calculating the LS index.

For further steps in the analysis it is also required, that we proceed with the centroids of the buildings, rather than the polygons.

We have bundled these tasks in the *buildingPrep* function of our data preparation workflow.

Functionality

The building preparation process is integrated in the *buildingPrep* function in our R-program.

The buildingPrep function takes as input i.) the city code in URAU format of the city that should be prepared, ii.) an input directory containing the layer with city boundaries, the downloaded OSM buildings, the directory containing the UA data and iii.) and output directory for writing the ready-to-use building layer to disk.

All directories containing the required input will be guessed automatically from the input directory.

If desired, individual file names can be specified.

The process of building preparation follows these steps: a) loading UA data, b) filtering OSM data, c) adding population values, d) adding building ID, e) calculating building centroids.

\textit{Loading UA data:} The UA dataset usually encompasses and area larger than the city core. To reduce this dataset and, thus, saving disk space and computation time, we reduce it to the city outline. For the reduction of the UA dataset, we first load the city boundary with the corresponding URAU code from our layer with all European cities. When loading the UA data from file, the city boundary will be converted to a well-known-text (WKT) filter. With the WKT filter we will only load UA data into the machines memory that is located inside the city’s boundaries. Consecutively, the UA data will be filtered so only the residential classes (11100, 11210, 11220, 11230, 11240, 11300) and the columns that are necessary for later steps will be left (population, identifier and class code). The UA dataset is now ready for the further building preparation.

\textit{Filtering OSM data:} The OSM building data that we downloaded from the Overpass API are stored in 2x2 km tiles. Like with the network data, there will be a surplus of data in those tiles that are overlapping the city boundary. Upon loading the individual OSM building tiles into the machine’s memory, we use a WKT filter similar to the one we used before to keep only entries that are located inside the city’s boundary. In the following steps, invalid and empty geometries will be filtered out and the geometries will be cast to polygons. These steps have proven necessary to prevent broken geometries from causing errors in later processing steps. Also, splitting multipolygons that are potentially still contained in the dataset into their respective parts has proven to reduce errors in the later workflow. In a further filtering step, we execute a spatial join to attach the UA residential data to the OSM buildings. All buildings that have no UA data joined to them can now be filtered out. In the last OSM filtering step, we make use of the information that is stored in the OSM data. For a fraction of the building data, the ‘*building*’ column contains information on the use or function of a building. Usually the value is merely ‘Yes’ or ‘NA’, but some for larger, non-residential buildings there is often a descriptive character string in here. On gathering the unique values of the building column from the OSM data of several cities, we could make out a list of words that could safely be excluded. We use this information to exclude buildings tagged with the following strings: *"^mall$", "train\\_station", "garages", "hospital", "parking", "sports\\_centre", "university", "gas\\_station", "school", "hall", "government", "prison", "sports\\_hall", carport", "garbage", "waste"*. After taking these filtering steps, we made sure to mostly use residential buildings in the further analysis.

\textit{Adding population values:} To later generate LS index values, it is necessary to add a population count to every residential building. In the first approach, we add the UA population to the OSM building via the respective size of the buildings. In this process, we sum up all building areas in each UA polygon and distribute the population values according to a building’s share of the total building area. This approach follows the assumption that buildings inside a UA polygon – which mostly represents a city block that is delimited by streets or other uses – have homogenous structure and population per area. After the first approach of adding population to the OSM buildings, we need to rule out one further error source. A fraction of the residential polygons in the UA dataset has not been assigned population values. To nonetheless add a population-count to the buildings that are located inside these polygons, we distribute population according to the average population value per building area for each UA residential class. Eventually, we remove all buildings that have not yet received a population value.

\textit{Adding building ID:} To be able to link the index values of the future analysis back to the building polygons, we create in an intermediate step an individual building-ID for each polygon. This is done by adding an incremental number to each building polygon in each UA residential polygon. The filtered building polygons will now be stored to file for later use.

\textit{Calculating building centroids:} Finally, to facilitate the integration of the building data into the network analysis, we transform the polygons into their respective centroids. For this step we decided to use the *st\\_point\\_on\\_surface* function from the sf package, rather than a function taking a geometric centroid. The centroid of all points that make up a geometry can fall outside of the initial geometry. Since we want to use the resulting points to later estimate building entries, a point outside the initial geometry would be undesirable. The thus filtered building centroids will be written to file for use in the later analysis.