BikeDNA Report

Belgrade

Bicycle Infrastructure
Data & Network Assessment

Report generated on 2023-06-25 15:38:20
with BikeDNA v.1.0.2 · github.com/anerv/BikeDNA

About BikeDNA

This report was automatically generated by BikeDNA (Bicycle Data & Network Assessment) by converting Jupyter notebooks into pdf. BikeDNA is a tool for assessing the quality of OpenStreetMap (OSM) and other bicycle infrastructure data sets in a reproducible way. It provides planners, researchers, data maintainers, cycling advocates, and others who work with bicycle networks a detailed, informed overview of data quality in a given area.

BikeDNA is maintained at https://github.com/anerv/BikeDNA by Ane Rahbek Vierø, Anastassia Vybornova, and Michael Szell. It is available under the AGPL 3.0 license.

A fair amount of research projects on OpenStreetMap and other forms of volunteered geographic information (VGI) have already been conducted, but few focus explicitly on bicycle infrastructure. Doing so is important because paths and tracks for cyclists and pedestrians often are mapped last and are more likely to have errors (Barron et al., 2014, Neis et al. 2012). Moreover, the spatial distribution of dips in data quality in crowdsourced data are often not random but correlate with population density and other characteristics of the mapped area (Forghani and Delavar, 2014), which requires a critical stance towards the data we use for our research and planning, despite the overall high quality of OSM.

Data quality covers a wide range of aspects. The conceptualization of data quality used here refers to *fitness-for-purpose* (Barron et al., 2014) - this means that data quality is interpreted as whether or not the data fulfils the user needs, rather than any universal definition of quality. To particularly support network-based research and planning, BikeDNA provides insights into the topological structure of the bicycle network apart from data coverage.

The purpose is not to give any final assessment of the data quality, but to highlight aspects that might be relevant for assessing whether the data for a given area is fit for use. While BikeDNA can make use of a reference dataset to compare with OSM, if one is available, BikeDNA cannot give any final assessment of the quality of a refernece data compared to OSM. However, OSM data on bicycle infrastructure is often at a comparable or higher quality than governmental datasets, and the interpretation of differences between the two requires adequate local knowledge.

1a. Initialize OSM data

This notebook:

- Loads the polygon defining the study area and then creates a grid overlay for the study area.
- Downloads street network data for the study area using OSMnx.
- Creates a network only with bicycle infrastructure (with queries defined in `config.yml`).
- Creates additional attributes in the data to be used in the analysis.

Sections

Load data for study area and create analysis grid

This step:

- Loads settings for the analysis from the configuration file `config.yml`.
- Reads data for the study area.
- Creates a grid overlay of the study area, with grid cell size as defined in `config.yml`.

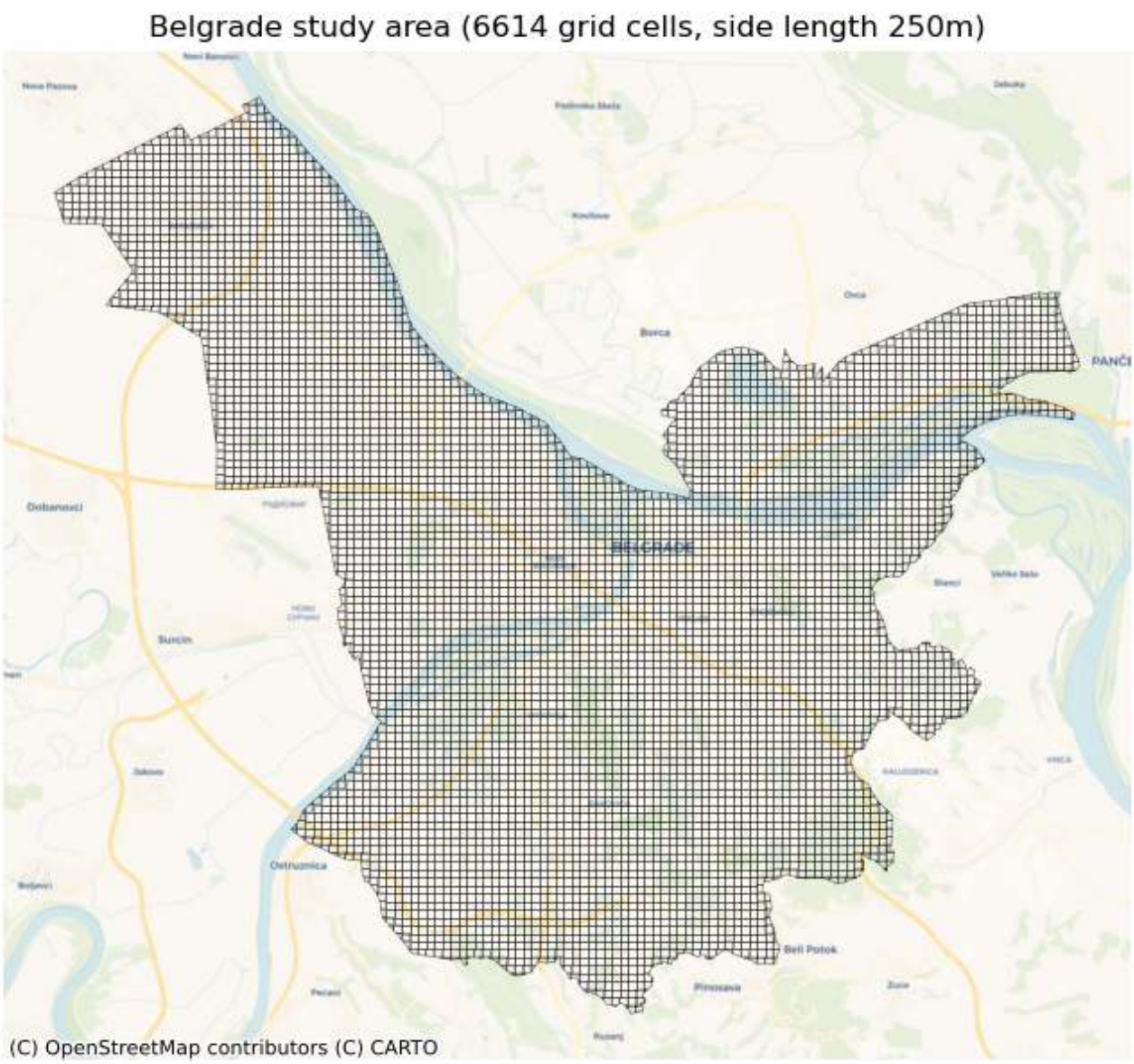
Load data for study area

The study area is defined by the user-provided polygon. It will be used for the computation of **global** results, i.e. quality metrics based on all data in the study area.

```
The size of the study area is 388.59 km2.
```

Create analysis grid

```
The grid contains 6614 square cells with a side length of 250 m and an area of 0.06 km2.
This grid will be used for local (grid cell level) analysis:
```



Download and preprocess OSM data

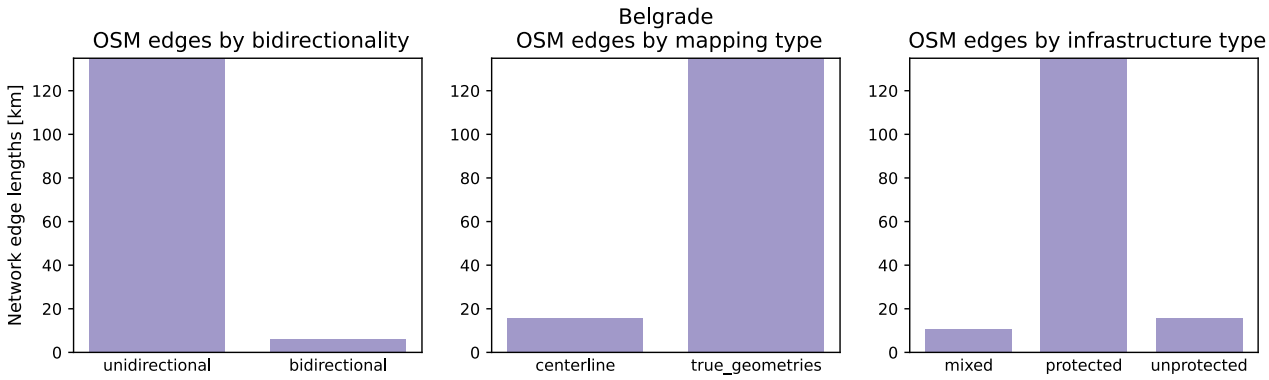
This step:

- Downloads data from OpenStreetMap using OSMnx.
- Projects the data to the chosen CRS.
- Creates a subnetwork consisting only of bicycle infrastructure.
- Classifies all edges in the bicycle network based on whether they are protected or unprotected bicycle infrastructure, how they have been digitized, and whether they allow for bidirectional travel or not.
- Simplifies the network.
- Creates copies of all edge and node data sets indexed by their intersecting grid cell.

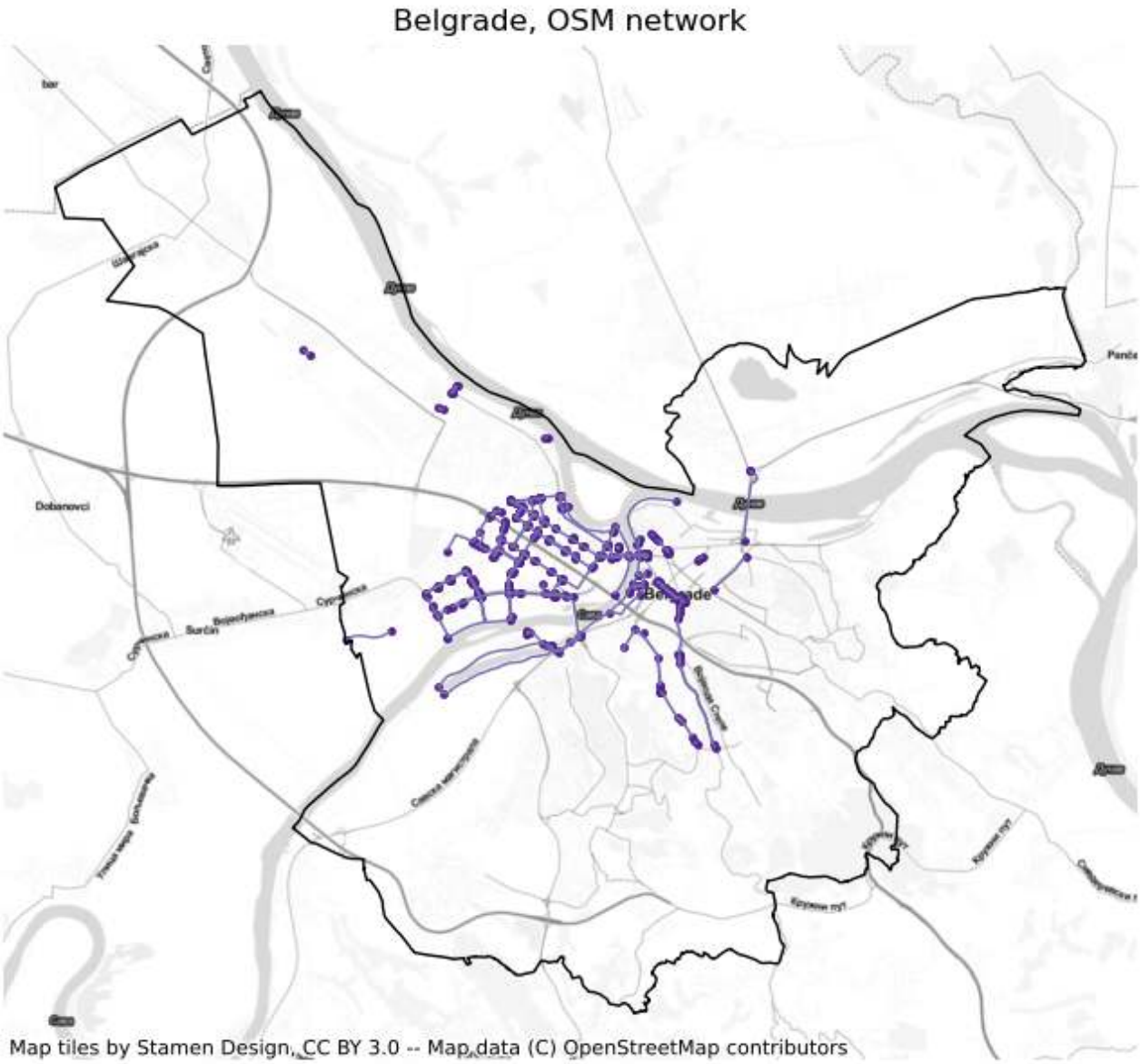
OSM data model

In OSM, street network data are stored using *nodes* (points) and *ways* (lines). In BikeDNA, OSM data are converted to a network structure consisting of *nodes* and *edges* (we use the terminology used in OSMnx). Edges represents the actual infrastructure, such as bike lanes and paths, while nodes represents the start and end points for the edges, as well as all intersections. For further details, read more about the OSM data model and the network data model.

```
No update necessary. Returning original bicycle graph.
Edges where 'bicycle_bidirectional' is False: 8255 out of 8479 (97.36%)
Edges where 'bicycle_bidirectional' is True: 224 out of 8479 (2.64%)


Edges where the geometry type is 'true_geometries': 7782 out of 8479 (91.78%)
Edges where the geometry type is 'centerline': 697 out of 8479 (8.22%)


Edges where the protection level is 'protected': 6581 out of 8479 (77.62%)
Edges where the protection level is 'mixed': 1201 out of 8479 (14.16%)
Edges where the protection level is 'unprotected': 697 out of 8479 (8.22%)
```




```
The network covers an area of 110.76 km2.
The length of the OSM network with bicycle infrastructure is 112.41 km.
```


Belgrade, OSM network
Dobanovci
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

1b. Intrinsic Analysis of OSM Data

This notebook analyzes the quality of OSM bicycle infrastructure data for a given area. The quality assessment is *intrinsic*, i.e. based only on the one input data set without makeing use of external information. For an extrinsic quality assessment that compares the OSM data to a user-provided reference data set, see the notebooks 3a and 3b.

The analysis assesses the *fitness for purpose* (Barron et al., 2014) of OSM data for a given area. Outcomes of the analysis can be relevant for bicycle planning and research - especially for projects that include a network analysis of bicycle infrastructure, in which case the topology of the geometries is of particular importance.

