Appendix 3 – Index building

%Why index building?

Now that all data necessary to create the Local Significance (LS) and Detour Index (DI) indices is ready to use, we can start calculating the two indices for the entire city.

The indices will be built on the city level during our index building workflow.

%Index building

In the process of index building, we want to calculate the DI for each building and the LS for each street segment.

To facilitate the creation of the two indices, we need to find the green space entry point that is closest to the respective residential building.

For calculating the DI for one building and the nearest green space entry, we need to measure the Euclidean distance and the network distance between the two points.

Consequently, we calculate the average DI for all green spaces that have an entry point located in a network distance of 500 meters of less.

The LS index also requires the network distance.

Additionally, we need the population of the residential building and the size of the park to calculate the LS.

We attach the LS value to each edge (i.e. each street segment) that is crossed on the way from the building entry and the green space entry.

Similar to the DI, we calculate the LS for each green space that can be reached in a network distance of 500 meters or less.

Finally, for each edge we sum up all LS values of all routes from each building to the respective green spaces.

This way we end up with one cumulative LS value for each edge in a city.

%Functionality

We have integrated the process of building the LS and DI indices into the getIndices function of our index building workflow.

The getIndices function takes as input a working directory that contains the previously created databases containing the nodes and edges data, as well as the file with the building polygons.

All paths can be handed manually to the getIndices function or will be created automatically based on the working directory.

By default, the output will be written to the working directory if not specified otherwise.

The index creation workflow follows these steps: a) listing the green space identifiers, b) inputting the data, c) calculating and saving the indices and d) joining the index values back to the network and buildings:

\textit{a) Listing the green space identifiers:} Creating the indices is a computationally intensive process. Particularly the process called routing (i.e. finding the fastest route between two points in a network) can use lots of resources. To increase the speed of completing the LS and DI computation for an entire city and to speed up each computation itself, we use a number of methods. First, we use the UA identifiers that come with the UA data to facilitate parallelization of the process. During the previous steps we have kept these identifiers together with the green space entry points. By iterating through the object identifiers of the green spaces, we can distribute the computation across multiple computational cores. Similar to the snapping and blending process in the previous chapter, our function uses 75% of the available computation cores. Setting this number higher might not be recommended under certain circumstances. The green space identifiers are now used to set up the parallel processing.

\textit{b) Inputting the data:} When we set up parallel processing in R on a Windows machine, we have to account for certain limitations. If run on Windows, each task that we hand to R to compute in parallel will start a new Windows process. Unfortunately, this means that we have to input all of our data separately into each of the processes, which can take up lots of memory. Particularly loading the spatial data into each process has to be planned meticulously to not overwhelm the machines memory and cause (not so) unforeseen crashes. On iterating through the green space identifiers, we first load the green space entries that are associated with the identifier via a SQL query from the nodes database directly into the memory. We then apply a buffer of the 500 meters distance (that we have chosen as a cutoff value) to the green space entries and convert them into a well-known-text (WKT) filter. The WKT filter we use to directly load only those building entries from the nodes dataset into memory that are needed for creating the DI and LS indices in a network distance of 500 meters from the green space entries. Similarly, we use the WKT to load the network data that is located inside the buffered area from the edges database. The data we have obtained for the analysis now is not yet a real representation of the 500 meters network distance, but a severe reduction of memory used. With all data necessary to calculate both, the DI and the LS indices for a single park, we can now proceed with the process.

\textit{c) Calculating and saving the indices:} In the first step, we generate an origin-destination cost-matrix to find the nearest green space entry point for each building entry. Secondly, we compute the Euclidean distance and the shortest paths between each pair of building and green space entry points. From the shortest paths, we can easily add the network distance to each path by summing up the lengths of each edge that the path encounters. At this point, we use the network distance to filter out buildings that are located further away than 500 meters from any of the green space entries. After these steps we have acquired all parameters necessary to calculate both indices: The population is attached to the building nodes as well as the network and the Euclidean distances. The size of the green space is attached to the green space entry nodes. After calculating the DI values for each building, we write the index values with their respective building IDs (see Appendix 2b) to a simple csv-file. Consequently, we write the LS values and their respective edge IDs (see Appendix 2a) to another csv-file. To save disk space, we strip the output of all spatial data. The building entry points were just a surrogate, anyhow, since we want to attach the DI values to their respective building polygons later. When calculating the indices, the computations for all green spaces in a city will be running in parallel on as many cores as we provide. Accordingly, we end up with two csv-files for each green space: one containing the building IDs and the DI values and one containing the edge IDs with the LS values.

\textit{d) Joining the index values back to the network and buildings:} The output that we have generated so far is void of any spatial information. For most interpretation of the data we have produced (i.e. for making beautiful maps), we have to join the output back to its spatial counterparts. Additionally, we have no information on the average DI at the building level or the cumulative LS at a street level. Accordingly, we first list and read all csv files that contain the DI values. The building IDs and DI values are gathered in a data frame from which we can calculate the average DI per building ID. Afterwards, we load the building polygons and join the DI values to them according to the building ID. Similarly, we gather all LS values from the csv files and sum them up via the individual edge IDs. After loading the edge database, we can now join the LS values to the network data. Finally, we can write both outputs to their geopackage databases and use them for plotting or further calculations.

After these steps, both indices have been calculated. We did not implement a step for deleting the intermediate output values of this processing workflow. Thus, enabling further analysis with the information on green space level: For each green space there are two csv files stored that is named after the UA identifier, containing the building and the edge IDs. This information can be further utilized to e.g. count the number of buildings (or if joined to the building polygons: population) in the service area of a particular green space.