Since the assessment does not make use of an external reference data set as the ground truth, no universal claims of data quality can be made. The idea is rather to enable those working with OSM-based bicycle networks to assess whether the data are good enough for their particular use case. The analysis assists in finding potential data quality issues but leaves the final interpretation of the results to the user.

The notebook makes use of quality metrics from a range of previous projects investigating OSM/VGI data quality, such as Ferster et al. (2020), Hochmair et al. (2015), Barron et al. (2014), and Neis et al. (2012).

Familiarity required
For a correct interpretation of some of the metrics for spatial data quality, some familiarity with the area is necessary.

Sections

Data completeness

Network density

In this setting, network density refers to the length of edges or number of nodes per km2. This is the usual definition of network density in spatial (road) networks, which is distinct from the *structural* network density known more generally in network science. Without comparing to a reference data set, network density does not in itself indicate spatial data quality. For anyone familiar with the study area, network density can however indicate whether parts of the area appear to be under- or over-mapped.

Method

The density here is not based on the geometric length of edges, but instead on the computed length of the infrastructure. For example, a 100-meter-long bidirectional path contributes with 200 meters of bicycle infrastructure. This method is used to take into account different ways of mapping bicycle infrastructure, which otherwise can introduce large deviations in network density. With `compute_network_density`, the number of elements (nodes, dangling nodes, and total infrastructure length) per unit area is calculated. The density is computed twice: first for the study area for both the entire network ('global density'), then for each of the grid cells ('local density'). Both global and local densities are computed for the entire network and for protected and unprotected infrastructure.

Interpretation

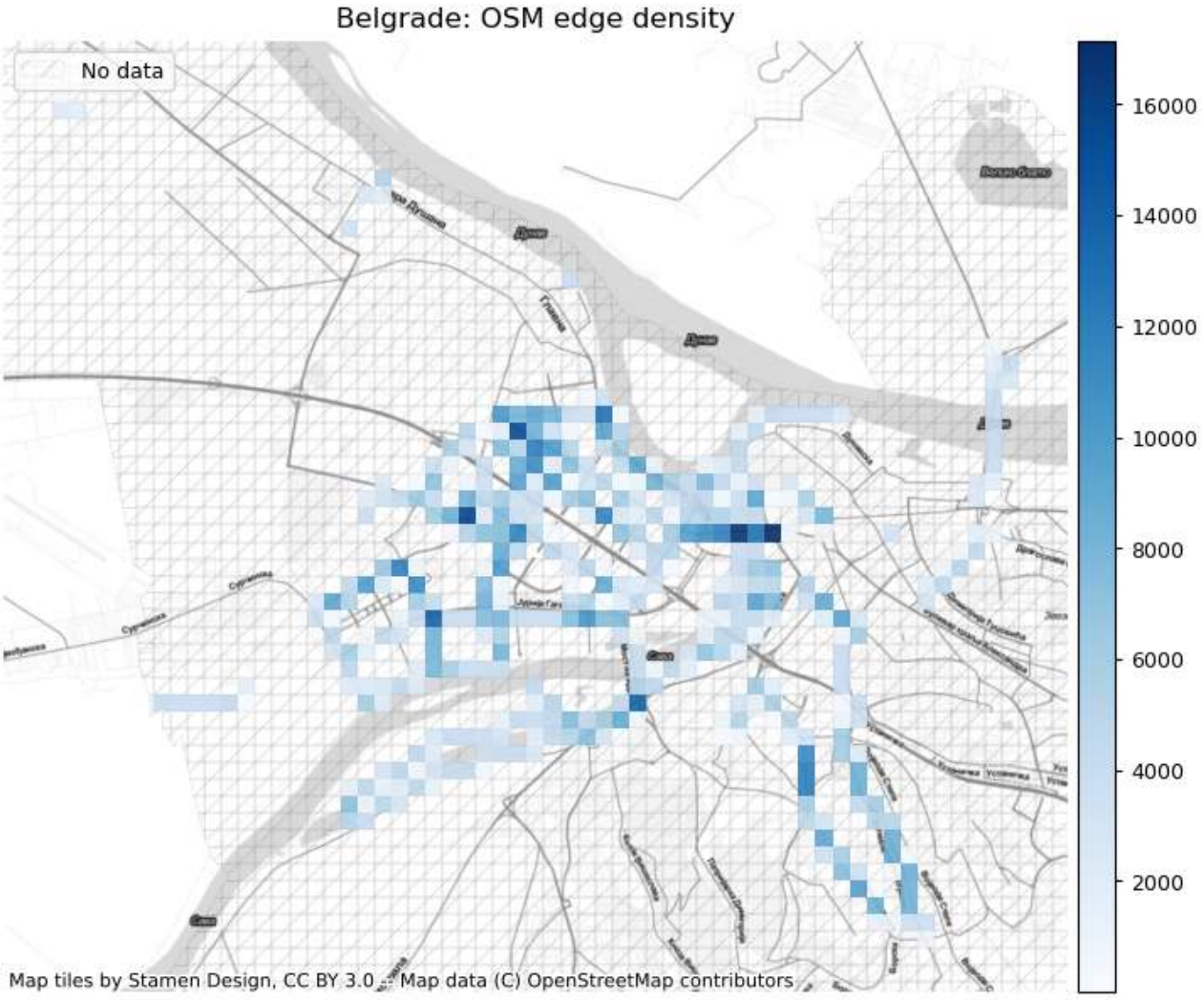
Since the analysis conducted here is intrinsic, i.e. it makes no use of external information, it cannot be known whether a low-density value is due to incomplete mapping, or due to actual lack of infrastructure in the area. However, a comparison of the grid cell density values can provide some insights, for example:

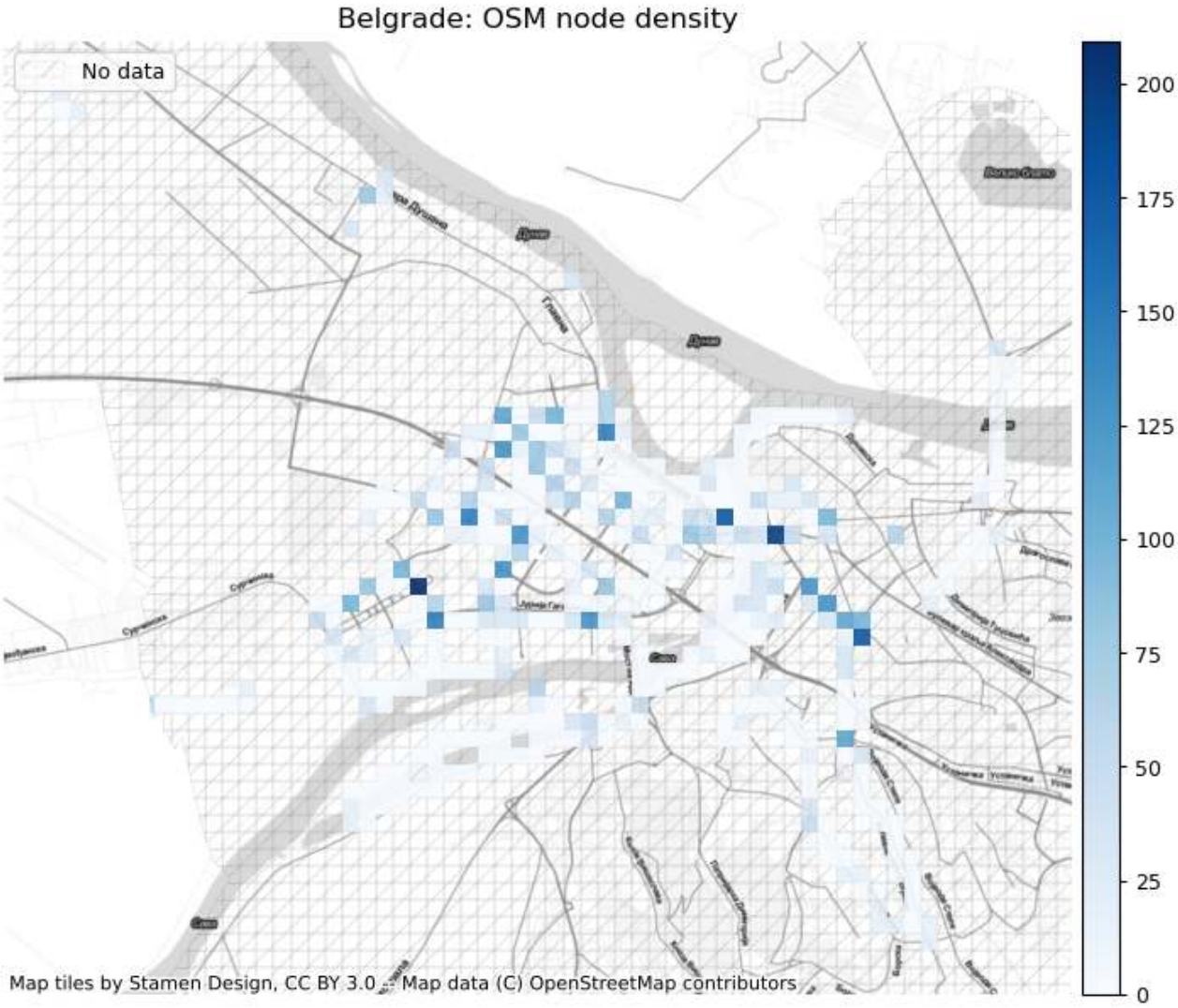
- lower-than-average infrastructure density indicates a locally sparser network
- higher-than-average node density indicates that there are relatively many intersections in a grid cell
- higher-than-average dangling node density indicates that there are relatively many dead ends in a grid cell

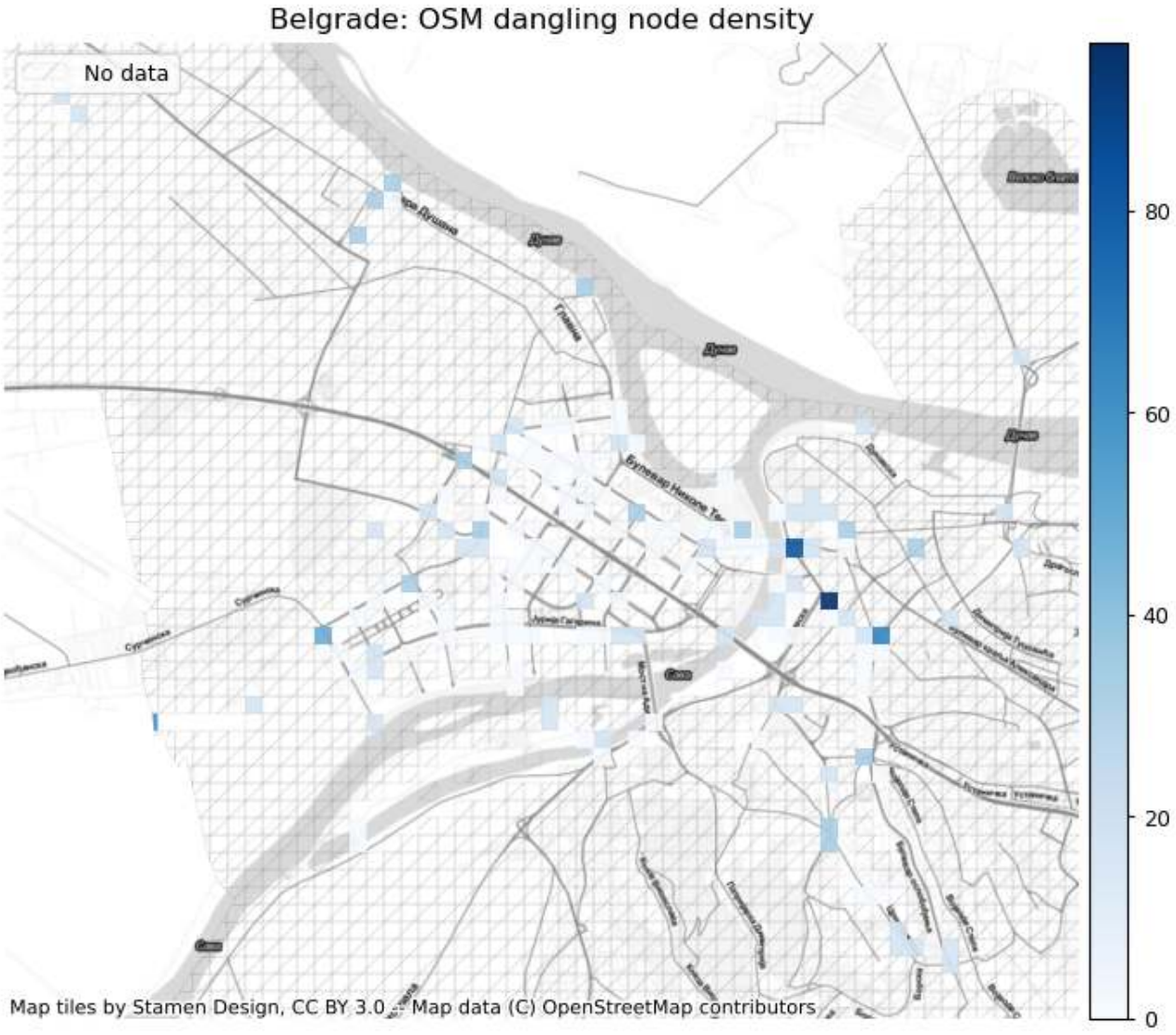
Global network density

```
For the entire study area, there are:
- 289.28 meters of bicycle infrastructure per km2.
- 1.16 nodes in the bicycle network per km2.
- 0.24 dangling nodes in the bicycle network per km2.
- 236.44 meters of protected bicycle infrastructure per km2.
- 38.29 meters of unprotected bicycle infrastructure per km2.
- 14.55 meters of mixed protection bicycle infrastructure per km2.
```

Local network density


Belgrade: OSM edge density
No data
16000
14000
12000
10000
8000
6000
4000
2000
Map tiles by Stamen Design, CC BY 3.0 — Map data (C) OpenStreetMap contributors

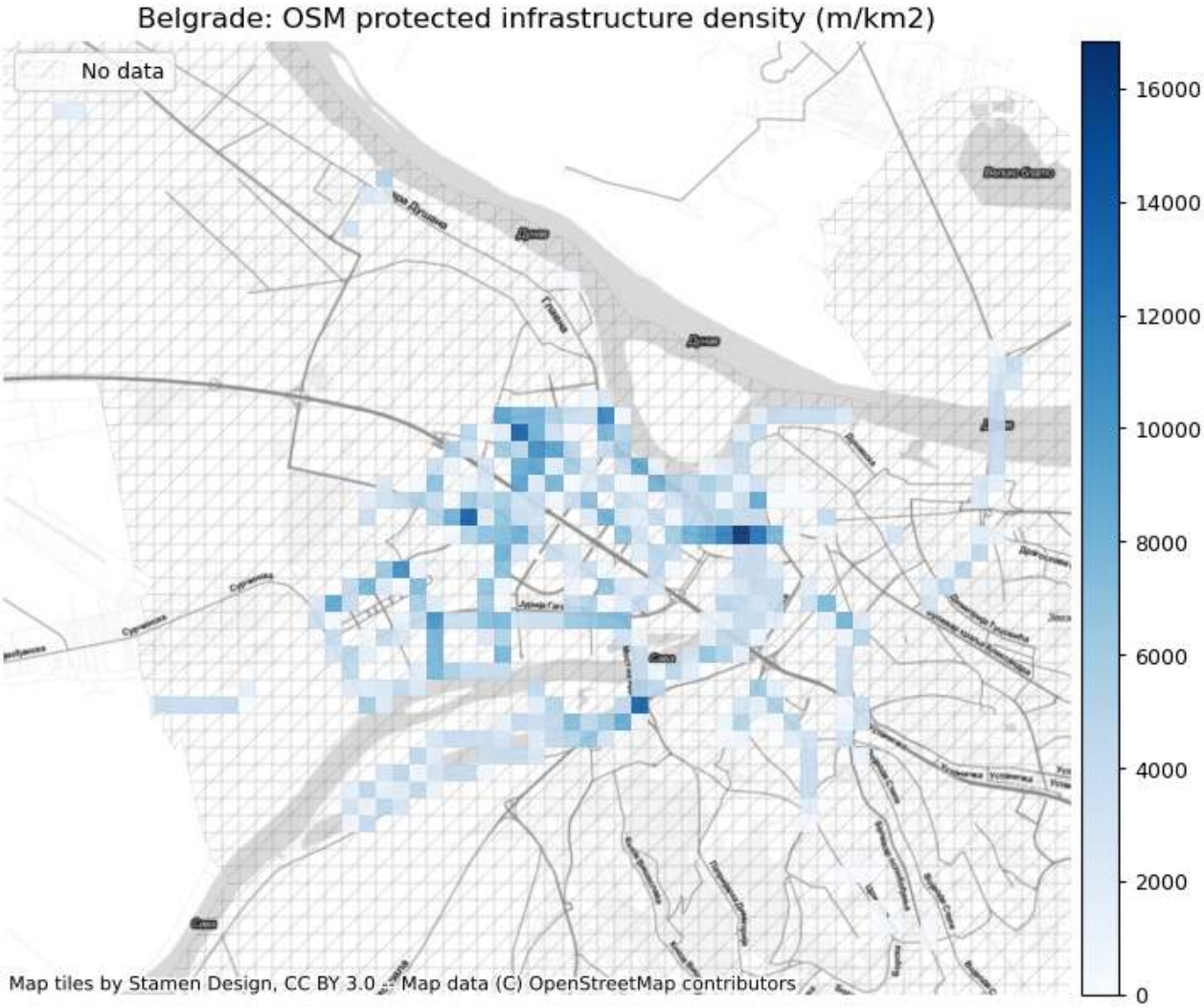

Belgrade: OSM node density
No data
200
175
150
125
100
75
50
25
0
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

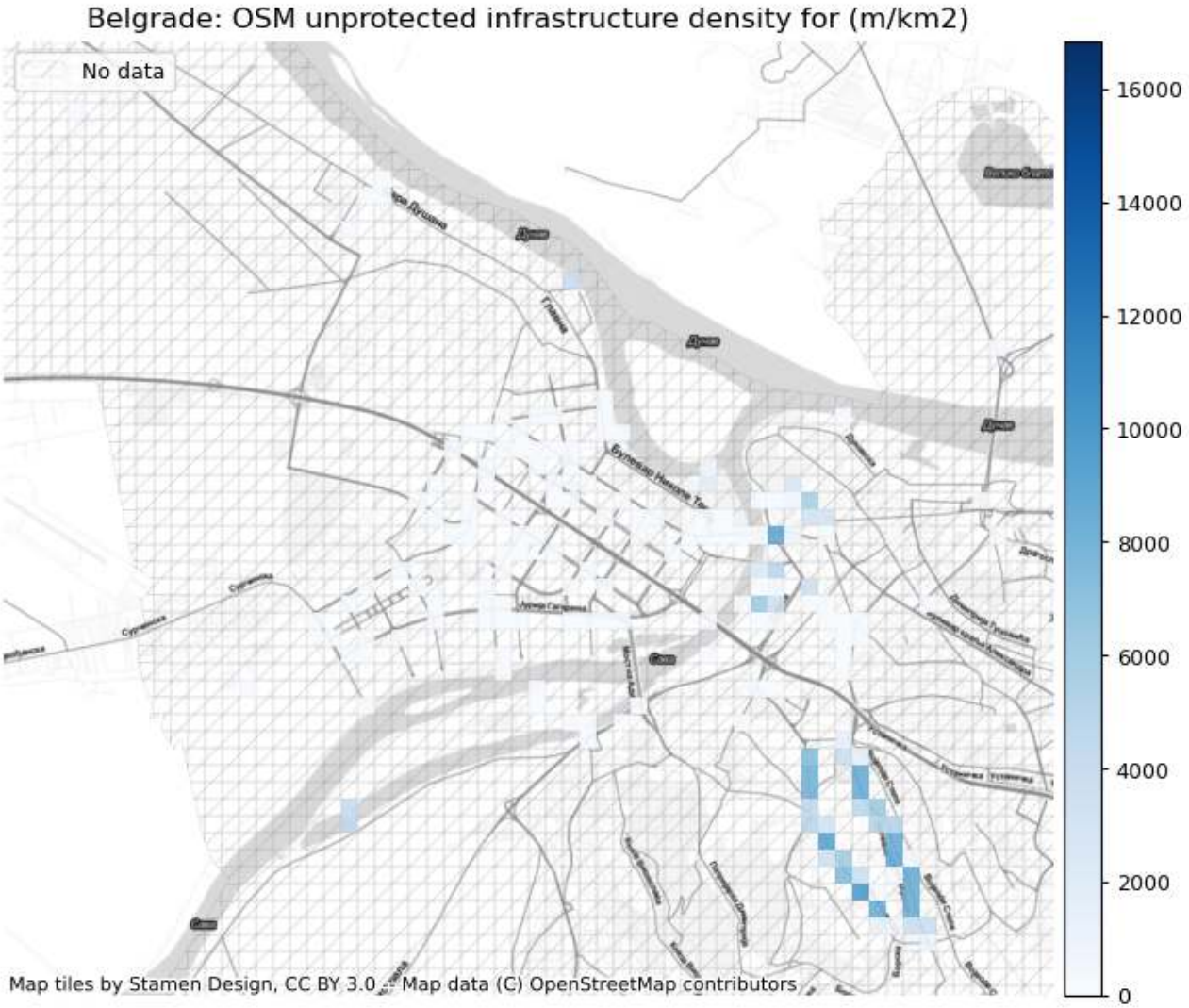


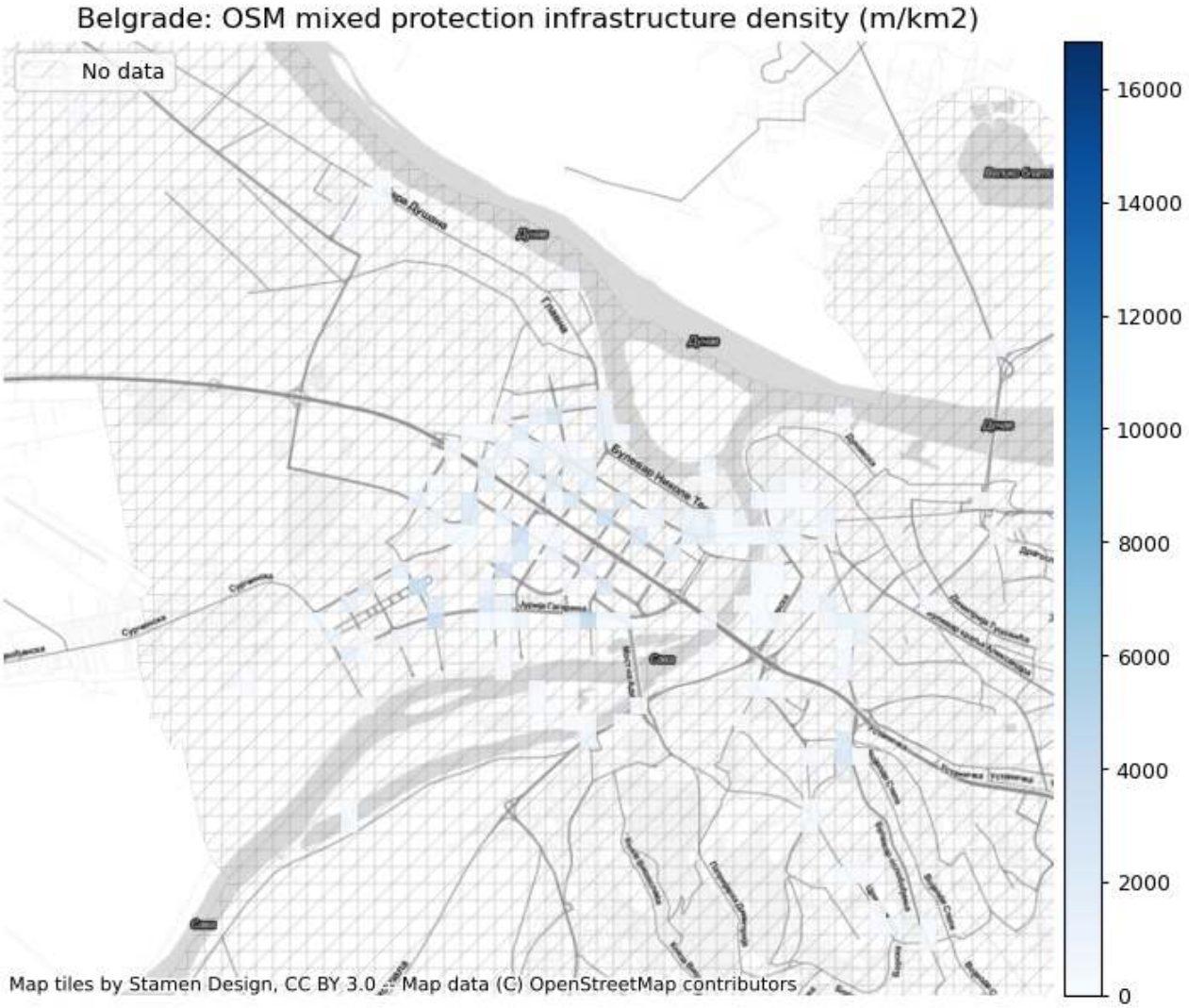
Densities of protected and unprotected infrastructure

In BikeDNA, *protected infrastructure* refers to all bicycle infrastructure which is either separated from car traffic by for example an elevated curb, bollards, or other physical barriers, or for cycle tracks that are not adjacent to a street.

Unprotected infrastructure are all other types of lanes that are dedicated for bicyclists, but which only are separated by car traffic by e.g., a painted line on the street.


Belgrade: OSM protected infrastructure density (m/km2)
No data
16000
14000
12000
10000
8000
6000
4000
2000
0
Map tiles by Stamen Design, CC BY 3.0 — Map data (C) OpenStreetMap contributors


Belgrade: OSM unprotected infrastructure density for (m/km2)
No data
16000
14000
12000
10000
8000
6000
4000
2000
0
Map tiles by Stamen Design, CC BY 3.0 — Map data (C) OpenStreetMap contributors



OSM tag analysis

For many practical and research purposes, more information than just the presence/absence of bicycle infrastructure is of interest. Information about e.g. the width of the infrastructure, speed limits, streetlights, etc. can be of high relevance, for example when evaluating the bike friendliness of an area or an individual network segment. The presence of these tags describing attributes of the bicycle infrastructure is however highly unevenly distributed in OSM, which poses a barrier to evaluations of bikeability and traffic stress. Likewise, the lack of restrictions on how OSM features can be tagged sometimes result in conflicting tags which can undermine the evaluation of cycling conditions.

This section includes analyzes of missing tags (edges with tags that lack information), incompatible tags (edges with tags labelled with two or more contradictory tags), and tagging patterns (the spatial variation of which tags are being used to describe bicycle infrastructure).

For the evaluation of tags, the non-simplified edges should be used to avoid issues with tags that have been aggregated in the simplification process.

Missing tags

The information that is required or desirable to obtain from the OSM tags depends on the use case - for example, the tag `lit` for a project that studies light conditions on cycle paths. The workflow below allows to quickly analyze the percentage of network edges that have a value available for the tag of interest.

Method

We analyze all tags of interest as defined in the `existing_tag_analysis` section of `config.yml`. For each of these tags, `analyze_existing_tags` is used to compute the total number and the percentage of edges that have a corresponding tag value.

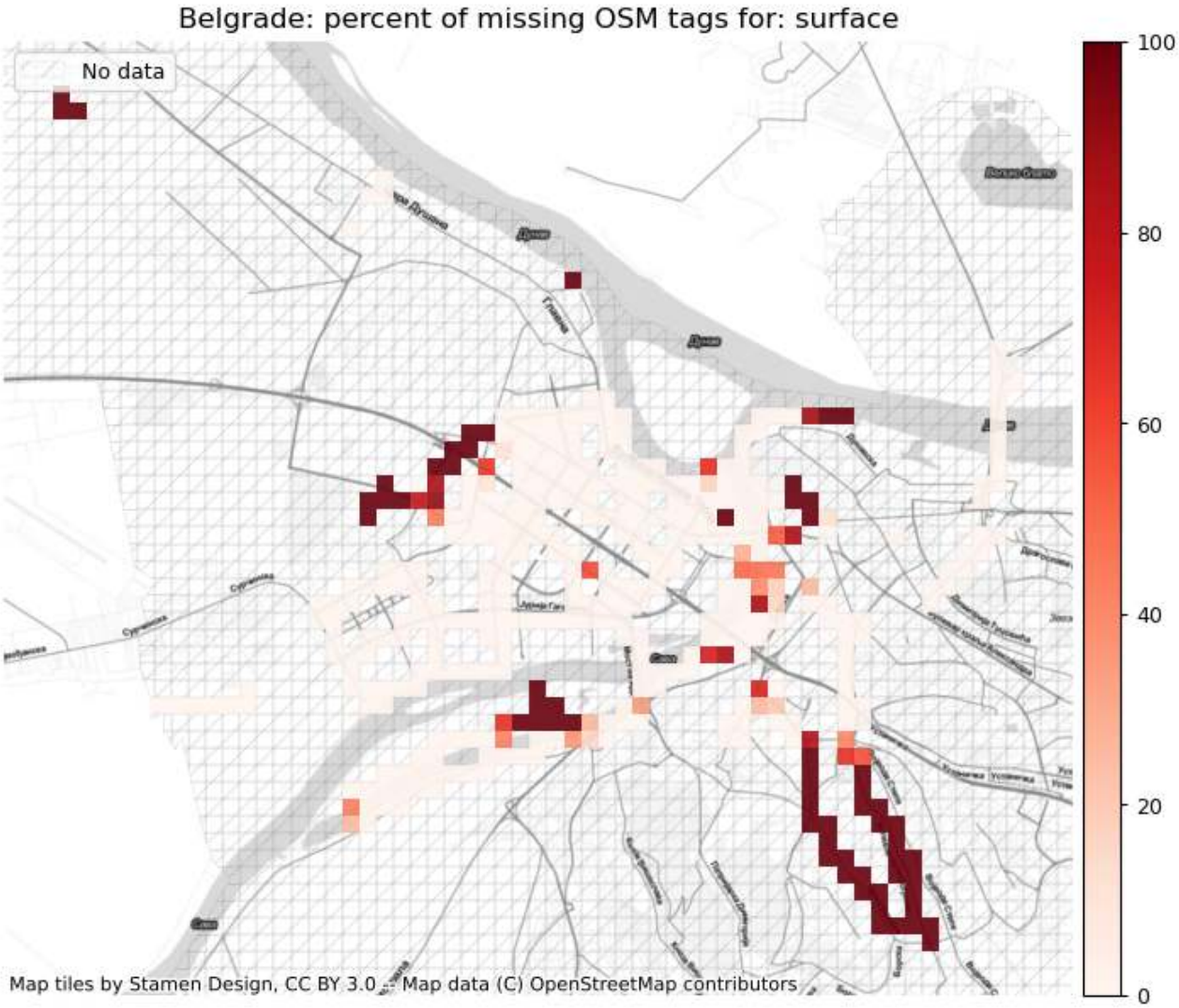
Interpretation

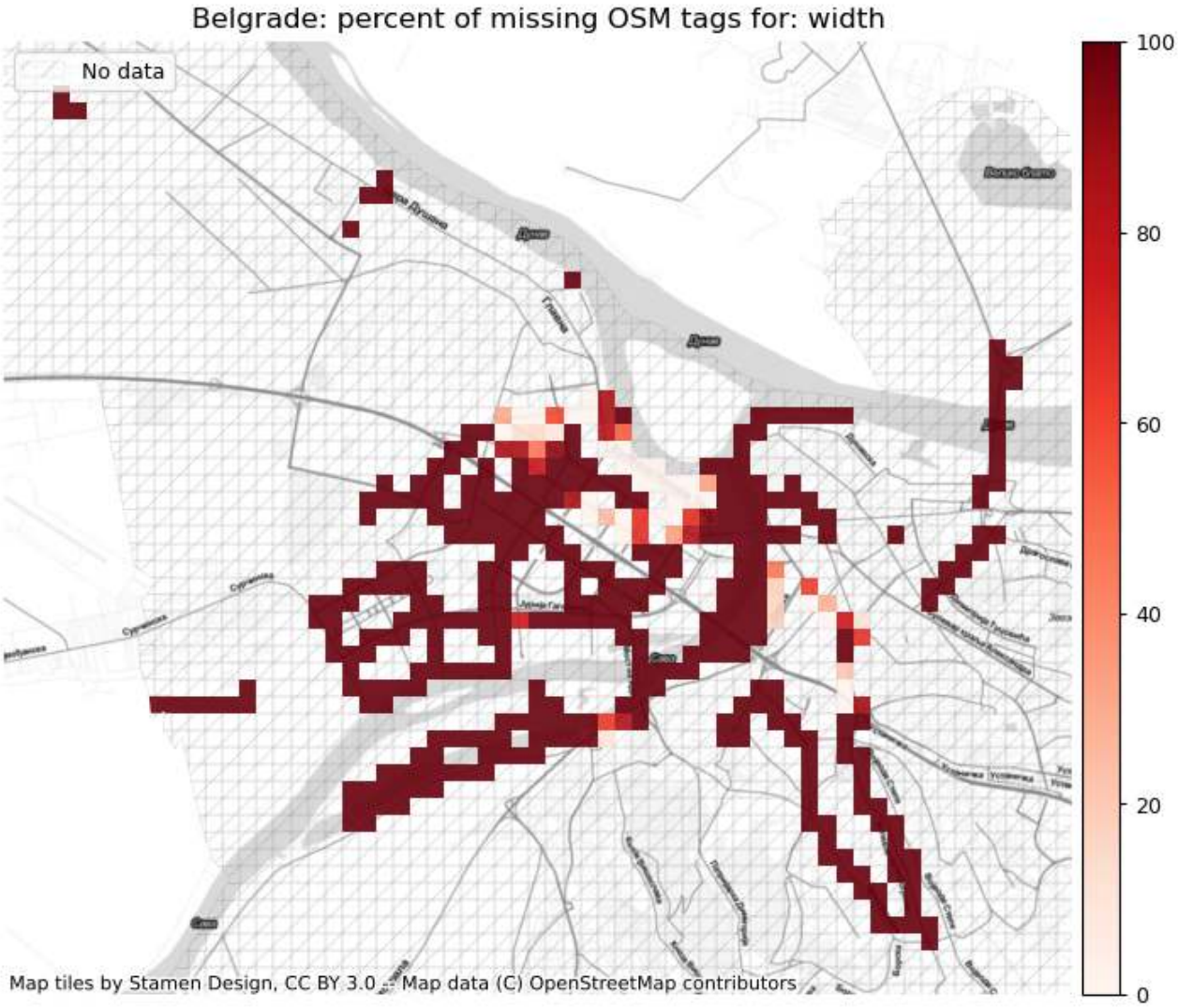
On the study area level, a higher percentage of existing tag values indicates in principle a higher quality of the data set. However, this is different from an estimation of whether the existing tag values are truthful. On the grid cell level, lower-than-average percentages for existing tag values can indicate a more poorly mapped area. However, the percentages are less informative for grid cells with a low number of edges: for example, if a cell contains one single edge that has a tag value for `lit`, the percentage of existing tag values is 100% - but given that there is only 1 data point, this is less informative than, say, a value of 80% for a cell that contains 200 edges.

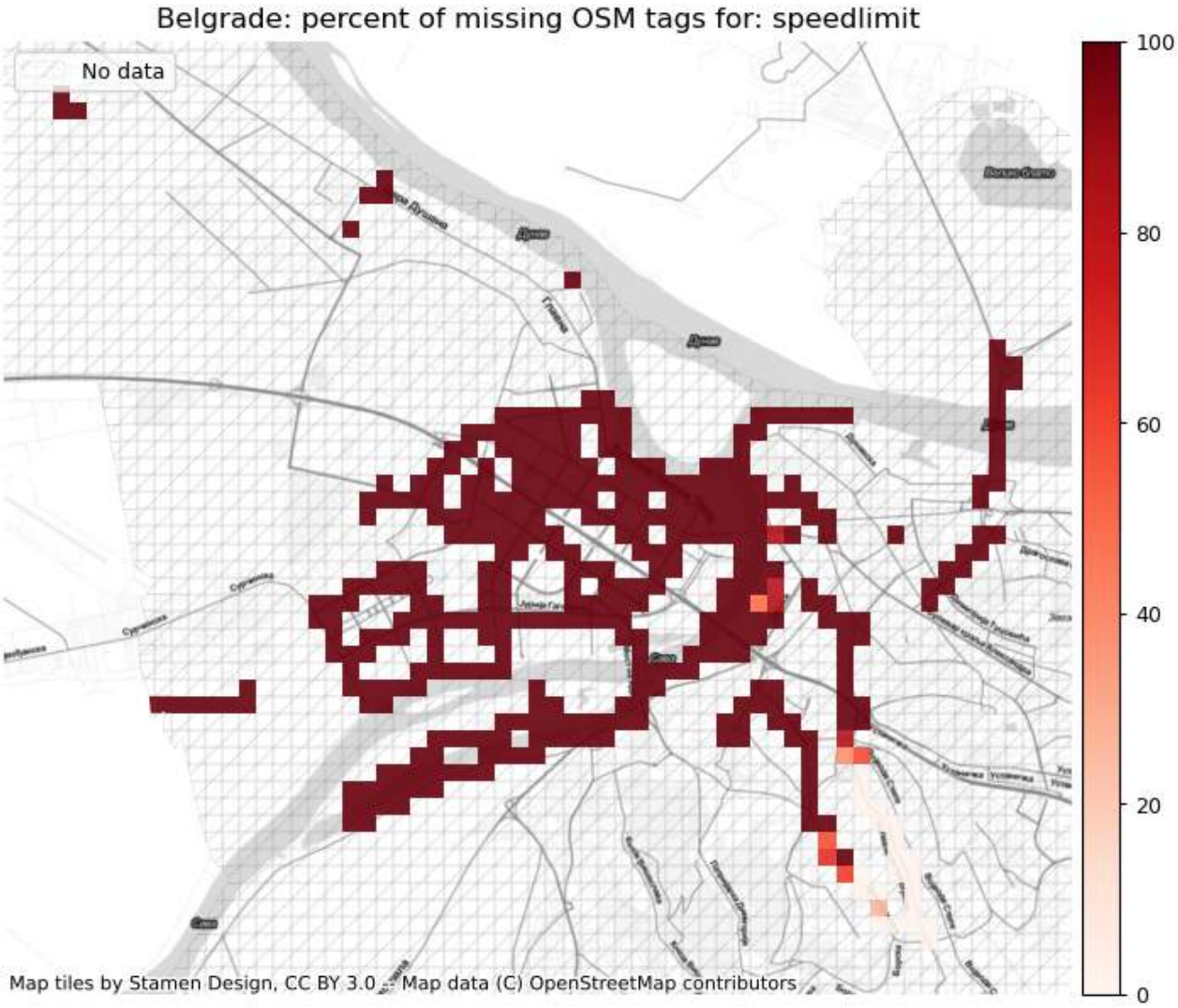
Global missing tags

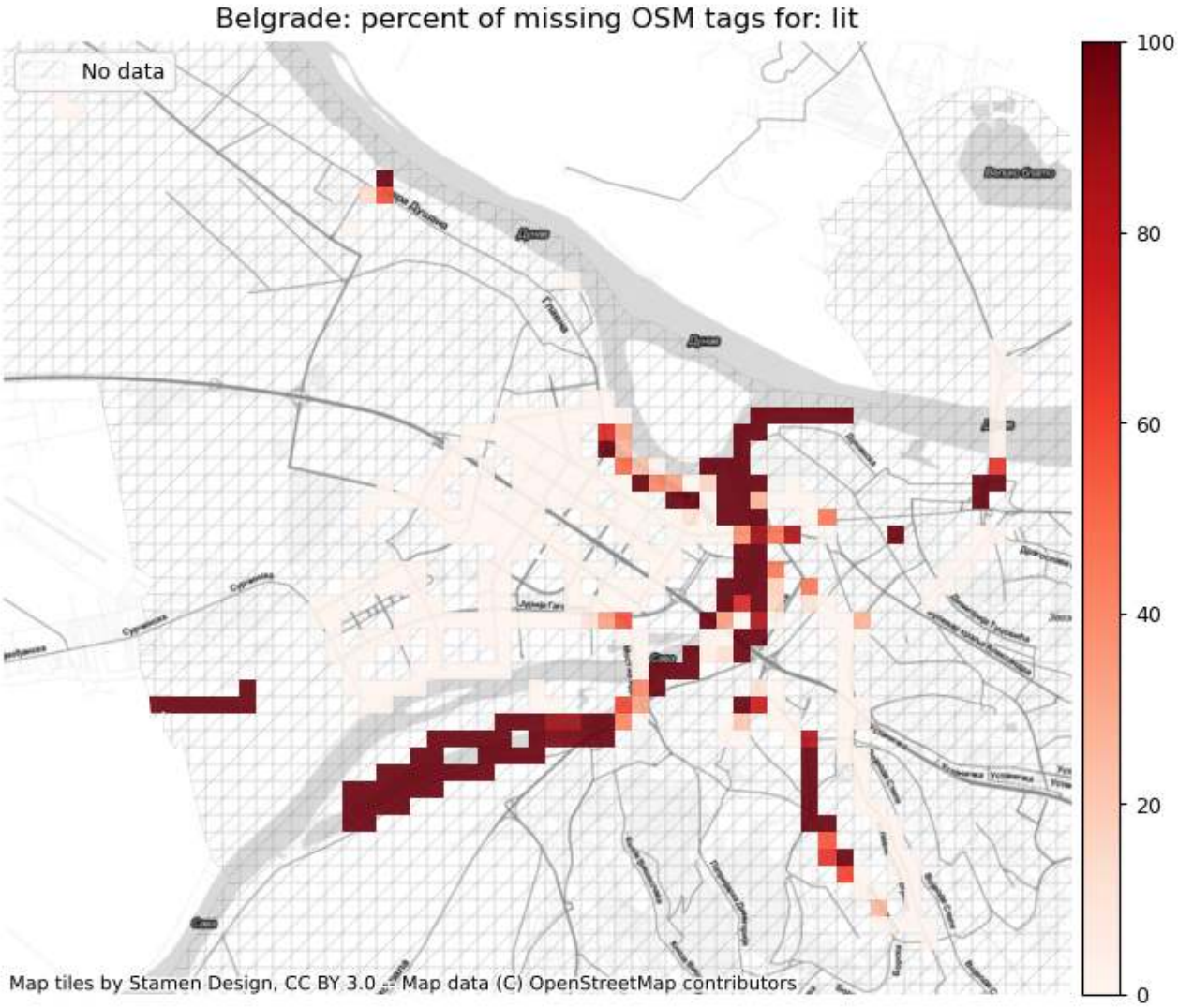
```
Analysing tags describing:
surface - width - speedlimit - lit -

surface: 6644 out of 8479 edges (78.36%) have information.
surface: 151 out of 199 km (75.62%) have information.


width: 869 out of 8479 edges (10.25%) have information.
width: 10 out of 199 km (5.01%) have information.


speedlimit: 383 out of 8479 edges (4.52%) have information.
speedlimit: 9 out of 199 km (4.40%) have information.


lit: 5482 out of 8479 edges (64.65%) have information.
lit: 142 out of 199 km (71.33%) have information.
```

Local missing tags


Belgrade: percent of missing OSM tags for: surface
No data
100
80
60
40
20
0
Map tiles by Stamen Design, CC BY 3.0 — Map data (C) OpenStreetMap contributors


Belgrade: percent of missing OSM tags for: width
No data
100
80
60
40
20
0
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors


Belgrade: percent of missing OSM tags for: speedlimit
No data
100
80
60
40
20
0
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors



Incompatible tags

Given that the tags in OSM data lack coherency at times and there are no restrictions in the tagging process (cf. Barron et al., 2014), incompatible tags might be present in the data set. For example, an edge might be tagged with the following two contradicting key-value pairs: `bicycle_infrastructure = yes` and `bicycle = no`.

Method

In the `config.yml` file, a list of incompatible key-value pairs for tags in the `incompatible_tags_analysis` is defined. Since there is no limitation to which tags a data set could potentially contain, the list is, by definition, non-exhaustive, and can be adjusted by the user. In the section below, `check_incompatible_tags` is run, which identifies all incompatibility instances for a given area, first on the study area level and then on the grid cell level.

Interpretation

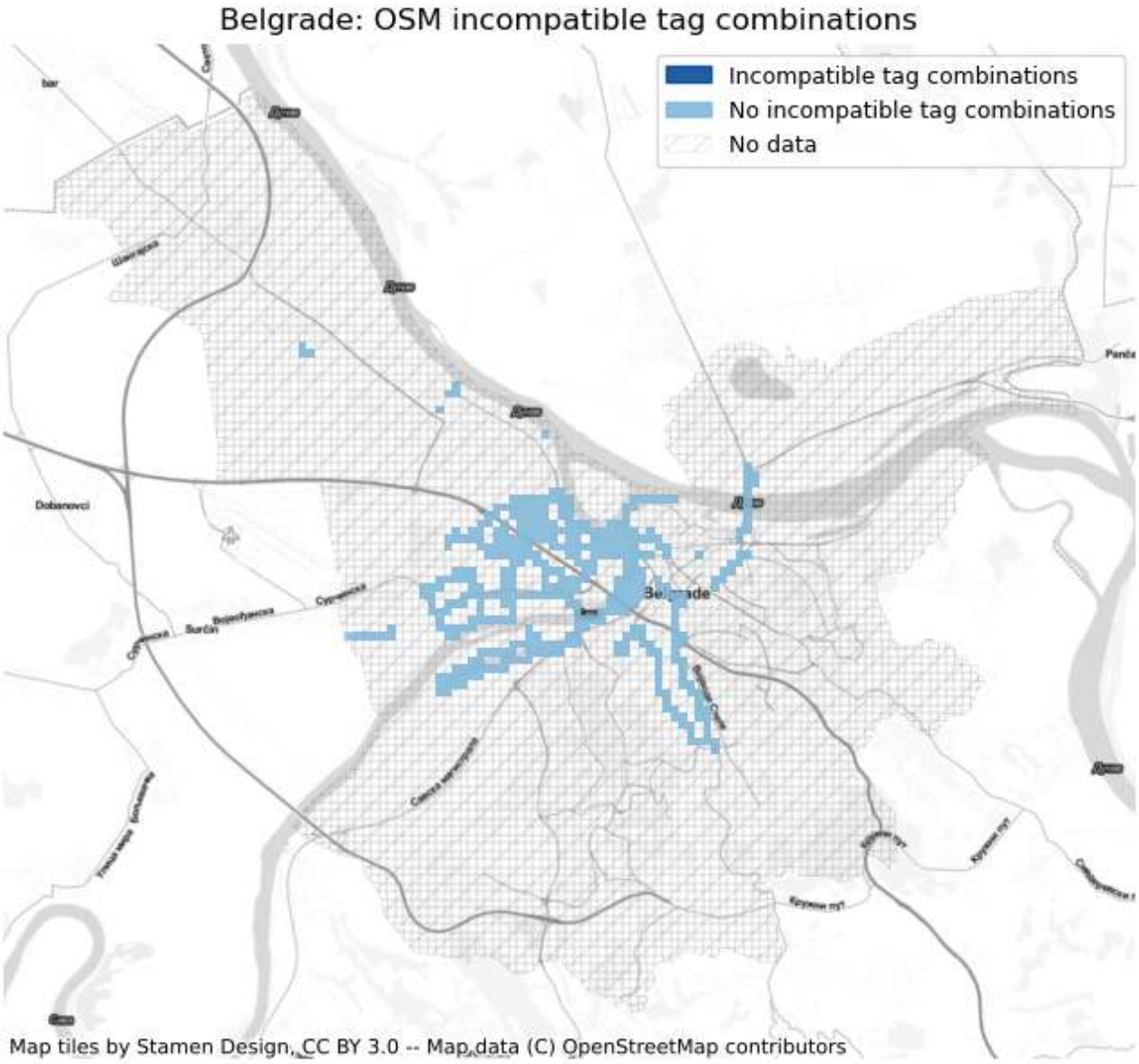
Incompatible tags are an undesired feature of the data set and render the corresponding data points invalid; there is no straightforward way to resolve the arising issues automatically, making it necessary to

either correct the tag manually or to exclude the data point from the data set. A higher-than-average number of incompatible tags in a grid cell suggests local mapping issues.

Global incompatible tags (total number)

```
In the entire data set, there are 0 incompatible tag combinations (of those defined in the confi
guration file).
```

Local incompatible tags (per grid cell)



Plotting incompatible tag geometries

```
Interactive map saved at results/OSM/Belgrade/maps_interactive/tagsincompatible_osm.html
```

Tagging patterns

Identifying bicycle infrastructure in OSM can be tricky due to the many different ways in which the presence of bicycle infrastructure can be indicated. The OSM Wiki is a great resource for recommendations for how OSM features should be tagged, but some inconsistencies and local variations can remain. The analysis of tagging patterns allows to visually explore some of the potential inconsistencies.

Regardless of how the bicycle infrastructure is defined, examining which tags contribute to which parts of the bicycle network allows to visually examine patterns in tagging methods. It also allows to estimate whether some elements of the query will lead to the inclusion of too many or too few features.

Likewise, 'double tagging' where several different tags have been used to indicate bicycle infrastructure can lead to misclassifications of the data. For this reason, identifying features that are included in more than one of the queries defining bicycle infrastructure can indicate issues with the tagging quality.

Method

We first plot individual subsets of the OSM data set for each of the queries listed in `bicycle_infrastructure_queries`, as defined in the `config.yml` file. The subset defined by a query is the set of edges for which this query is *True*. Since several queries can be *True* for the same edge, the subsets can overlap. In the second step below, all overlaps between 2 or more queries are plotted, i.e. all edges that have been assigned several, potentially competing, tags.

Interpretation

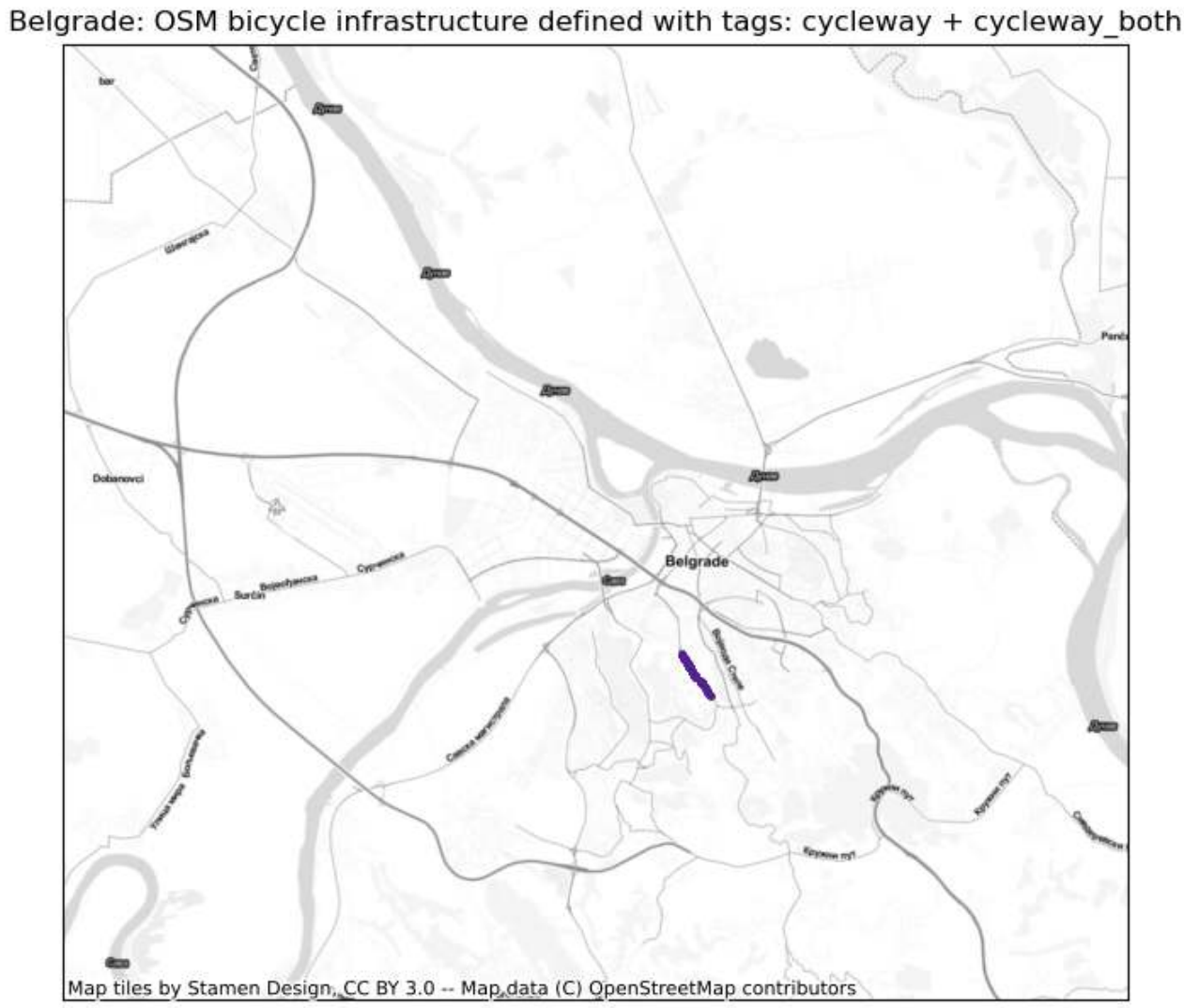
The plots for each tagging type allow for a quick visual overview of different tagging patterns present in the area. Based on local knowledge, the user may estimate whether the differences in tagging types are due to actual physical differences in the infrastructure or rather an artefact of the OSM data. Next, the user can access overlaps between different tags; depending on the specific tags, this may or may not be a data quality issue. For example, in case of `'cycleway:right'` and `'cycleway:left'`, having data for both tags is valid, but other combinations such as `'cycleway'='track'` and `'cycleway:left=lane'` gives an ambiguouos picture of what type of bicycle infrastructure is present.

Tagging types

```
Interactive map saved at results/OSM/Belgrade/maps_interactive/taggingtypes_osm.html
```

Multiple tagging

Belgrade: OSM bicycle infrastructure defined with tags: highway + cycleway




```
Interactive map saved at results/OSM/Belgrade/maps_interactive/taggingcombinations_osm.html
```

Network topology

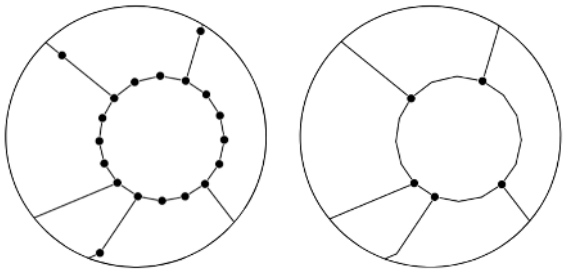
This section explores the geometric and topological features of the data. These are, for example, network density, disconnected components, and dangling (degree one) nodes. It also includes exploring whether there are nodes that are very close to each other but do not share an edge - a potential sign of edge undershoots - or if there are intersecting edges without a node at the intersection, which might indicate a digitizing error that will distort routing on the network.

Due to the fragmented nature of most bicycle networks, many metrics, such as missing links or network gaps, can simply reflect the true extent of the infrastructure (Natera Orozco et al., 2020). This is different for road networks, where e.g., disconnected components could more readily be interpreted as a data quality issue. Therefore, the analysis only takes very small network gaps into account as potential data quality issues.

Simplification outcome

To compare the structure and true ratio between nodes and edges in the network, a simplified network representation which only includes nodes at endpoints and intersections was created in notebook `1a` by removing all interstitial nodes.

Comparing the degree distribution for the networks before and after simplification is a quick sanity check for the simplification routine. Typically, the vast majority of nodes in the non-simplified network will be of degree two; in the simplified network, however, most nodes will have degrees other than two. Degree two nodes are retained in only two cases: if they represent a connection point between two different types of infrastructure; or if they are needed in order to avoid self-loops (edges whose start and end points are identical) or multiple edges between the same pair of nodes.



Non-simplified network (left) and simplified network (right).

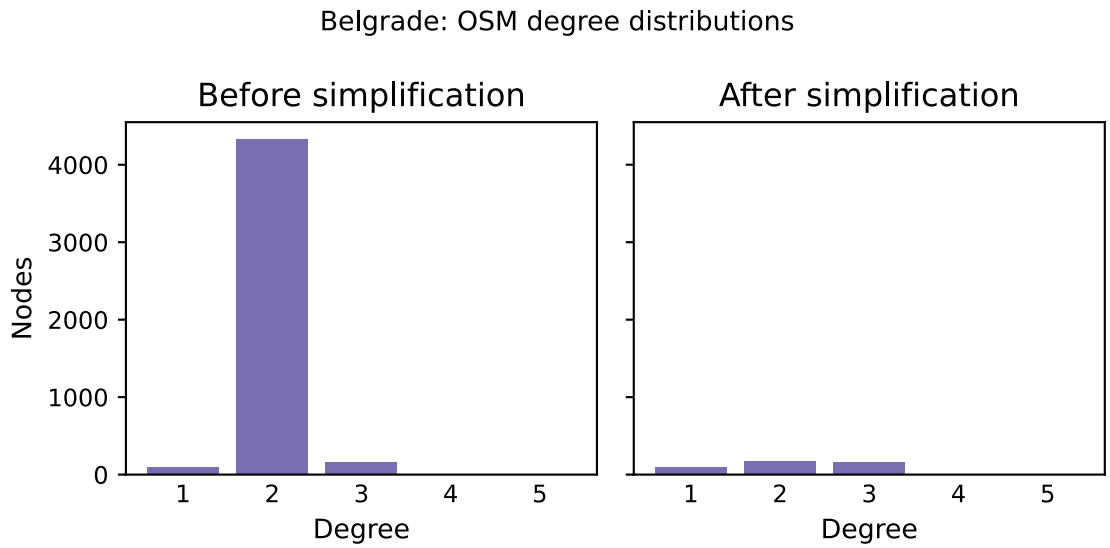
Method

The degree distributions before and after simplification are plotted below.

Interpretation

Typically, the degree distribution will go from high (before simplification) to low (after simplification) counts of degree two nodes, while it will not change for all other degrees (1, or 3 and higher). Further, the total number of nodes will see a strong decline. If the simplified graph still maintains a relatively high number of degree two nodes, or if the number of nodes with other degrees changes after the simplification, this might point to issues either with the graph conversion or with the simplification process.

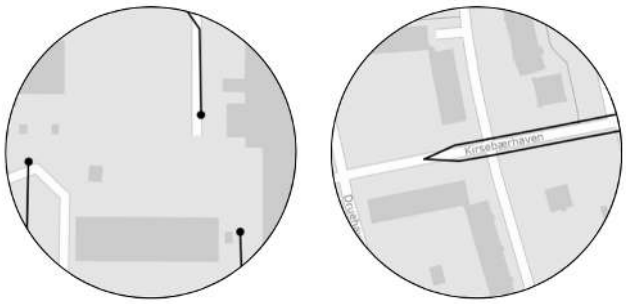
```
Simplifying the network decreased the number of edges by 94.1% and the number of nodes by 90.2%.
```



Dangling nodes

Dangling nodes are nodes of degree one, i.e. they have only one single edge attached to them. Most networks will naturally contain a number of dangling nodes. Dangling nodes can occur at actual dead-ends (representing a cul-de-sac) or at the endpoints of certain features, e.g. when a bicycle path ends in the middle of a street. However, dangling nodes can also occur as a data quality issue in case of over/undershoots (see next section). The number of dangling nodes in a network does to some extent also depend on the digitization method, as shown in the illustration below.

Therefore, the presence of dangling nodes is in itself not a sign of low data quality. However, a high number of dangling nodes in an area that is not known for containing many dead-ends can indicate digitization errors and problems with edge over/undershoots.



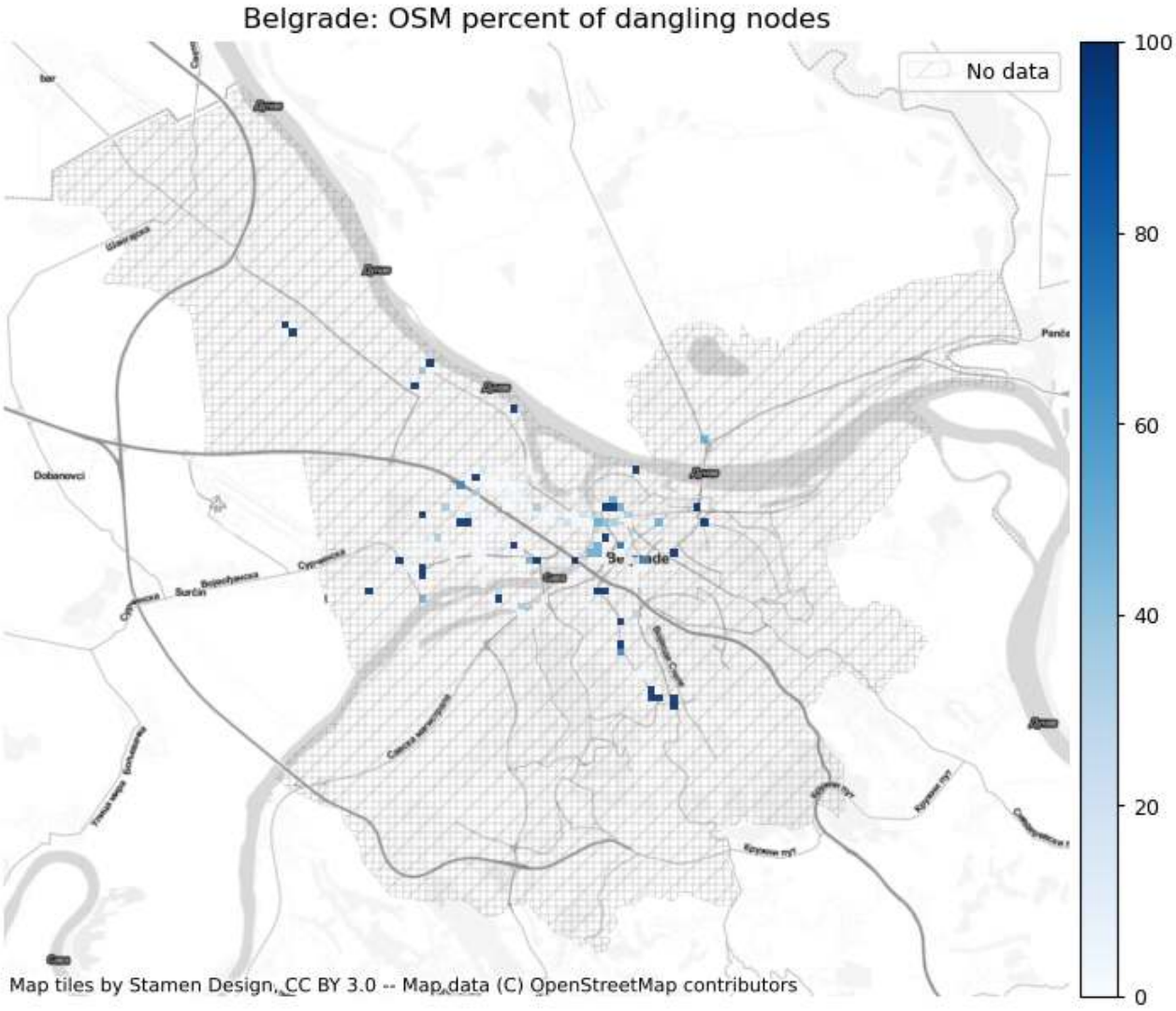
Left: Dangling nodes occur where road features end. Right: However, when separate features are joined at the end, there will be no dangling nodes.

Method

Below, a list of all dangling nodes is obtained with the help of `get_dangling_nodes`. Then, the network with all its nodes is plotted. The dangling nodes are shown in color, all other nodes are shown in black.

Interpretation

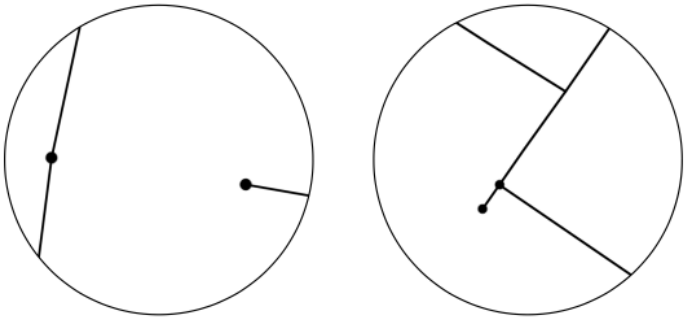
We recommend a visual analysis in order to interpret the spatial distribution of dangling nodes, with particular attention to areas of high dangling node density. It is important to understand where dangling nodes come from: are they actual dead-ends or digitization errors (e.g., over/undershoots)? A higher number of digitization errors points to lower data quality.




```
Interactive map saved at results/OSM/Belgrade/maps_interactive/danglingmap_osm.html
```

Under/overshoots

When two nodes in a simplified network are placed within a distance of a few meters, but do not share a common edge, it is often due to an edge over/undershoot or another digitizing error. An undershoot occurs when two features are supposed to meet, but instead are just in close proximity to each other. An overshoot occurs when two features meet and one of them extends beyond the other. See the image below for an illustration. For a more detailed explanation of over/undershoots, see the GIS Lounge website.



Left: Undershoots happen when two line features are not properly joined, for example at an intersection. Right: Overshoots refer to situations where a line feature extends too far beyond at intersecting line, rather than ending at the intersection.

Method

Undershoots: First, the `length_tolerance` (in meters) is defined in the cell below. Then, with `find_undershoots`, all pairs of dangling nodes that have a maximum of `length_tolerance` distance between them, are identified as undershoots, and the results are plotted.

Overshoots: First, the `length_tolerance` (in meters) is defined in the cell below. Then, with `find_overshoots`, all network edges that have a dangling node attached to them and that have a maximum length of `length_tolerance` are identifed as overshoots, and the results are plotted.

The method for over/undershoot detection is inspired by Neis et al. (2012).

Interpretation

Under/overshoots are not necessarily always a data quality issue - they might be instead an accurate representation of the network conditions or of the digitization strategy. For example, a cycle path might end abruptly soon after a turn, which results in an overshoot. Protected cycle paths are sometimes digitized in OSM as interrupted at intersections which results in intersection undershoots.

The interpretation of the impact of over/undershoots on data quality is context dependent. For certain applications, such as routing, overshoots do not present a particular challenge; they can, however, pose an issue for other applications such as network analysis, given that they skew the network structure. Undershoots, on the contrary, are a serious problem for routing applications, especially if only bicycle infrastructure is considered. They also pose a problem for network analysis, for example for any path-based metric, such as most centrality measures like betweenness centrality.

```
2 potential overshoots were identified using a length tolerance of 3 m.
2 potential undershoots were identified using a length tolerance of 3 m.
```

```
Interactive map saved at results/OSM/Belgrade/maps_interactive/underovershoots_3_3_osm.html
```

Missing intersection nodes

When two edges intersect without having a node at the intersection - and if neither edges are tagged as a bridge or a tunnel - there is a clear indication of a topology error.

Method

First, with the help of `check_intersection`, each edge which is not tagged as either tunnel or bridge is checked for any *crossing* with another edge of the network. If this is the case, the edge is marked as having an intersection issue. The number of intersection issues found is printed and the results are plotted for visual analysis. The method is inspired by Neis et al. (2012).

Interpretation

A higher number of intersection issues points to a lower data quality. However, it is recommended with a manual visual check of all intersection issues with a certain knowledge of the area, in order to determine the origin of intersection issues and confirm/correct/reject them.

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[33], line 1
----> 1 missing_nodes_edge_ids, edges_with_missing_nodes = eval_func.find_missing_intersections(
      2     osm_edges, "edge_id"
      3 )
      5 count_intersection_issues = (
      6     len(missing_nodes_edge_ids) / 2
      7 )  # The number of issues is counted twice since both intersecting osm_edges are returne
d
      9 print(
     10     f"{count_intersection_issues:.0f} place(s) appear to be missing an intersection node
or a bridge/tunnel tag."
     11 )

File ~/Library/CloudStorage/OneDrive-ITU/projects/BikeDNA-usecases/src/evaluation_functions.py:4
70, in find_missing_intersections(edges, edge_id_col, return_edges)
    461 """
    462 Detects topological errors in gdf with edges from OSM data.
    463 If two edges are intersecting (i.e. no node at intersection) and neither is tagged as a
bridge or a tunnel,
    464 it is considered an error in the data.
    465 """
    467 # Don't include tunnels or bridges
    468 edges_subset = edges.loc[
    469     ~(
--> 470         edges.tunnel.isin(
    471             ["yes", "Yes", True, "passage", "building_passage", "movable"]
    472         )
    473         | edges.bridge.isin(
    474             ["yes", "Yes", True, "passage", "building_passage", "movable"]
    475         )
    476     )
    477 ].copy()
    479 edges_subset["intersection_issues"] = edges_subset.apply(
    480     lambda x: check_crossing(row=x, gdf=edges_subset), axis=1
    481 )
    483 missing_nodes = list(
    484     edges_subset.loc[
    485         (edges_subset.intersection_issues.notna())
   (...)
    488     ][edge_id_col].values
    489 )

File ~/opt/anaconda3/envs/bikedna/lib/python3.11/site-packages/pandas/core/generic.py:5902, in N
DFrame.__getattr__(self, name)
   5895 if (
```

```
   5896     name not in self._internal_names_set
   5897     and name not in self._metadata
   5898     and name not in self._accessors
   5899     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5900 ):
   5901     return self[name]
-> 5902 return object.__getattribute__(self, name)

AttributeError: 'GeoDataFrame' object has no attribute 'tunnel'
```

Network components

Disconnected components do not share any elements (nodes/edges). In other words, there is no network path that could lead from one disconnected component to the other. As mentioned above, most real-world networks of bicycle infrastructure do consist of many disconnected components (Natera Orozco et al., 2020). However, when two disconnected components are very close to each other, it might be a sign of a missing edge or another digitizing error.

Method

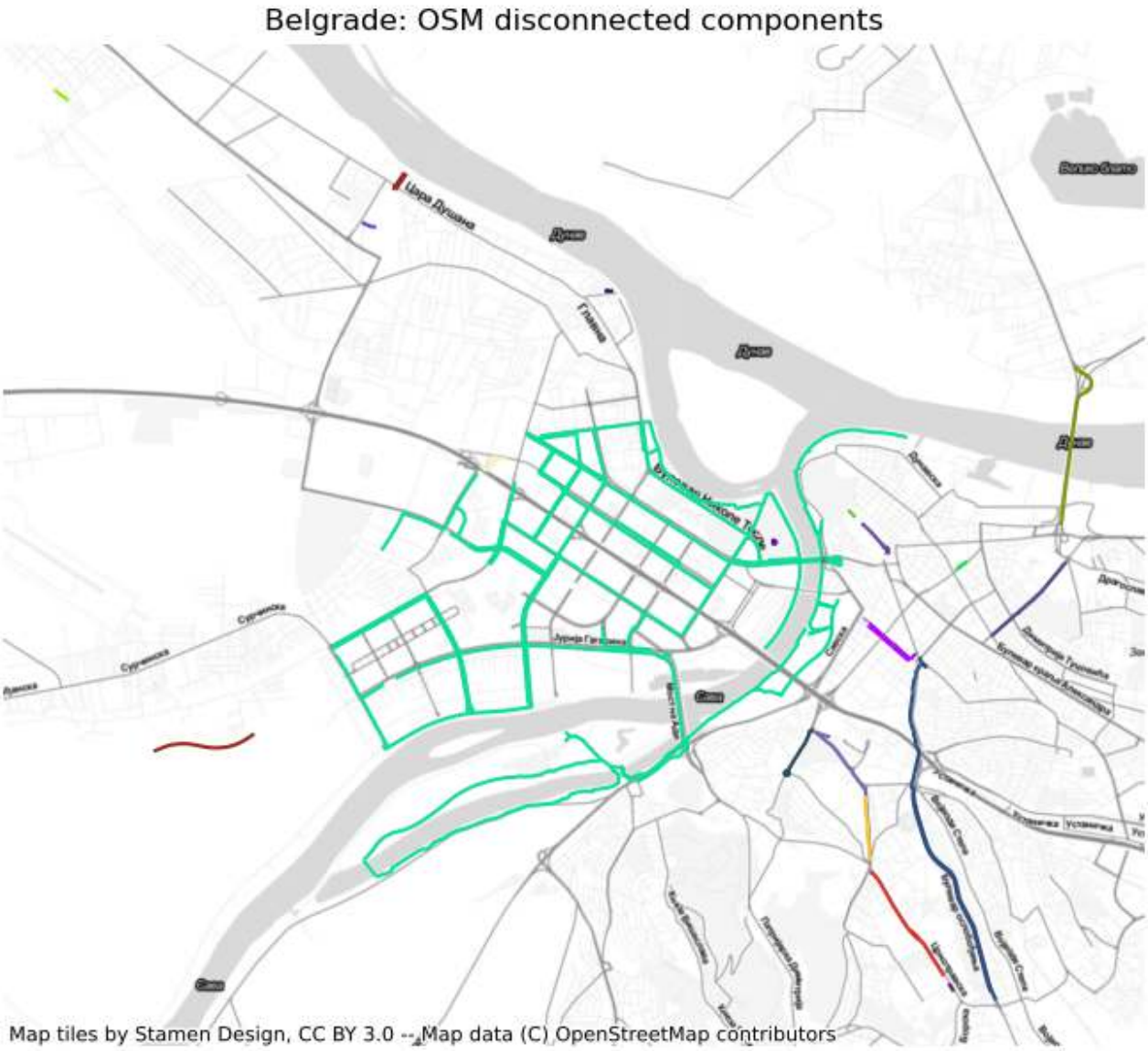
First, with the help of `return_components`, a list of all (disconnected) components of the network is obtained. The total number of components is printed and all components are plotted in different colors for visual analysis. Next, the component size distribution (with components ordered by the network length they contain) is plotted, followed by a plot of the largest connected component.

Interpretation

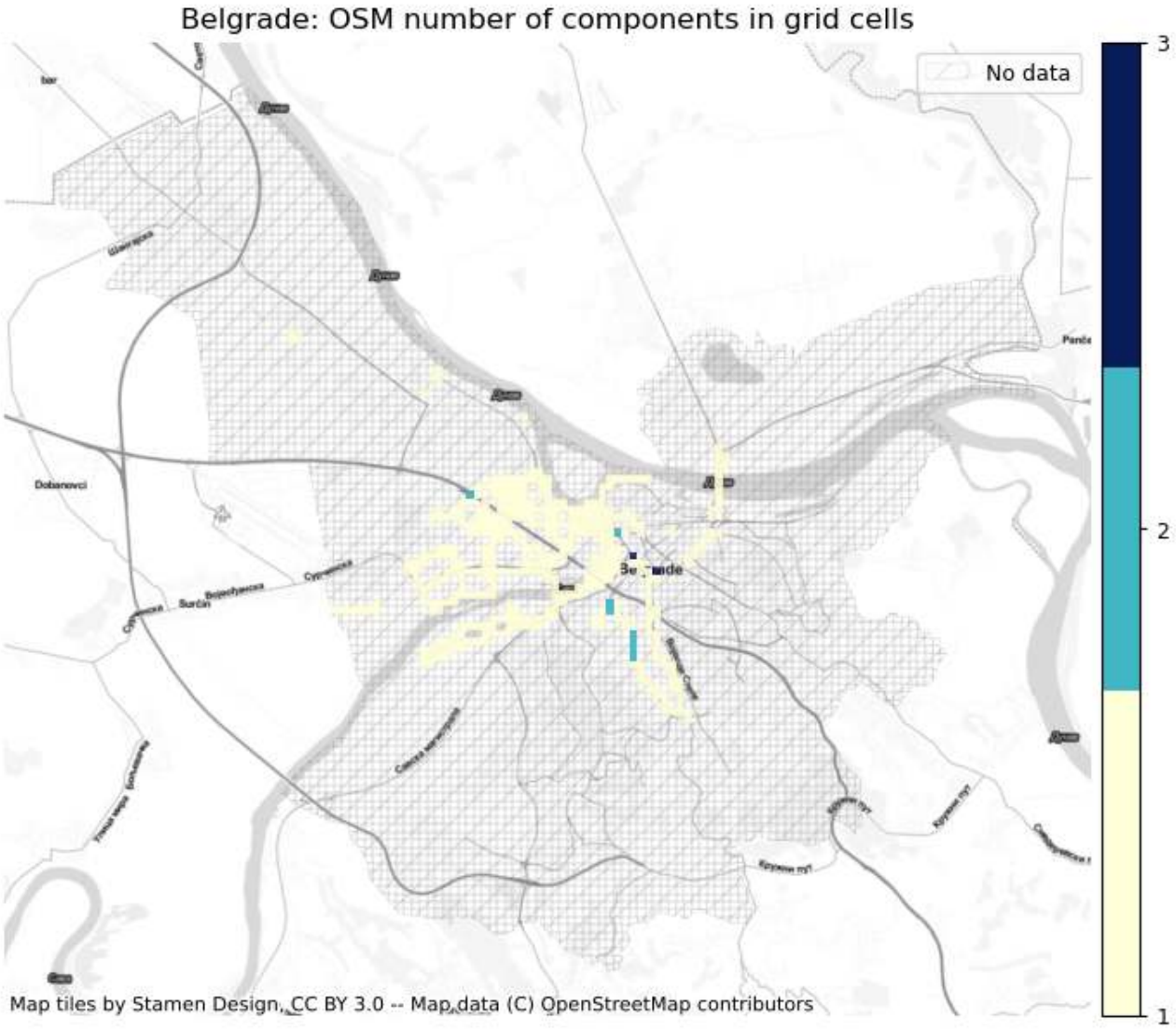
As with many of the previous analysis steps, knowledge of the area is crucial for a correct interpretation of component analysis. Given that the data represents the actual infrastructure accurately, bigger components indicate coherent network parts, while smaller components indicate scattered infrastructure (e.g., one single bicycle path along a street that does not connect to any other bicycle infrastructure). A high number of disconnected components in near vicinity of each other indicates digitization errors or missing data.

Disconnected components

```
The network in the study area has 24 disconnected components.
```



Components per grid cell

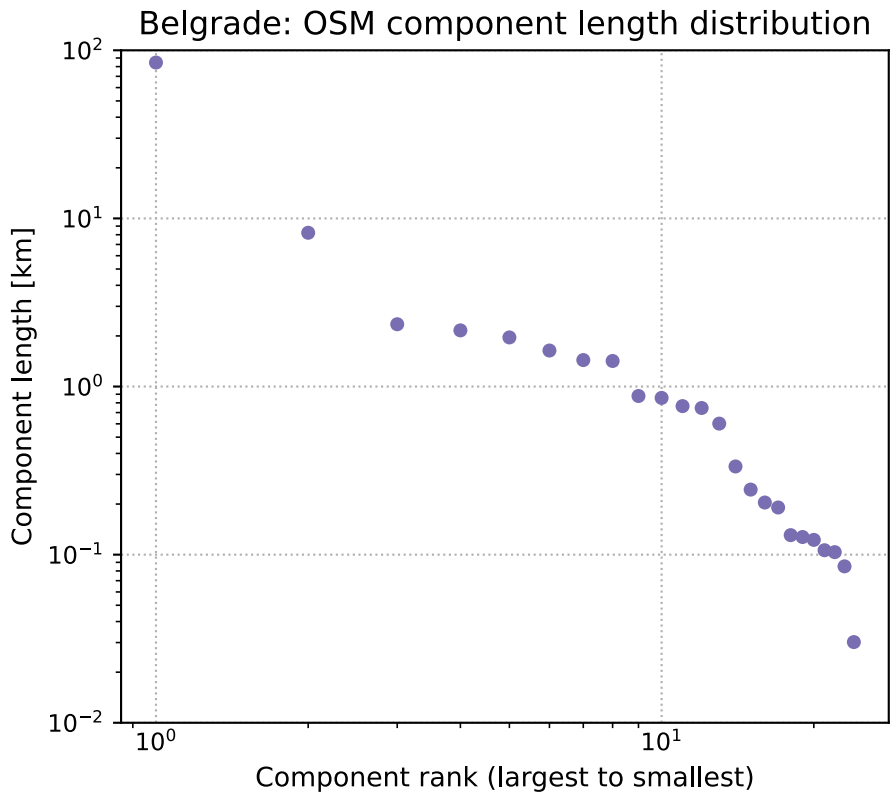


Component length distribution

The distribution of all network component lengths can be visualized in a so-called *Zipf plot*, which orders the lengths of each component by rank, showing the largest component's length on the left, then the second largest component's length, etc., until the smallest component's length on the right. When a Zipf plot follows a straight line in log-log scale, it means that there is a much higher chance to find small disconnected components than expected from traditional distributions (Clauset et al., 2009). This can mean that there has been no consolidation of the network, only piece-wise or random additions (Szell et al., 2022), or that the data itself suffers from many gaps and topology errors resulting in small disconnected components.

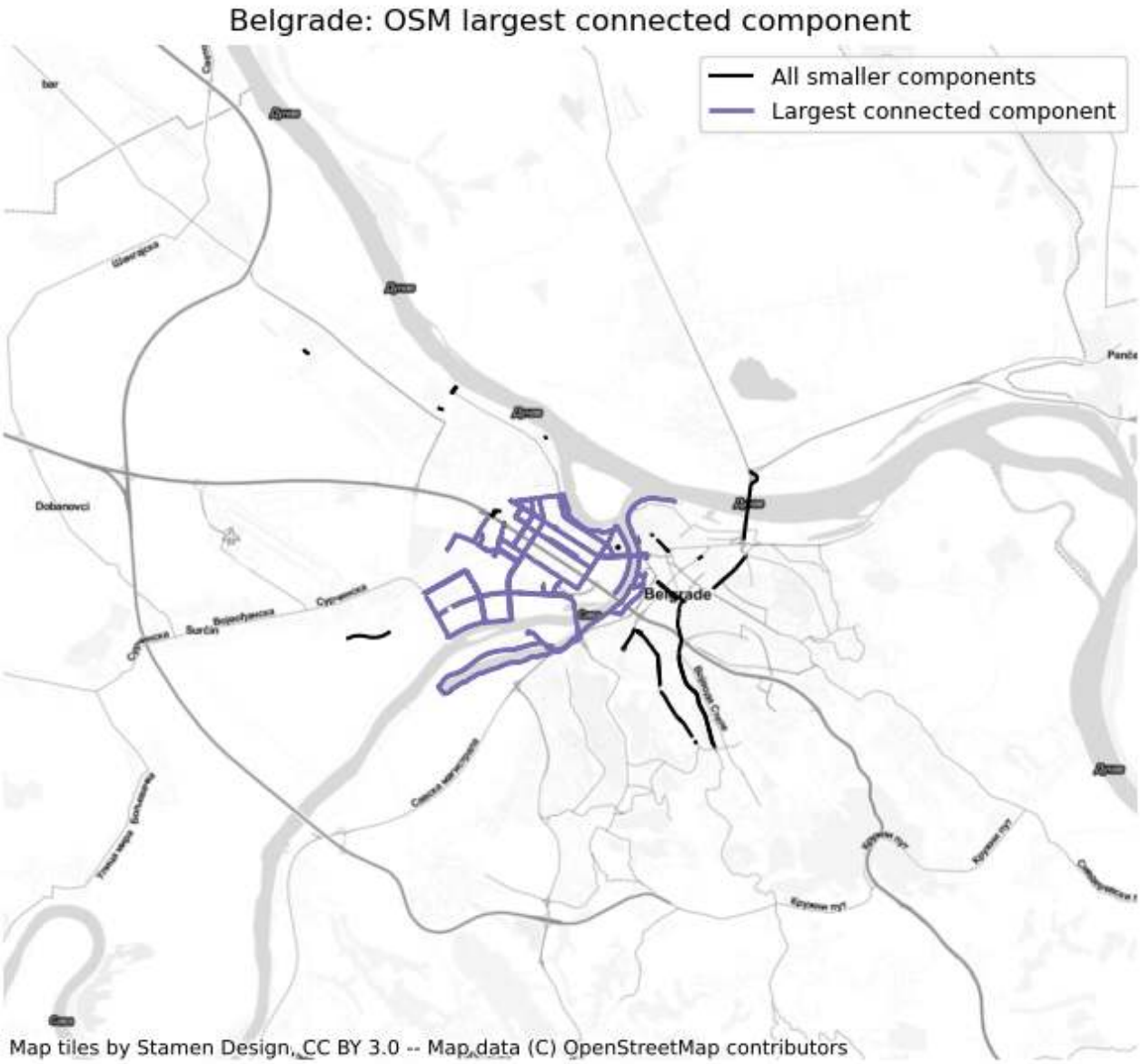
However, it can also happen that the largest connected component (the leftmost marker in the plot at rank 10^0) is a clear outlier, while the rest of the plot follows a different shape. This can mean that at the infrastructure level, most of the infrastructure has been connected to one large component, and that the data reflects this - i.e. the data is not suffering from gaps and missing links to a large extent.

Bicycle networks might also be somewhere inbetween, with several large components as outliers.



Largest connected component

The largest connected component contains 77.40% of the network length.



Missing links

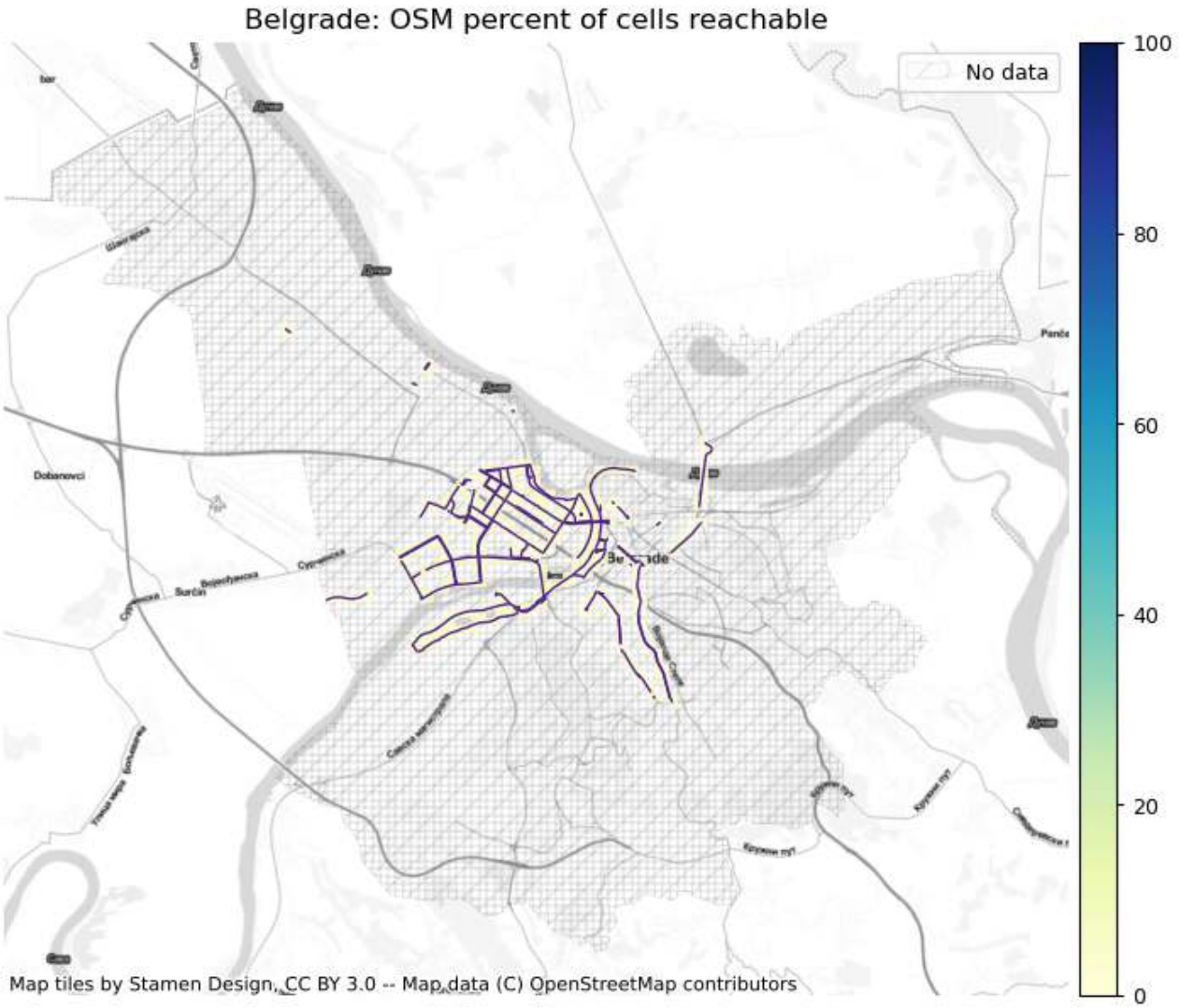
In the plot of potential missing links between components, all edges that are within the specified distance of an edge on another component are plotted. The gaps between disconnected edges are highlighted with a marker. The map thus highlights edges which, despite being in close proximity of each other, are disconnected and where it thus would not be possible to bike on cycling infrastructure between the edges.

```
Running analysis with component distance threshold of 10 meters.
```

```
Interactive map saved at results/OSM/Belgrade/maps_interactive/component_gaps_10_osm.html
```

Component connectivity

Here we visualize differences between how many cells can be reached from each cell. This is a crude measure for network connectivity but has the benefit of being computationally cheap and thus able to quickly highlight stark differences in network connectivity.



Summary

Intrinsic Quality Metrics - OSM data

Total infrastructure length (km)	112
Protected bicycle infrastructure density (m/km2)	236
Unprotected bicycle infrastructure density (m/km2)	38
Mixed protection bicycle infrastructure density (m/km2)	15

Bicycle infrastructure density (m/km2)	289
Nodes	450
Dangling nodes	95
Nodes per km2	1
Dangling nodes per km2	0
Incompatible tag combinations	0
Overshoots	2
Undershoots	2
Components	24
Length of largest component (km)	85
Largest component's share of network length	77%
Component gaps	7

Appendix A: config.yml

This notebook shows the content of `config.yml`, i.e. the parameters that were used to run the analysis.

```
{'area_name': 'Belgrade',
 'reference_name': None,
 'study_area': 'Belgrade',
 'study_crs': 'EPSG:8682',
 'plot_resolution': 'low',
 'bicycle_infrastructure_queries': {'A': "highway == 'cycleway'",
                                    'B': 'cycleway in '
                                         "['lane','track','opposite_lane','opposite_track','desi
gnated','crossing']",
                                    'C': 'cycleway_left in '
                                         "['lane','track','opposite_lane','opposite_track','desi
gnated','crossing']",
                                    'D': 'cycleway_right in '
                                         "['lane','track','opposite_lane','opposite_track','desi
gnated','crossing']",
                                    'E': 'cycleway_both in '
                                         "['lane','track','opposite_lane','opposite_track','desi
gnated','crossing']"},
 'simplify_tags_queries': {'centerline_false_bidirectional_true': ['highway == '
                                                                   "'cycleway' "
                                                                   '& '
                                                                   "(oneway=='no' "
                                                                   'or '
                                                                   "oneway_bicycle=='no')",
                                                                   'highway == '
                                                                   "'track' & "
                                                                   'bicycle == '
                                                                   "'designated' "
                                                                   '& '
                                                                   "(oneway=='no' "
                                                                   'or '
                                                                   'oneway_bicycle '
                                                                   "=='no')",
                                                                   'highway == '
                                                                   "'path' & "
                                                                   'bicycle == '
                                                                   "'designated' "
                                                                   '& '
                                                                   "(oneway=='no' "
                                                                   'or '
                                                                   'oneway_bicycle '
                                                                   "=='no')"],
                           'centerline_false_bidirectional_false': ['highway '
                                                                    '== '
                                                                    "'cycleway' "
                                                                    '& (oneway '
                                                                    "!='no' or "
                                                                    'oneway_bicycle '
                                                                    "!= 'no')",
                                                                    'highway '
                                                                    '== '
                                                                    "'track' & "
```

```
                                                  'bicycle '
                                                  '== '
                                                  "'designated' "
                                                  '& (oneway '
                                                  "!='no' or "
                                                  'oneway_bicycle '
                                                  "!='no')",
                                                  'highway '
                                                  "== 'path' "
                                                  '& bicycle '
                                                  '== '
                                                  "'designated' "
                                                  '& (oneway '
                                                  "!='no' or "
                                                  'oneway_bicycle '
                                                  "!='no')"],
                          'centerline_true_bidirectional_true': ['cycleway_left '
                                                                 'in '
                                                                 "['lane','track','opposite_lan
e','opposite_track','designated','crossing'] "
                                                                 'and '
                                                                 '(cycleway_right '
                                                                 'in '
                                                                 "['no','none','separate'] "
                                                                 'or '
                                                                 'cycleway_right.isnull() '
                                                                 'or '
                                                                 'cycleway_right '
                                                                 'not in '
                                                                 "['lane','track','opposite_lan
e','opposite_track','designated','crossing']) "
                                                                 'and '
                                                                 'oneway_bicycle '
                                                                 "=='no'",
                                                                 'cycleway_right '
                                                                 'in '
                                                                 "['lane','track','opposite_lan
e','opposite_track','designated','crossing'] "
                                                                 'and '
                                                                 '(cycleway_left '
                                                                 'in '
                                                                 "['no','none','separate'] "
                                                                 'or '
                                                                 'cycleway_left.isnull() '
                                                                 'or '
                                                                 'cycleway_left '
                                                                 'not in '
                                                                 "['lane','track','opposite_lan
e','opposite_track','designated','crossing']) "
                                                                 'and '
                                                                 'oneway_bicycle '
                                                                 "=='no'",
                                                                 'cycleway in '
                                                                 "['lane','track','opposite_lan
e','opposite_track','designated','crossing'] "
                                                                 'and '
                                                                 '(oneway_bicycle '
                                                                 "== 'no' or "
                                                                 'oneway_bicycle.isnull())',
                                                                 'cycleway_both '
                                                                 'in '
                                                                 "['lane','track','opposite_lan
e','opposite_track','designated','crossing'] "
```

```
                                                                'and '
                                                                '(oneway_bicycle '
                                                               "== 'no' or "
                                                                'oneway_bicycle.isnull())',
                                                                'cycleway_left '
                                                                'in '
                                                                "['lane','track','opposite_lan
e','opposite_track','designated','crossing'] "
                                                                'and '
                                                                'cycleway_right '
                                                                'in '
                                                               "['lane','track','opposite_lan
e','opposite_track','designated','crossing']"],
                           'centerline_true_bidirectional_false': ['cycleway_left '
                                                                  'in '
                                                                  "['lane','track','opposite_la
ne','opposite_track','designated','crossing'] "
                                                                  'and '
                                                                  '(cycleway_right '
                                                                  'in '
                                                                  "['no','none','separate'] "
                                                                  'or '
                                                                  'cycleway_right.isnull()  '
                                                                  'or '
                                                                  'cycleway_right '
                                                                  'not in '
                                                                  "['lane','track','opposite_la
ne','opposite_track','designated','crossing']) "
                                                                  'and '
                                                                  'oneway_bicycle '
                                                                  "!='no'",
                                                                  'cycleway_right '
                                                                  'in '
                                                                  "['lane','track','opposite_la
ne','opposite_track','designated','crossing'] "
                                                                  'and '
                                                                  '(cycleway_left '
                                                                  'in '
                                                                  "['no','none','separate'] "
                                                                  'or '
                                                                  'cycleway_left.isnull()  '
                                                                  'or '
                                                                  'cycleway_left '
                                                                  'not in '
                                                                  "['lane','track','opposite_la
ne','opposite_track','designated','crossing']) "
                                                                  'and '
                                                                  'oneway_bicycle '
                                                                  "!= 'no'",
                                                                  'cycleway '
                                                                  'in '
                                                                  "['lane','track','opposite_la
ne','opposite_track','designated','crossing'] "
                                                                  'and '
                                                                  'oneway_bicycle '
                                                                  "== 'yes'",
                                                                  'cycleway_both '
                                                                  'in '
                                                                  "['lane','track','opposite_la
ne','opposite_track','designated','crossing'] "
                                                                  'and '
                                                                  'oneway_bicycle '
                                                                  "== 'yes'"]},
```

```
'osm_bicycle_infrastructure_type': {'protected': ["highway == 'cycleway'",
                                                  'cycleway in '
                                                  "['track','opposite_track']",
                                                  'cycleway_left in '
                                                  "['track','opposite_track']",
                                                  'cycleway_right in '
                                                  "['track','opposite_track']",
                                                  'cycleway_both in '
                                                  "['track','opposite_track']"],
                                    'unprotected': ['cycleway in '
                                                    "['lane','opposite_lane','crossing']",
                                                    'cycleway_left in '
                                                    "['lane','opposite_lane','crossing']",
                                                    'cycleway_right in '
                                                    "['lane','opposite_lane','crossing']",
                                                    'cycleway_both in '
                                                    "['lane','opposite_lane','crossing']"],
                                    'unknown': ["cycleway in ['designated']",
                                                'cycleway_left in '
                                                "['designated']",
                                                'cycleway_right in '
                                                "['designated']",
                                                'cycleway_both in '
                                                "['designated']"]},
'osm_way_tags': ['access',
                 'barrier',
                 'bridge',
                 'bicycle',
                 'bicycle_road',
                 'crossing',
                 'cycleway',
                 'cycleway:left',
                 'cycleway:right',
                 'cycleway:both',
                 'cycleway:buffer',
                 'cycleway:left:buffer',
                 'cycleway:right:buffer',
                 'cycleway:both:buffer',
                 'cycleway:width',
                 'cycleway:left:width',
                 'cycleway:right:width',
                 'cycleway:both:width',
                 'cycleway:surface',
                 'foot',
                 'footway',
                 'highway',
                 'incline',
                 'junction',
                 'layer',
                 'lit',
                 'maxspeed',
                 'maxspeed:advisory',
                 'moped',
                 'moter_vehicle',
                 'motorcar',
                 'name',
                 'oneway',
                 'oneway:bicycle',
                 'osm_id',
                 'segregated',
                 'surface',
                 'tracktype',
                 'tunnel',
```

```
                'width'],
'existing_tag_analysis': {'surface': {'true_geometries': ['surface',
                                                          'cycleway_surface'],
                                      'centerline': ['cycleway_surface']},
                          'width': {'true_geometries': ['width',
                                                        'cycleway_width',
                                                        'cycleway_left_width',
                                                        'cycleway_right_width',
                                                        'cycleway_both_width'],
                                    'centerline': ['cycleway_width',
                                                   'cycleway_left_width',
                                                   'cycleway_right_width',
                                                   'cycleway_both_width']},
                          'speedlimit': {'all': ['maxspeed']},
                          'lit': {'all': ['lit']}},
'incompatible_tags_analysis': {'bicycle_infrastructure': {'yes': [['bicycle',
                                                                   'no'],
                                                                  ['bicycle',
                                                                   'dismount'],
                                                                  ['car',
                                                                   'yes']]}},
'grid_cell_size': 250,
'reference_geometries': 'will_not_be_used',
'bidirectional': 'will_not_be_used',
'ref_bicycle_infrastructure_type': {'protected': ['will_not_be_used']},
'reference_id_col': 'will_not_be_used'}
```